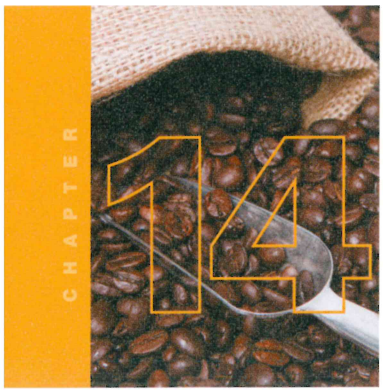


14

고급 스윙 컴포넌트

Objectives

- JMenuBar, JMenu, JMenuItem을 이용한 메뉴를 작성할 수 있다.
- 메뉴 아이템의 Action 이벤트를 처리하는 방법을 안다.
- JToolBar를 이용한 툴바를 작성할 수 있다.
- 스윙 컴포넌트에 툴팁을 달 수 있다.
- JDialog를 이용한 다이얼로그를 만들 수 있다.
- JOptionPane을 이용한 입력 다이얼로그, 확인 다이얼로그, 메시지 다이얼로그를 활용할 수 있다.
- JFileChooser를 이용한 파일 다이얼로그를 만들 수 있다.
- JColorChooser를 이용한 컬러 다이얼로그를 만들 수 있다.
- JTabbedPane을 이용한 탭팬을 작성할 수 있다.
- 자바 오디오 API를 알고 오디오 재생, 중지 등 오디오를 제어할 수 있다.



CHAPTER 14

JAVA PROGRAMMING

고급 스윙 컴포넌트

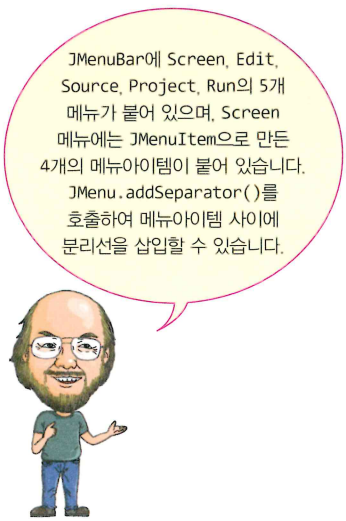
14.1 메뉴 만들기

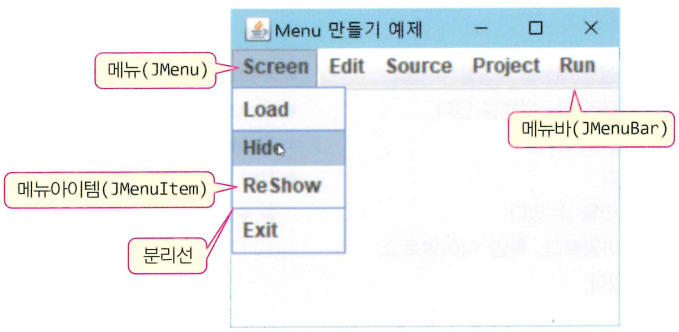
메뉴 구성

메뉴

스윙 프로그램에 메뉴를 만들어 달아보자. [그림 14-1]은 메뉴를 만든 예를 보여준다. 메뉴를 만들기 위해 다음 요소가 필요하다.

- JMenuBar
  메뉴바의 기능을 구현한 클래스이다. 이곳에 JMenu로 만든 메뉴를 여러 개 단다.
- JMenu
  하나의 메뉴를 구현한 클래스이다. 이곳에 JMenuItem으로 만든 메뉴아이템을 여러 개 단다.
- JMenuItem
  하나의 메뉴아이템을 구현한 클래스이다.





[그림 14-1] 스윙으로 만든 메뉴 예

• 분리선

메뉴아이템 사이의 분리선으로서 separator라고 부르며, JMenu의 addSeparator() 메소드를 호출하면 메뉴에 분리선이 삽입된다.

addSeparator()

메뉴 만들기

이제 직접 메뉴를 만들어보자. 메뉴를 만드는 과정은 단계별로 다음과 같으며 이 과정을 [그림 14-2]에 표현하였다.

1. 메뉴바 만들기

```
JMenuBar mb = new JMenuBar();
```

JMenuBar

2. 메뉴를 만들고 메뉴바에 붙이기

메뉴 이름은 문자열로 JMenu의 생성자에 전달한다. 메뉴바에 메뉴를 붙일 때 JMenuBar의 add() 메소드를 이용한다. "Screen" 메뉴를 만들어 메뉴바에 붙이는 코드는 다음과 같다.

JMenu

```
JMenu screenMenu = new JMenu("Screen"); // Screen 메뉴 생성
mb.add(screenMenu); // 메뉴바에 Screen 메뉴 붙이기
```

3. 메뉴아이템을 생성하여 메뉴에 붙이기

메뉴아이템의 이름은 문자열로서 JMenuItem의 생성자에 전달한다. 메뉴아이템을 메뉴에 붙일 때는 JMenu의 add() 메소드를 이용한다. screenMenu에 메뉴아이템과 분리선을 붙이는 코드는 다음과 같다.

add()

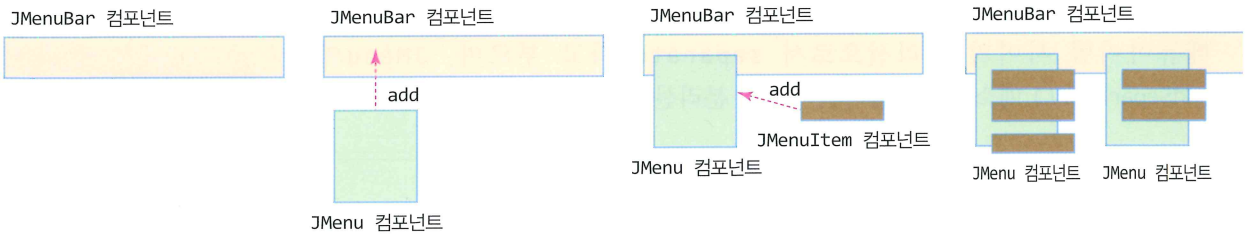
```
screenMenu.add(new JMenuItem("Load"));
screenMenu.add(new JMenuItem("Hide"));
screenMenu.add(new JMenuItem("ReShow"));
screenMenu.addSeparator(); // ReShow 메뉴아이템 다음에 분리선 삽입
screenMenu.add(new JMenuItem("Exit"));
```

4. 메뉴바를 프레임에 붙이기

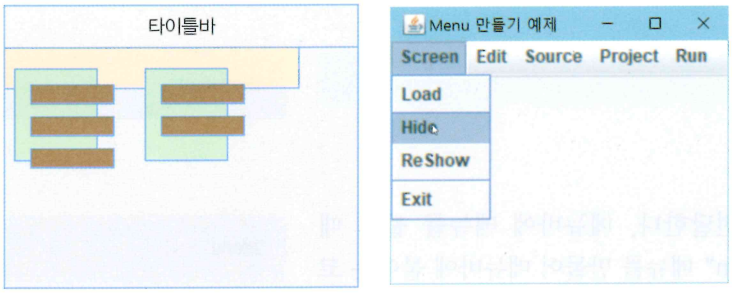
JFrame의 setJMenuBar() 메소드를 이용하여 메뉴바를 붙이는 예는 다음과 같다.

setJMenuBar()

```
frame.setJMenuBar(mb);
```



(1) JMenuBar 컴포넌트 생성 (2) JMenu 컴포넌트를 생성하여 JMenuBar에 붙인다. (3) JMenuItem 컴포넌트를 생성하여 JMenu에 붙인다. (3′) 여러 개의 메뉴와 메뉴아이템을 생성한다.



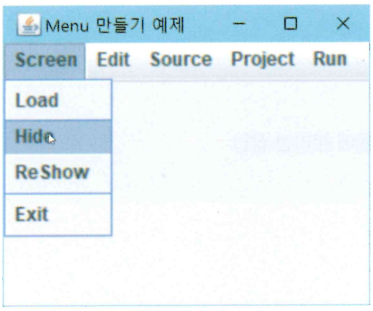
(4) JMenuBar 컴포넌트를 JFrame에 붙인다.

[그림 14-2] 스윙 메뉴 만드는 과정 예



예제 14-1 메뉴 만들기

그림과 같이 Screen, Edit, Source, Project, Run의 5개 메뉴를 가지며, Screen 메뉴에만 4개의 메뉴아이템과 하나의 분리선(separator)을 가지도록 프로그램을 작성하라.



```
1  import javax.swing.*;
2
3  public class MenuEx extends JFrame {
4     public MenuEx() {
5        setTitle("Menu 만들기 예제");
6        createMenu(); // 메뉴 생성, 프레임에 삽입
7        setSize(250,200);
8        setVisible(true);
9     }
10
11    // 메뉴를 만들어 프레임에 삽입한다.
12    private void createMenu() {
13       JMenuBar mb = new JMenuBar(); // 메뉴바 생성
14       JMenu screenMenu = new JMenu("Screen"); // Screen 메뉴 생성
15
16       // Screen 메뉴에 메뉴아이템 생성 삽입
17       screenMenu.add(new JMenuItem("Load"));
18       screenMenu.add(new JMenuItem("Hide"));
19       screenMenu.add(new JMenuItem("ReShow"));
20       screenMenu.addSeparator(); // 분리선 삽입
21       screenMenu.add(new JMenuItem("Exit"));
22
23       // 메뉴바에 메뉴 삽입
24       mb.add(screenMenu); // Screen 메뉴 삽입
25       mb.add(new JMenu("Edit")); // Edit 메뉴 생성 삽입
26       mb.add(new JMenu("Source")); // Source 메뉴 생성 삽입
27       mb.add(new JMenu("Project")); // Project 메뉴 생성 삽입
28       mb.add(new JMenu("Run")); // Run 메뉴 생성 삽입
29
30       // 메뉴바를 프레임에 부착
31       setJMenuBar(mb);
32    }
33
34    public static void main(String [] args) {
35       new MenuEx();
36    }
37 }
```

메뉴바를 프레임에 붙임.
비로소 메뉴가 보인다.

메뉴아이템에 Action 이벤트 달기

Action 이벤트

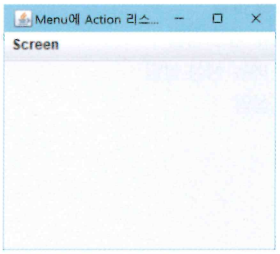
많은 경우 메뉴아이템은 사용자로부터 작업 명령을 받는 데 사용된다. 사용자가 메뉴아이템을 선택하면 Action 이벤트가 발생한다. 다음은 메뉴아이템에 Action 이벤트를 처리하기 위해 Action 리스너를 등록하는 코드 예이다.

```
JMenuItem item = new JMenuItem("Load");
item.addActionListener(new MenuActionListener()); // 메뉴아이템에 Action 리스너 등록
screenMenu.add(item);
...
class MenuActionListener implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        // 사용자가 Load 메뉴아이템을 선택하는 경우 처리할 작업 구현
    }
}
```

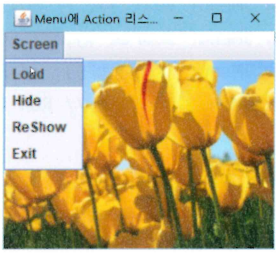


예제 14-2 메뉴아이템 선택시 Action 리스너 활용

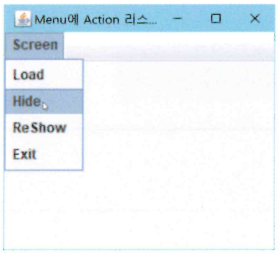
그림과 같이 Screen 메뉴에 4개의 메뉴아이템을 만들고, Load 메뉴아이템을 선택하면 이미지를 로딩하여 출력하고, Hide 메뉴아이템을 선택하면 이미지를 보이지 않게 하며, ReShow 메뉴아이템을 선택하면 숨겨진 이미지를 다시 보이게 하고, Exit 메뉴아이템을 선택하면 프로그램을 종료하도록 Action 리스너를 작성하라.



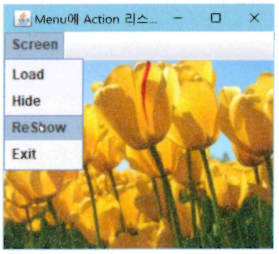
초기 상태



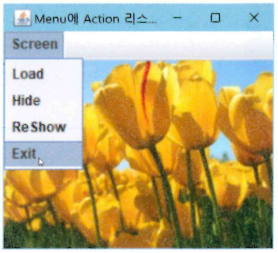
Load 메뉴아이템 선택하면
레이블을 이용하여 이미지 출력



Hide 메뉴아이템 선택으로
이미지를 보이지 않게 함



ReShow 메뉴아이템 선택으로
숨겨진 이미지가 다시 보이게 함



Exit 메뉴아이템 선택하면
프로그램 종료

```
1  import javax.swing.*;
2  import java.awt.event.*;
3  import java.awt.*;
4
5  public class MenuActionEventEx extends JFrame {
6     private JLabel imgLabel = new JLabel(); // 빈 이미지를 가진 레이블
7     public MenuActionEventEx() {
8        setTitle("Menu에 Action 리스너 만들기 예제");
9        createMenu();
10       getContentPane().add(imgLabel, BorderLayout.CENTER);
11       setSize(250,220);
12       setVisible(true);
13    }
14
15    private void createMenu() { // 메뉴바와 Screen 메뉴 생성. Screen 메뉴에 4개의
                                  메뉴아이템 삽입
16       JMenuBar mb = new JMenuBar(); // 메뉴바 생성
17       JMenuItem [] menuItem = new JMenuItem [4];
18       String[] itemTitle = {"Load", "Hide", "ReShow", "Exit"};
19       JMenu screenMenu = new JMenu("Screen");
20
21       // 4개의 메뉴아이템을 Screen 메뉴에 삽입한다.
22       MenuActionListener listener = new MenuActionListener(); // Action 리스너 생성
23       for(int i=0; i<menuItem.length; i++) {
24          menuItem[i] = new JMenuItem(itemTitle[i]); // 메뉴아이템 생성
25          menuItem[i].addActionListener(listener); // 메뉴아이템에 Action 리스너 등록
26          screenMenu.add(menuItem[i]); // 메뉴아이템을 Screen 메뉴에 삽입
27       }
28       mb.add(screenMenu); // 메뉴바에 Screen 메뉴 삽입
29       setJMenuBar(mb); // 메뉴바를 프레임에 부착
30    }
31
32    class MenuActionListener implements ActionListener { // 메뉴아이템 처리
                                                            Action 리스너
33       public void actionPerformed(ActionEvent e) {
34          String cmd = e.getActionCommand(); // 사용자가 선택한 메뉴아이템의 문자열 리턴
35          switch(cmd) { // 메뉴 아이템의 종류 구분
36             case "Load" :
37                if(imgLabel.getIcon() != null)
38                   return; // 이미 로딩되었으면 리턴
39                imgLabel.setIcon(new ImageIcon("images/img.jpg"));
40                break;
41             case "Hide" :
```

이미지를 출력할 레이블

이미지 레이블을 콘텐트팬의 중앙에 부착

imgLabel에 이미지 로딩 및 출력

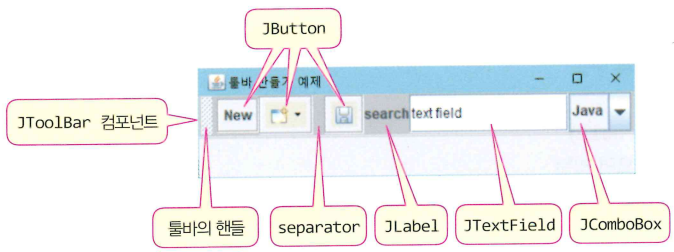
```
42                imgLabel.setVisible(false);   // imgLabel이 보이지 않도록 변경
43                break;
44             case "ReShow" :
45                imgLabel.setVisible(true);    // imgLabel이 보이도록 변경
46                break;
47             case "Exit" :
48                System.exit(0);
49                break;
50          }
51       }
52    }
53    public static void main(String [] args) {
54       new MenuActionEventEx();
55    }
56 }
```

14.2 툴바

JToolBar

JToolBar
툴바

JToolBar는 툴바를 구현한 컴포넌트이다. 툴바는 [그림 14-3]과 같은 바(bar) 모양의 컨테이너로, 다양한 스윙 컴포넌트를 담아 아이콘 형태의 메뉴를 제공하기 위해 사용된다.





[그림 14-3] JToolBar로 구현된 툴바

툴바는 BorderLayout 배치관리자를 가진 컨테이너에만 부착되며, 동(EAST), 서(WEST), 남(SOUTH), 북(NORTH)의 아무 곳이나 부착 가능하다. 사용자는 실행 중 [그림 14-3]에 보이는 툴바의 핸들을 마우스로 잡아 드래깅하면, 4개의 영역 중 아무 곳이나 옮길 수 있다. 뿐만 아니라 툴바의 핸들을 드래깅하여 4개의 영역 중 어디에도 붙이지 않고 독립적으로 떠다니게(floatable) 할 수도 있다.

BorderLayout 배치관리
핸들

툴바 만들기

툴바를 만드는 과정은 메뉴를 만드는 과정과 비슷하며 다음과 같이 간단하다.

1. JToolBar 객체 생성

JToolBar의 생성자에 툴바 이름을 전달하여 툴바를 만든다. 아래는 툴바 이름이 "Kitae Menu"인 툴바를 생성하는 코드이다.

```
JToolBar toolBar = new JToolBar("Kitae Menu");
```

2. 메뉴로 사용할 컴포넌트를 툴바에 삽입

모든 스윙 컴포넌트가 JToolBar에 삽입 가능하며, 컴포넌트 사이에 분리 공간(separator)을 삽입할 수 있다. 예를 들면 다음과 같다.

분리 공간

```
toolBar.add(new JButton("New")); // 버튼 삽입
toolBar.addSeparator(); // 분리 공간 삽입
toolBar.add(new JTextField("text field")); // 텍스트필드 삽입
```

3. 툴바를 컨테이너에 삽입

툴바는 BorderLayout 배치관리자가 설정된 컨테이너에만 부착된다. 다음은 컨테이너 container의 NORTH 영역에 툴바를 부착하는 사례이다.

```
container.add(toolBar, BorderLayout.NORTH); // 컨테이너의 NORTH에 툴바 부착
```

툴바 이동 제어

툴바는 기본적으로 사용자가 핸들을 드래깅하여 다른 영역으로 이동할 수 있지만, 다음과 같이 JToolBar.setFloatable(false)를 호출하면 툴바의 핸들이 보이지 않게 되며 사용자는 툴바를 다른 곳으로 이동할 수 없다.

핸들 드래깅

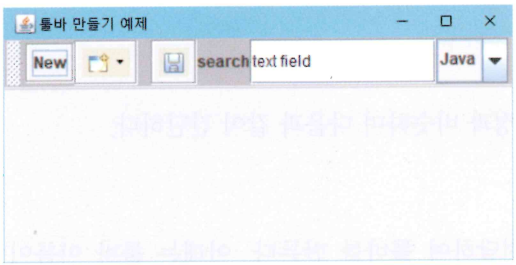
```
toolBar.setFloatable(false); // 툴바 이동 불가능. true이면 이동 가능
```



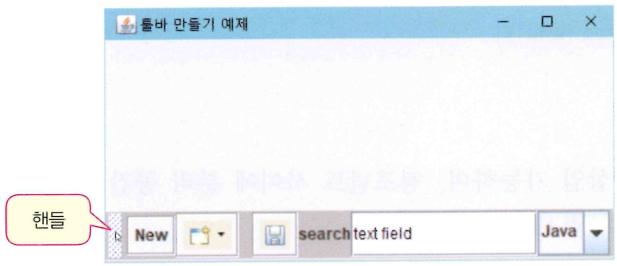
예제 14-3 툴바 만들기

다음 그림과 같은 툴바를 작성하라.

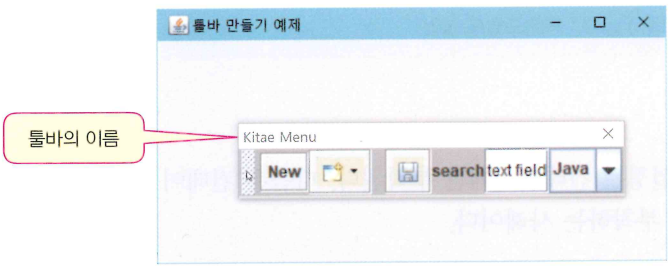
툴바 핸들을 드래깅하여 툴바를 이동시킬 수 있다. 아래 그림과 같이 툴바를 컨테이너의 동, 서, 남, 북이 아닌 곳으로 움직이면 임의의 위치에 둘 수 있다. 이때 툴바의 이름("Kitae Menu")이 나타난다.



초기 컨텐트팬의 NORTH에 부착된 툴바



핸들을 드래그하여 SOUTH로 이동



핸들을 드래그하여 임의의 위치로 이동

```
1  import javax.swing.*;
2  import java.awt.*;
3
4  public class ToolBarEx extends JFrame {
5     private Container contentPane;
6
7     public ToolBarEx() {
```

```
8         setTitle("툴바 만들기 예제");
9         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
10        contentPane = getContentPane();
11        createToolBar();
12        setSize(400,200);
13        setVisible(true);
14     }
15
16     // 툴바를 생성하여 컨텐츠팬에 부착한다.
17     private void createToolBar() {
18        // 툴바 생성
19        JToolBar toolBar = new JToolBar("Kitae Menu"); // 툴바의 이름 "Kitae Menu"
20        toolBar.setBackground(Color.LIGHT_GRAY);
21
22        // 툴바에 메뉴로 사용할 컴포넌트를 삽입한다.
23        toolBar.add(new JButton("New"));
24        toolBar.add(new JButton(new ImageIcon("images/open.jpg")));
25        toolBar.addSeparator();
26        toolBar.add(new JButton(new ImageIcon("images/save.jpg")));
27        toolBar.add(new JLabel("search"));
28        toolBar.add(new JTextField("text field"));
29
30        JComboBox<String> combo = new JComboBox<String>();
31        combo.addItem("Java");
32        combo.addItem("C#");
33        combo.addItem("C");
34        combo.addItem("C++");
35        toolBar.add(combo);
36
37        // 툴바를 컨텐츠팬의 NORTH에 부착한다.
38        contentPane.add(toolBar, BorderLayout.NORTH);
39     }
40
41     public static void main(String [] args) {
42        new ToolBarEx();
43     }
44  }
```

이 라인 밑에 toolBar.setFloatable(false);를 삽입하면 툴바를 이동시킬 수 없게 된다.

컨텐트팬은 디폴트 배치관리자가 BorderLayout이다.

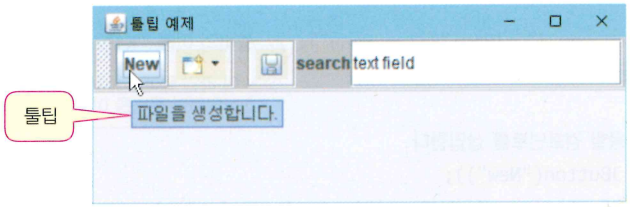
14.3 툴팁

툴팁 생성 및 달기

툴팁

스윙 컴포넌트에 마우스를 올리면 잠깐 나타났다가 사라지는 문자열을 툴팁(tooltip)이라고 부른다. 툴팁은 [그림 14-4]와 같이 스윙 컴포넌트의 기능을 설명하기 위한 목적이다.





[그림 14-4] 툴팁 예

setToolTipText(String msg)

모든 스윙 컴포넌트들이 툴팁을 가질 수 있으며 생성 방법 또한 단순하다. 간단히 JComponent의 setToolTipText(String msg) 메소드를 호출하면, 문자열(msg)을 컴포넌트의 툴팁으로 등록한다. [그림 14-4]의 New 버튼에 툴팁을 다는 예는 다음과 같다.

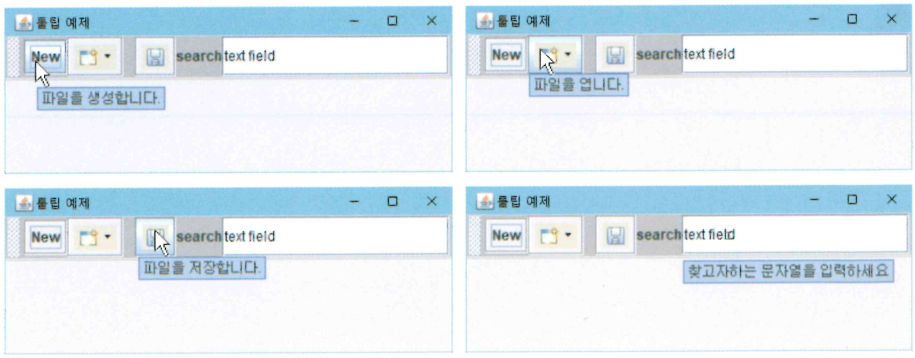
```
JButton b = new JButton("New");
b.setToolTipText("파일을 생성합니다."); // 버튼에 툴팁 등록
```



예제 14-4 툴팁 달기

예제 14-3의 '툴바 만들기' 응용프로그램의 툴바에 부착된 스윙 컴포넌트에 툴팁을 달아보라.

New 버튼에는 "파일을 생성합니다."를, 그 다음 버튼에는 "파일을 엽니다."를, 그 다음 버튼에는 "파일을 저장합니다."를, 텍스트필드 컴포넌트에는 "찾고자 하는 문자열을 입력하세요."를 툴팁으로 달아본다.



```
1  import javax.swing.*;
2  import java.awt.*;
3
4  public class ToolTipEx extends JFrame {
5     private Container contentPane;
6
7     public ToolTipEx() {
8        setTitle("툴팁 예제");
9        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
10       contentPane = getContentPane();
11       createToolBar();
12       setSize(400,150);
13       setVisible(true);
14    }
15
16    // 툴바를 생성하여 컨텐트팬의 NORTH에 부착한다.
17    private void createToolBar() {
18       JToolBar bar = new JToolBar("Kitae Menu"); // 툴바 생성
19       bar.setBackground(Color.LIGHT_GRAY);
20
21       JButton newBtn = new JButton("New");
22       newBtn.setToolTipText("파일을 생성합니다."); // New 버튼에 툴팁 달기
23       bar.add(newBtn);
24
25       JButton openBtn = new JButton(new ImageIcon("images/open.jpg"));
26       openBtn.setToolTipText("파일을 엽니다."); // 열기 버튼에 툴팁 달기
27       bar.add(openBtn);
28       bar.addSeparator();
29
30       JButton saveBtn = new JButton(new ImageIcon("images/save.jpg"));
31       saveBtn.setToolTipText("파일을 저장합니다."); // 저장 버튼에 툴팁 달기
32       bar.add(saveBtn);
33       bar.add(new JLabel("search"));
34
35       JTextField tf = new JTextField("text field");
36       tf.setToolTipText("찾고자하는 문자열을 입력하세요"); // 텍스트필드에 툴팁 달기
37       bar.add(tf);
38       contentPane.add(bar, BorderLayout.NORTH);
39    }
40
41    public static void main(String [] args) {
42       new ToolTipEx();
43    }
44 }
```

setToolTipText()를 이용하여 쉽게 툴팁을 달 수 있다.

ToolTipManager와 툴팁 시간 제어

ToolTipManager

툴팁은 마우스를 올리면 나타났다가 일정 시간 후에 사라진다. ToolTipManager 클래스를 이용하여 툴팁과 관련된 시간을 제어할 수 있다. 툴팁에 관한 시간 제어는 모든 툴팁에 일괄적으로 적용되며 각 툴팁마다 관련 시간을 따로 지정할 수 없다.

● ToolTipManager 객체 얻기

툴팁 시간을 지정하려면, 우선 다음과 같이 ToolTipManager 객체를 얻어야 한다.

```
ToolTipManager m = ToolTipManager.sharedInstance(); // ToolTipManager 객체 리턴
```

● 툴팁 활성화

setEnabled(boolean b)

ToolTipManager의 setEnabled(boolean b) 메소드를 이용하면, 툴팁이 나타나도록 할지 나타나지 않도록 할지 제어할 수 있다. 이러한 툴팁의 활성화/비활성화는 응용프로그램에 속한 모든 툴팁에 일괄적으로 적용되며 컴포넌트별 별로 제어할 수 없다. 다음은 응용프로그램의 모든 툴팁을 비활성화시키는 코드이다.

```
m.setEnabled(false); // 툴팁이 나타나지 않도록 설정
```

위의 문장에서 인자를 true를 주면 모든 툴팁이 활성화된다.

● 초기 툴팁 출력 시간 제어

setInitialDelay(int millis)

ToolTipManager의 setInitialDelay(int millis) 메소드를 호출하면, 마우스가 컴포넌트 위에 올라간 후, 툴팁이 출력되는 데까지 걸리는 시간을 정할 수 있다. 인자는 밀리초 단위의 시간이다. 이 시간 역시 모든 컴포넌트에 일괄적으로 적용된다.

```
m.setInitialDelay(1000); // 마우스가 올라온 지 1000ms 후 툴팁이 출력되도록 설정
```

● 툴팁 지속 시간 제어

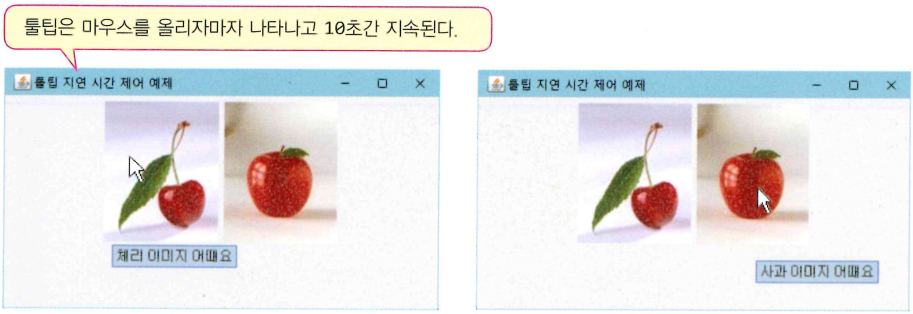
setDismissDelay(int millis)

ToolTipManager의 setDismissDelay(int millis) 메소드를 호출하면, 출력된 툴팁이 지속되는 시간을 정할 수 있다. 인자는 밀리초 단위의 시간이다.

```
m.setDismissDelay(1000); // 툴팁이 출력되어 있는 지속 시간을 1000ms로 설정
```

툴팁 시간 제어 예제 14-5

다음 그림과 같이 JLabel을 이용하여 두 개의 이미지를 출력하는 스윙 응용프로그램을 작성하고, JLabel 컴포넌트에 마우스를 올리자마자 툴팁이 나타나게 하고, 툴팁은 10초간 지속되게 하라.




```
1  import javax.swing.*;
2  import java.awt.*;
3
4  public class ToolTipDelayEx extends JFrame {
5     public ToolTipDelayEx() {
6        setTitle("툴팁 지연 시간 제어 예제");
7        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
8        Container c = getContentPane();
9        c.setLayout(new FlowLayout());
10
11       // 체리 이미지 레이블 생성
12       JLabel cherryLabel = new JLabel(new ImageIcon("images/cherry.jpg"));
13       cherryLabel.setToolTipText("체리 이미지 어때요"); // 툴팁 달기
14
15       // 사과 이미지 레이블 생성
16       JLabel appleLabel = new JLabel(new ImageIcon("images/apple.jpg"));
17       appleLabel.setToolTipText("사과 이미지 어때요"); // 툴팁 달기
18
19       c.add(cherryLabel);
20       c.add(appleLabel);
21
22       // ToolTipManager 객체를 얻기
23       ToolTipManager m = ToolTipManager.sharedInstance();
24       m.setInitialDelay(0); // 초기 툴팁 출력 지연 시간을 0초로 설정
25       m.setDismissDelay(10000); // 툴팁 지속 시간을 10초로 설정
26
27       setSize(400,220);
28       setVisible(true);
29    }
```

ToolTipManager 객체를 얻고 툴팁이 초기에 출력되는데 걸리는 시간을 0초로, 툴팁의 지속 시간을 10초로 설정한다.

```
30
31      public static void main(String [] args) {
32          new ToolTipDelayEx();
33      }
34  }
```

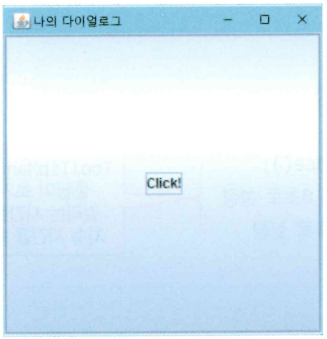
14.4 다이얼로그 만들기

JDialog

다이얼로그
JDialog
최상위 컨테이너

다이얼로그란 보여주고자 하는 내용을 스크린에 출력하고, 사용자로부터 입력을 받는 대화 상자이다. 개발자는 JDialog를 상속받아 자신만의 다이얼로그를 만들 수 있다. JDialog는 JFrame처럼 다른 컨테이너에 속할 필요 없이 화면에 출력 가능한 최상위 컨테이너(Top Level Container)이다.

스윙 다이얼로그를 만드는 방법은 쉽다. [그림 14-5]는 버튼을 하나 가지고 있는 300x300 크기의 다이얼로그를 만든 코드이다. 다이얼로그의 타이틀은 "나의 다이얼로그"로 설정하고, 다이얼로그에 "click!" 버튼을 부착하였다. JDialog의 배치 관리자는 디폴트로 BorderLayout이므로 버튼이 다이얼로그의 중앙(CENTER)에 부착되었다. JDialog 클래스의 주요 멤버는 〈표 14-1〉과 같다.




```
JDialog dialog = new JDialog(); // 다이얼로그 생성
dialog.setTitle("나의 다이얼로그"); // 타이틀 달기
dialog.add(new JButton("click!")); // 다이얼로그에 버튼 삽입
dialog.setSize(300,300); // 다이얼로그 크기 설정
dialog.setVisible(true); // 다이얼로그를 화면에 출력
```

[그림 14-5] new JDialog()에 의해 생성된 다이얼로그 모양

〈표 14-1〉
JDialog 클래스의 주요 멤

메소드	내용
JDialog() JDialog(Frame owner) JDialog(Frame owner, String title) JDialog(Frame owner, String title, boolean modal)	다이얼로그를 생성하는 생성자이다. owner는 다이얼로그의 주인, title은 타이틀, modal은 다이얼로그의 종류를 뜻한다. modal 값이 true이면 모달 다이얼로그를, false이면 모달리스 다이얼로그를 생성한다. 디폴트는 모달리스 타입이다.
void setVisible(boolean b)	b가 true이면 다이얼로그를 화면에 출력하고 false이면 숨긴다.
void setTitle(String title)	title 문자열을 다이얼로그 타이틀로 설정한다.

다이얼로그 만들기

이제 JDialog를 상속받아 다이얼로그를 만드는 방법을 설명해보자. 텍스트필드 창과 OK 버튼을 가진 다이얼로그를 만들고 OK 버튼에는 Action 리스너를 달아보자.

● JDialog를 상속받는 다이얼로그 클래스 작성

JDialog를 상속받은 MyDialog 클래스를 작성한다.

```
class MyDialog extends JDialog {
}
```

● 다이얼로그 생성자 작성

JDialog 클래스의 생성자는 〈표 14-1〉과 같이 여러 가지가 있으며, 다음과 같이 MyDialog의 생성자를 작성한다.

```
class MyDialog extends JDialog {
    public MyDialog(JFrame frame, String title) {
        super(frame, title); // frame은 다이얼로그의 주인이며 title은 다이얼로그의 타이틀이다.
    }
}
```

생성자에서 super()를 호출하여 다이얼로그의 주인(owner)이 되는 컴포넌트와 타이틀 문자열을 JDialog에게 알려주어 등록하게 한다.

● 다이얼로그에 삽입할 컴포넌트 생성

다이얼로그에 삽입할 텍스트필드와 OK 버튼 컴포넌트를 생성한다.

```
class MyDialog extends JDialog {
    private JTextField tf = new JTextField(10); // 10칸 크기의 텍스트필드 컴포넌트 생성
    private JButton okButton = new JButton("OK"); // OK 버튼 생성

    public MyDialog(JFrame frame, String title) {
        super(frame, title);
}
```

주목

● 다이얼로그에 컴포넌트 부착

생성자에 컴포넌트를 부착하는 코드를 작성한다. 아래의 코드는 다이얼로그의 배치관
리자를 FlowLayout으로 설정하고 텍스트필드와 버튼을 삽입한다.

```
class MyDialog extends JDialog {
    private JTextField tf = new JTextField(10);
    private JButton okButton = new JButton("OK");

    public MyDialog(JFrame frame, String title) {
        super(frame, title);
        setLayout(new FlowLayout()); // FlowLayout 배치관리자로 변경
        add(tf); // 텍스트필드 부착
        add(okButton); // OK 버튼 부착
    }
}
```

주목

● OK 버튼에 Action 리스너 달기

이제, OK 버튼에 Action 리스너를 달아, OK 버튼이 선택되면 다이얼로그가 화면에서
사라지게 해보자. 사실, 다이얼로그는 숨겨지는 것이지 사라지는 것은 아니다. 코드는
아래와 같다.

```
1  class MyDialog extends JDialog {
2      private JTextField tf = new JTextField(10);
3      private JButton okButton = new JButton("OK");
4
5      public MyDialog(JFrame frame, String title) {
6          super(frame,title);
7          setLayout(new FlowLayout());
8          add(tf);
9          add(okButton);
10         setSize(200, 100);
11
```

```
12          okButton.addActionListener(new ActionListener() { // Action 리스너 등록
13              public void actionPerformed(ActionEvent e) {
14                  setVisible(false); // OK 버튼이 선택되면 다이얼로그를 화면에서 숨김
15              }
16          });
17      }
18  }
```

주목

MyDialog.this.setVisible(false);와 동일

● MyDialog 다이얼로그를 생성하는 프레임 작성

마지막으로 MyDialog의 다이얼로그를 생성하는 스윙 프레임과 main() 메소드를 작성해보자.

```
1   public class DialogEx extends JFrame {
2       private MyDialog dialog; // 다이얼로그의 레퍼런스
3
4       public DialogEx() {
5           super("DialogEx 예제 프레임");
6
7           dialog = new MyDialog(this, "Test Dialog"); // 다이얼로그 생성
8           JButton btn  = new JButton("Show Dialog"); // 프레임에 버튼 삽입
9
10          // 버튼이 선택되면 다이얼로그를 작동시킨다.
11          btn.addActionListener(new ActionListener() {
12              public void actionPerformed(ActionEvent e) {
13                  dialog.setVisible(true); // 다이얼로그를 출력하고 작동시킨다.
14              }
15          });
16          getContentPane().add(btn);
17          setSize(250, 200);
18          setVisible(true);
19      }
20
21      public static void main(String[] args) {
22          new DialogEx();
23      }
24  }
```

다음과 같이 프레임의 생성자에서 MyDialog의 다이얼로그를 생성한다. 하지만 아직 다이얼로그가 화면에 보이진 않는다.

```
MyDialog dialog = new MyDialog(this, "Test Dialog");
```

그러고 나서 "Show Dialog" 버튼이 선택될 때마다 Action 리스너에서 다이얼로그가 화면에 나타나도록 다음 코드를 실행한다.

```
dialog.setVisible(true);
```

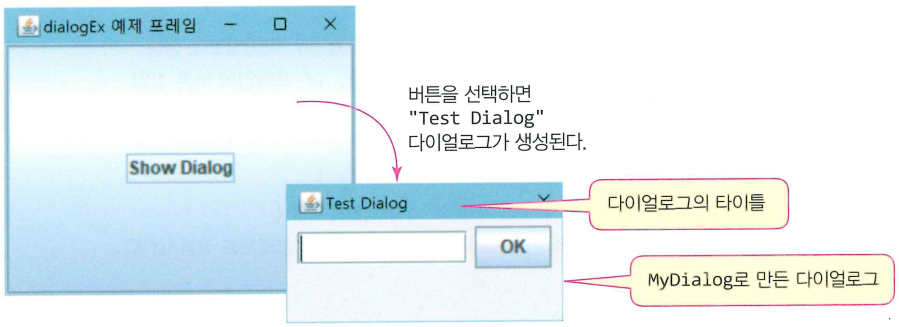
지금까지 설명한 코드로 예제 14-6을 만들어보았다.



예제 14-6 JDialog를 상속받아 다이얼로그 만들기

아래 그림과 같이 Show Dialog 버튼을 가진 프레임을 생성하고, 이 버튼을 선택하면 "Test Dialog" 타이틀을 가진 다이얼로그가 출력되도록 응용프로그램을 작성하라.

다이얼로그의 OK 버튼을 선택하면 다이얼로그가 사라진다.




```
1  import java.awt.*;
2  import java.awt.event.*;
3  import javax.swing.*;
4
5  class MyDialog extends JDialog {
6     private JTextField tf = new JTextField(10); // 다이얼로그에 삽입할 텍스트 필드
7     private JButton okButton = new JButton("OK"); // 다이얼로그에 삽입할 OK 버튼
8
9     public MyDialog(JFrame frame, String title) {
10       super(frame,title);
11       setLayout(new FlowLayout());
12       add(tf);
13       add(okButton);
```

```
14        setSize(200, 100);
15
16        // 다이얼로그의 OK 버튼에 Action 리스너 달기
17        // OK 버튼이 선택되면 다이얼로그가 화면에서 사라지게 한다.
18        okButton.addActionListener(new ActionListener() {
19           public void actionPerformed(ActionEvent e) {
20              setVisible(false); // 다이얼로그를 보이지 않게 한다.
21           }
22        });
23     }
24  }
25
26  public class DialogEx extends JFrame{
27     private MyDialog dialog; // 다이얼로그의 레퍼런스
28
29     public DialogEx() {
30        super("dialogEx 예제 프레임");
31        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
32        JButton btn  = new JButton("Show Dialog");
33
34        // 다이얼로그 생성
35        dialog = new MyDialog(this, "Test Dialog");
36
37        // Show Dialog 버튼을 선택하면 다이얼로그를 작동시킨다.
38        btn.addActionListener(new ActionListener() {
39           public void actionPerformed(ActionEvent e) {
40              dialog.setVisible(true); // 다이얼로그를 출력하고 작동시킨다.
41           }
42        });
43        getContentPane().add(btn);
44        setSize(250,200);
45        setVisible(true);
46     }
47     public static void main(String[] args) {
48        new DialogEx();
49     }
50  }
```

다이얼로그의 OK 버튼이 선택되면 다이얼로그는 화면에서 사라진다. 그러나 다이얼로그 자체가 소멸된 것은 아니다. 40라인에 의해 다시 화면에 출력된다.

"Test Dialog" 다이얼로그 객체를 생성한다. 이것으로 다이얼로그가 화면에 나타나고 작동하는 것이 아니다. 40라인의 setVisible(true)이 호출되어야 한다.

버튼을 선택할 때마다 다이얼로그를 화면에 출력한다. 다이얼로그는 하나이다.

모달 다이얼로그와 모달리스 다이얼로그

모달(modal)
모달리스(modeless)
사용자 입력을 독점

다이얼로그의 타입은 모달(modal)과 모달리스(modeless)의 두 가지가 있다. 모달 타입은 다이얼로그가 일단 출력되면 다이얼로그를 닫기 전에는 다른 작업을 전혀 할 수 없도록 사용자 입력을 독점하는 타입이며, 모달리스 타입은 다른 창과 모달리스 다이얼로그가 각자 독립적으로 작동하는 타입이다. 그러므로 모달리스 다이얼로그를 열어 놓은 채 다른 창에서 입력 작업이 가능하다. 앞의 예제 14-6에서 만든 다이얼로그는 모달리스 타입이다.

다이얼로그를 만들 때는 목적에 맞는 다이얼로그 타입을 결정해야 한다. 예를 들어 파일을 선택하는 파일 다이얼로그는 모달 타입으로 해야 한다. 파일 선택이 끝나기 전에는 파일 읽기 등 다른 작업을 하게 해서는 안 되기 때문이다.

모달/모달리스 타입 선택은 JDialog()의 다음 생성자를 이용한다.

```
JDialog(Frame owner, String title, boolean modal)
```

생성자의 3번째 인자 modal 값이 true이면 모달 다이얼로그가 생성되며, false이면 모달리스 타입이 생성된다.

예제 14-6에서 MyDialog를 모달 타입으로 만들려면 라인 10을 다음과 같이 수정해야 한다. 다이얼로그의 디폴트는 모달리스 타입이다.

```
super(frame, title, true); // true는 다이얼로그를 모달 타입으로 지정
```

이렇게 수정하여 예제 14-6을 다시 실행하면 생성된 MyDialog 다이얼로그를 닫기 전에는 절대로 프레임의 Show Dialog 버튼이 클릭되지 않는다. 모달 다이얼로그가 되든 키나 마우스의 입력에 대한 독점권을 가지기 때문이다.

다이얼로그로부터 사용자 입력 값 전달받기

독자들도 알고 있듯이 다이얼로그의 목적은 사용자로부터 입력을 받는 것이다. 다이얼로그가 사용자로부터 입력받은 내용을 어떻게 프레임에 있는 다른 요소에게 전달할 수 있을까? 이 문제에 대해 자세히 알아보자.

예제 14-6에서 작성한 MyDialog에 getInput() 메소드를 추가하여 다음과 같이 MyModalDialog 클래스를 작성하였다. getInput() 메소드는 텍스트필드 컴포넌트에 사용자가 입력한 문자열을 리턴한다. 입력된 문자열이 없는 경우에는 null을 리턴하도록 하였다.

```
class MyModalDialog extends JDialog {
    private JTextField tf = new JTextField(10); // 다이얼로그에 텍스트필드 삽입
    private JButton okButton = new JButton("OK"); // 다이얼로그에 OK 버튼 삽입

    public MyModalDialog(JFrame frame, String title) {
        super(frame, title, true); // 모달 다이얼로그 타입
        setLayout(new FlowLayout());
        add(tf);
        add(okButton);
        setSize(200, 100); // 다이얼로그의 크기는 200×100

        okButton.addActionListener(new ActionListener() { // OK 버튼의 Action 리스너
            public void actionPerformed(ActionEvent e) {
                                                // OK 버튼이 선택되면 다이얼로그를 닫는다.
                setVisible(false);
            }
        });
    }

    String getInput() { // 사용자가 다이얼로그에 입력한 문자열을 리턴하는 메소드
        if(tf.getText().length() == 0) return null; // 텍스트필드가 빈 경우
        else return tf.getText(); // 텍스트필드에 사용자가 입력한 문자열 리턴
    }
}
```

그리고 DialogEx 프레임의 소스를 수정하여 DialogEx2 클래스를 다음과 같이 작성하였다. 이 코드는 사용자가 다이얼로그의 텍스트필드 창에 입력한 문자열로 프레임에 있는 Show Dialog 버튼의 문자열을 변경한다.

```
public class DialogEx2 extends JFrame {
    private MyModalDialog dialog;

    public DialogEx2() {
        super("DialogEx2 예제 프레임");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        // 모달 다이얼로그 생성
        dialog = new MyModalDialog(this, "Test Modal Dialog");

        JButton btn  = new JButton("Show Modal Dialog");
        btn.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
```

```
14                  // 모달 다이얼로그이므로 setVisible(true) 메소드는
15                  // 다이얼로그가 닫힐 때까지 리턴하지 않는다.
16                  dialog.setVisible(true);
17
18                  String text = dialog.getInput(); // 텍스트필드에 입력한 문자열 얻음
19                  if(text == null) return; // 텍스트필드 창이 비어 있으면 그냥 리턴
20
21                  JButton btn = (JButton)e.getSource();
22                  btn.setText(text); // 버튼의 문자열을 text로 변경
23              }
24          });
25          getContentPane().add(btn);
26          setSize(250,200);
27          setVisible(true);
28      }
29      public static void main(String[] args) {
30          new DialogEx2();
31      }
32  }
```

앞의 소스에서 다이얼로그의 OK 버튼이 클릭되든지 사용자가 다이얼로그를 일방적으로 닫아버리는 경우, setVisible(true)로부터 리턴된다. 그러고 나서 사용자가 입력한 내용을 다이얼로그로부터 읽어 오기 위해 다음 코드를 이용하였다.

```
String text = dialog.getInput();
```

text의 값이 null이면 사용자가 아무 값도 입력하지 않는 것이므로 단순히 리턴하도록 다음 코드를 삽입하였다.

```
if(text == null) return;
```

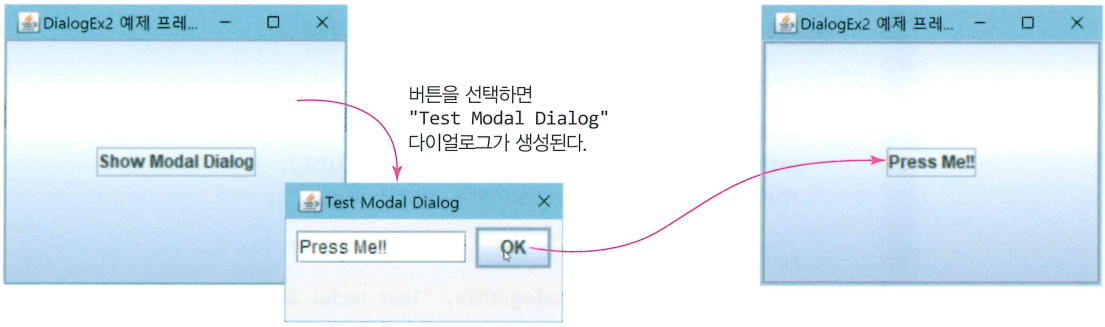
다음 코드는 text 값으로 JButton 컴포넌트의 문자열을 변경한다.

```
btn.setText(text);
```

지금까지 설명한 코드로 완성된 예제 14-7을 만들었다.

모달 다이얼로그로부터 사용자의 입력 값 알아내기 예제 14-7

텍스트필드 창과 OK 버튼을 가진 모달 다이얼로그를 만들고, 사용자가 텍스트필드 창에 문자열을 입력하고 OK 버튼을 클릭하면 입력한 문자열을 Show Modal Dialog 버튼의 문자열을 변경하도록 예제 14-6을 수정하라.



사용자가 "Press Me!!"라고 입력한 후 OK 버튼을 누르면
"Show Modal Dialog" 버튼의 문자열이 변경된다.

```
1  import java.awt.*;
2  import java.awt.event.*;
3  import javax.swing.*;
4
5  class MyModalDialog extends JDialog {
7     private JTextField tf = new JTextField(10);
8     private JButton okButton = new JButton("OK");
9
10    public MyModalDialog(JFrame frame, String title) {
11       super(frame, title, true); // true는 모달 타입을 만들도록 지시
12       setLayout(new FlowLayout());
13       add(tf);
14       add(okButton);
15       setSize(200, 100);
16
17       okButton.addActionListener(new ActionListener() {
18          public void actionPerformed(ActionEvent e) {
19             setVisible(false); // 다이얼로그를 닫는다. 라인 44에서 리턴하게 한다.
20          }
21       });
22    }
23    // 텍스트필드 창에 사용자가 입력한 문자열을 리턴한다.
24    // 입력된 내용이 없으면 null을 리턴한다.
25    public String getInput() {
26       if(tf.getText().length() == 0) return null;
```

```
27        else return tf.getText();
28     }
29  }
30
31  public class DialogEx2 extends JFrame{
32     private MyModalDialog dialog; // 다이얼로그의 레퍼런스
33
34     public DialogEx2() {
35        super("DialogEx2 예제 프레임");
36        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
37        JButton btn  = new JButton("Show Modal Dialog");
38
39        // 모달 다이얼로그 생성
40        dialog = new MyModalDialog(this, "Test Modal Dialog");
41
42        btn.addActionListener(new ActionListener() {
43           public void actionPerformed(ActionEvent e) {
44              dialog.setVisible(true); // 모달 다이얼로그 작동 시작
45
46              // 다이얼로그로부터 사용자가 입력한 문자열을 받아 온다.
47              String text = dialog.getInput();
48
49              if(text == null) return; // 입력한 문자열이 없는 경우
50              JButton btn = (JButton)e.getSource();
51              btn.setText(text); // 입력한 문자열로 이 버튼의 문자열을 변경한다.
52           }
53        });
54        getContentPane().add(btn);
55        setSize(250,200);
56        setVisible(true);
57     }
58     public static void main(String[] args) {
59        new DialogEx2();
60     }
61  }
```

모달 다이얼로그를 생성한다.

모달 다이얼로그는 setVisible() 내에서 다이얼로그가 닫힐 때까지 리턴하지 않는다.

모달 다이얼로그가 닫히면 비로소 이 라인이 실행되며 사용자가 다이얼로그의 텍스 트필드에 입력한 문자열을 얻어 온다.

14.5 팝업 다이얼로그

팝업 다이얼로그와 JOptionPane

팝업 다이얼로그

지금까지 JDialog를 상속받아 다이얼로그를 만드는 방법을 설명하였다. 한편 자바 스윙 패키지에는 개발자가 가져다 사용할 수 있는 유용한 다이얼로그가 많이 있다. 지금부터 설명할 팝업 다이얼로그는 스윙 패키지에 구현된 간단한 팝업 창으로 사용자에게 메시지를 전달하거나 간단한 문자열을 입력받는 유용한 다이얼로그이다.

JOptionPane
static 메소드

JOptionPane 클래스는 여러 종류의 팝업 다이얼로그를 출력하는 static 메소드를 지원한다. JOptionPane 클래스에 의해 지원되는 팝업 다이얼로그는 모두 모달 타입이다. 그러므로 팝업 다이얼로그를 닫기 전에는 프레임을 포함하여 어떤 창으로도 이동할 수 없다. 이제, 많이 사용되는 3가지 팝업 다이얼로그를 활용하는 방법에 대해 알아보자.

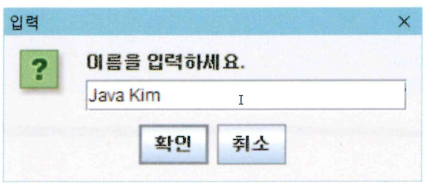
입력 다이얼로그, JOptionPane.showInputDialog()

showInputDialog()
한 줄의 문자열을 입력

JOptionPane의 showInputDialog() 메소드를 호출하면 한 줄의 문자열을 입력받는 입력 다이얼로그를 출력할 수 있다.

```
static String JOptionPane.showInputDialog(String msg)
• msg: 다이얼로그 메시지
• 리턴 값: 사용자가 입력한 문자열. 취소 버튼이 선택되거나 창이 닫히면 null 리턴
```

[그림 14-6]은 JOptionPane.showInputDialog("이름을 입력하세요")를 호출하여 입력 다이얼로그를 출력한 사례이다. 사용자가 입력 창에 "Java Kim"이라고 입력하고 "확인" 버튼을 누르면 showInputDialog()는 "Java Kim"을 리턴하지만, "취소" 버튼을 누르거나 창을 닫으면 null을 리턴한다.




```
String name = JOptionPane.showInputDialog("이름을 입력하세요.");
// name에 "Java Kim"이 리턴
// 사용자가 입력 없이 팝업 다이얼로그를 닫으면 null 리턴
```

[그림 14-6] JOptionPane.showInputDialog()로 만든 입력 다이얼로그

확인 다이얼로그, JOptionPane.showConfirmDialog()

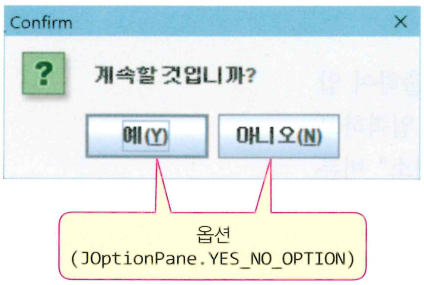
showConfirmDialog()

사용자로부터 확인/취소를 입력받기 위한 팝업 다이얼로그를 출력하는 showConfirmDialog() 메소드는 다음과 같다.

```
static int JOptionPane.showConfirmDialog(Component parentComponent, Object msg,
                                         String title, int optionType)
```
- parentComponent: 다이얼로그의 부모 컴포넌트로서 다이얼로그가 출력되는 영역의 범위 지정을 위해 사용(예: 프레임). null이면 전체 화면 중앙에 출력
- msg: 다이얼로그 메시지
- title: 다이얼로그 타이틀
- optionType: 다이얼로그 옵션 종류 지정
  YES_NO_OPTION, YES_NO_CANCEL_OPTION, OK_CANCEL_OPTION
- 리턴 값: 사용자가 선택한 옵션 종류
  YES_OPTION, NO_OPTION, CANCEL_OPTION, OK_OPTION, CLOSED_OPTION

YES_NO_OPTION
YES_OPTION
NO_OPTION
CLOSED_OPTION

[그림 14-7]은 YES_NO_OPTION 타입의 확인 다이얼로그를 출력한 사례이다. 이 확인 다이얼로그는 "예", "아니오"의 두 개의 버튼만 가진다. showConfirmDialog()는 사용자가 "예" 버튼을 선택하면 YES_OPTION을, "아니오" 버튼을 선택하면 NO_OPTION을 리턴한다. 그러나 사용자가 어떤 버튼도 선택하지 않고 다이얼로그를 종료한 경우 CLOSED_OPTION 값을 리턴한다.




```
int result = JOptionPane.showConfirmDialog(null, "계속할 것입니까?",
                                           "Confirm", JOptionPane.YES_NO_OPTION);
if(result == JOptionPane.CLOSED_OPTION) {
    // 사용자가 "예"나 "아니오"의 선택 없이 다이얼로그 창을 닫은 경우
}
else if(result == JOptionPane.YES_OPTION) {
    // 사용자가 "예"를 선택한 경우
}
else {
    // 사용자가 "아니오"를 선택한 경우
}
```

[그림 14-7] JOptionPane.showConfirmDialog()로 만든 확인 다이얼로그

메시지 다이얼로그, JOptionPane.showMessageDialog()

showMessageDialog() 메소드를 이용하면, 사용자에게 문자열 메시지를 출력하기 위한 메시지 다이얼로그를 출력한다. 이 메소드는 다음과 같다.

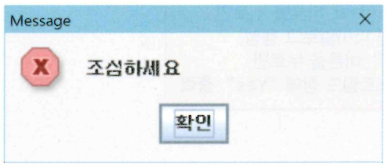
showMessageDialog()

```
static void JOptionPane.showMessageDialog(Component parentComponent,
                    Object msg, String title, int messageType)
```

- parentComponent: 다이얼로그의 부모 컴포넌트로서 다이얼로그가 출력되는 영역의 범위 지정을 위해 사용(예: 프레임). null이면 전체 화면 중앙에 출력
- msg: 다이얼로그 메시지
- title: 다이얼로그 타이틀
- messageType: 다이얼로그의 종류로서 다음 중 하나
  ERROR_MESSAGE, INFORMATION_MESSAGE, WARNING_MESSAGE, QUESTION_MESSAGE, PLAIN_MESSAGE

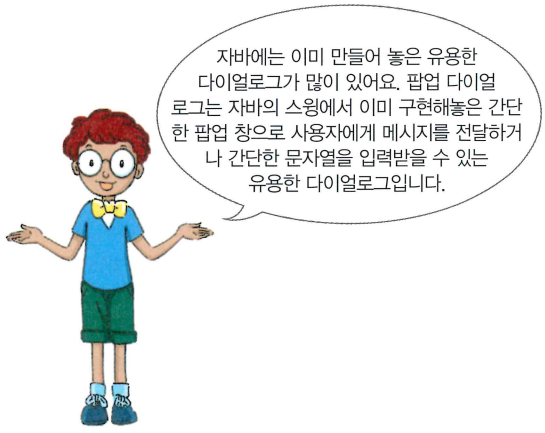
[그림 14-8]은 ERROR_MESSAGE 타입의 메시지 다이얼로그를 출력한 사례이다. 이 메시지 다이얼로그는 메시지와 "확인" 버튼만 출력하며, 사용자가 "확인" 버튼을 누르거나 다이얼로그를 닫으면 showMessageDialog()는 단순 리턴한다.

ERROR_MESSAGE




```
JOptionPane.showMessageDialog(null, "조심하세요", "Message",
                                    JOptionPane.ERROR_MESSAGE);
```

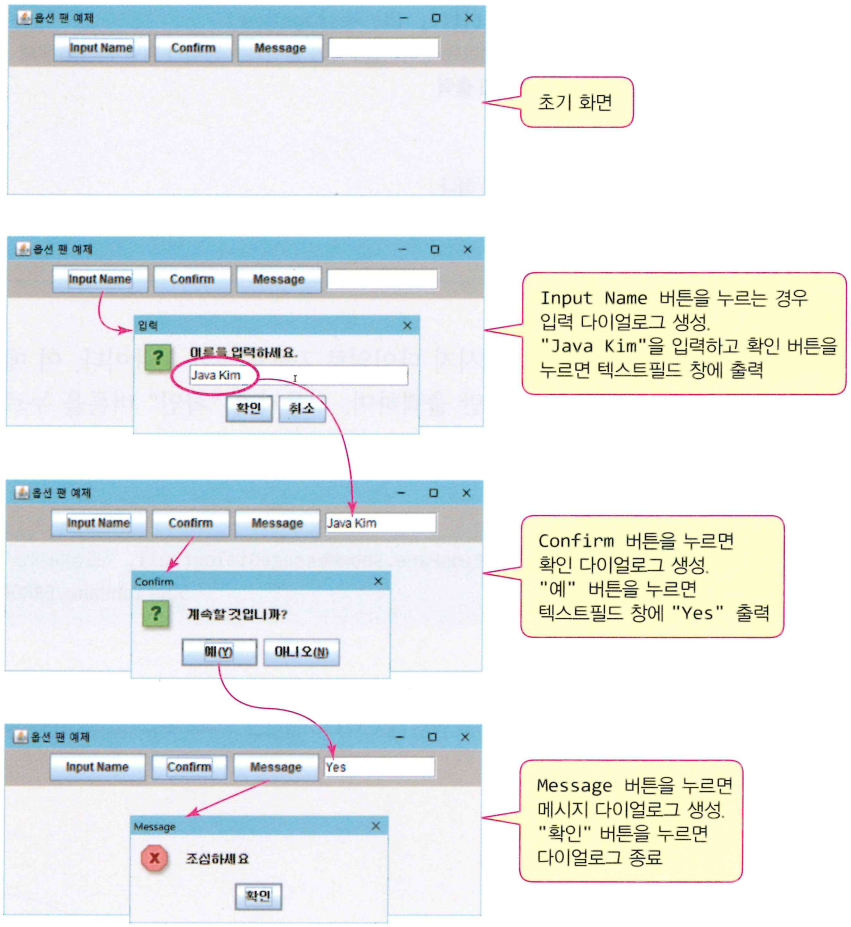
[그림 14-8] JOptionPane.showMessageDialog()로 만든 메시지 다이얼로그



예제 14-8 JOptionPane을 사용한 팝업 다이얼로그 작성

다음 그림과 같이 3개의 팝업 다이얼로그를 출력하는 응용프로그램을 작성해보라.

각 버튼을 클릭하면 해당하는 팝업 다이얼로그를 출력하며, 이때 각 팝업에 대해 사용자가 입력한 내용이나 선택한 버튼 정보를 텍스트필드 창에 출력한다.




```
1  import javax.swing.*;
2  import java.awt.event.*;
3  import java.awt.*;
4
5  public class OptionPaneEx extends JFrame {
6     public OptionPaneEx() {
7        setTitle("옵션 팬 예제");
8        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
9        Container c = getContentPane();
```

```
10        setSize(500,200);
11        c.add(new MyPanel(), BorderLayout.NORTH);
12        setVisible(true);
13     }
14
15     class MyPanel extends Panel {
16        private JButton inputBtn = new JButton("Input Name");
17        private JTextField tf = new JTextField(10);
18        private JButton confirmBtn = new JButton("Confirm");
19        private JButton messageBtn = new JButton("Message");
20
21        public MyPanel() {
22           setBackground(Color.LIGHT_GRAY);
23           add(inputBtn);
24           add(confirmBtn);
25           add(messageBtn);
26           add(tf);
27
28           inputBtn.addActionListener(new ActionListener() {
29              public void actionPerformed(ActionEvent e) {
30                 // 입력 다이얼로그 생성
31                 String name = JOptionPane.showInputDialog("이름을 입력하세요.");
32                 if(name != null)
33                    tf.setText(name); // 사용자가 입력한 문자열을 텍스트필드 창에 출력
34              }
35           });
36
37           confirmBtn.addActionListener(new ActionListener() {
38              public void actionPerformed(ActionEvent e) {
39                 // 확인 다이얼로그 생성
40                 int result = JOptionPane.showConfirmDialog(null,
41                    "계속할 것입니까?", "Confirm", JOptionPane.YES_NO_OPTION);
42
43                 // 사용자가 선택한 버튼에 따라 문자열을 텍스트필드 창에 출력
44                 if(result == JOptionPane.CLOSED_OPTION)
45                    tf.setText("Just Closed without Selection");
46                 else if(result == JOptionPane.YES_OPTION)
47                    tf.setText("Yes");
48                 else
49                    tf.setText("No");
50              }
51           });
52
```

Input Name 버튼을 선택하면 입력 다이얼로그를 생성한다.

Confirm 버튼을 선택하면 확인 다이얼로그를 생성한다.

Message 버튼을 선택하면 메시지 다이얼로그를 생성한다.

```
53            messageBtn.addActionListener(new ActionListener() {
54               public void actionPerformed(ActionEvent e) {
55                  // 메시지 다이얼로그 생성
56                  JOptionPane.showMessageDialog(null, "조심하세요", "Message",
                                                JOptionPane.ERROR_MESSAGE);
57               }
58            });
59         }
60      }
61
62      public static void main(String [] args) {
63         new OptionPaneEx();
64      }
65   }
```

14.6 파일 다이얼로그

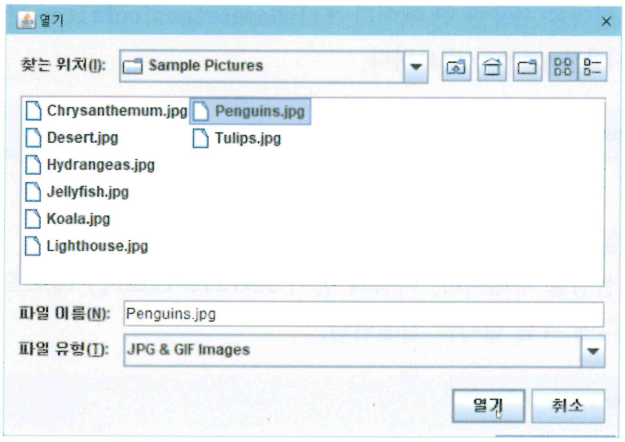
파일 다이얼로그
파일 이름 입력

한글이나 워드를 사용하는 중에 파일 읽기나 쓰기를 하려고 메뉴를 선택하면, [그림 14-9]와 같이 파일의 이름을 선택하는 다이얼로그가 출력된다. 이 다이얼로그를 파일 다이얼로그라고 부른다. 스윙 패키지는 응용프로그램에서 쉽게 파일 다이얼로그를 출력할 수 있도록 지원한다. 파일 다이얼로그의 목적이 사용자로부터 파일 이름을 입력받고자 하는 것임을 기억하기 바란다.

JFileChooser

JFileChooser
파일 열기 다이얼로그
파일 저장 다이얼로그

JFileChooser는 파일 탐색기(File Browser)와 같은 기능을 하는 파일 다이얼로그를 구현한 스윙 컴포넌트이다. JFileChooser는 [그림 14-9]와 같은 모양의 다이얼로그를 출력하여 사용자가 파일이나 디렉터리를 선택하게 한다. JFileChooser를 이용하면 파일 열기 다이얼로그(File Open Dialog)와 파일 저장 다이얼로그(File Save Dialog)를 모두 출력할 수 있다. 파일 열기 다이얼로그는 열고자 하는 파일의 이름을 사용자로부터 입력받기 위한 것이며, 파일 저장 다이얼로그는 저장하려는 파일 이름을 입력받기 위한 것이다.



[그림 14-9] JFileChooser를 이용하여 출력한 파일 열기 다이얼로그

파일 열기 다이얼로그 생성

우선, 파일 열기 다이얼로그부터 만들어보자. 파일 열기 다이얼로그를 생성하고 다루는 과정은 7단계로서 다음과 같다.

1. JFileChooser 객체 생성

```
JFileChooser chooser = new JFileChooser();
```

2. 파일 필터 생성

파일 필터(file filter)란 파일 다이얼로그가 특정 확장자를 가진 파일만 보여주기 위해 사용되는 필터 정보로, FileNameExtensionFilter 클래스를 이용하여 다음과 같이 생성한다.

파일 필터
FileNameExtensionFilt

```
FileNameExtensionFilter filter = new FileNameExtensionFilter(
                "JPG & GIF Images", "jpg", "gif");
```

"JPG & GIF Images" : 파일 종류 창에 출력되는 문자열
"jpg", "gif" : 필터링하는 파일 확장자들

이 코드에서 첫 번째 인자인 "JPG & GIF Images"는 파일 다이얼로그의 "파일 종류" 창에 출력되는 문자열 정보이며, 두 번째 인자부터는 필터링하고자 하는 확장자를 나열한다. 확장자는 무한정 나열할 수 있다. 앞의 코드에서는 "jpg"와 "gif"의 2개 확장자만 나열하였으므로, 다이얼로그는 확장자가 jpg이거나 gif인 이미지 파일만 나

열한다. [그림 14-10]은 파일 필터를 사용한 한 예이다. FileNameExtensionFilter 클래스를 사용하기 위해서는 다음 import가 필요하다.

```
import javax.swing.filechooser.*;
```

3. JFileChooser에 파일 필터 설정

setFileFilter()

파일 필터를 다이얼로그에 설정할 차례이다. 다음과 같이 setFileFilter() 메소드를 이용하여 파일 다이얼로그에 파일 필터를 설정한다.

```
chooser.setFileFilter(filter);
```

addChoosableFileFilter()

JFileChooser는 여러 개의 파일 필터를 가질 수 있다. JFileChooser에 새로운 파일 필터를 추가하려면 다음과 같이 addChoosableFileFilter() 메소드를 이용하면 된다.

```
chooser.addChoosableFileFilter(filter);
```

4. 파일 대신 디렉터리를 선택하고자 할 때

setFileSelectionMode()

setFileSelectionMode() 메소드를 이용하면 파일 다이얼로그에서 파일이나, 디렉터리, 혹은 둘 다 선택 가능하도록 설정할 수 있다. 다음 코드는 파일이나 디렉터리 모두 선택 가능하게 한다.

```
chooser.setFileSelectionMode(JFileChooser.FILES_AND_DIRECTORIES);
```

이 코드가 없으면 파일만 선택 가능하다.

5. 파일 열기 다이얼로그 출력

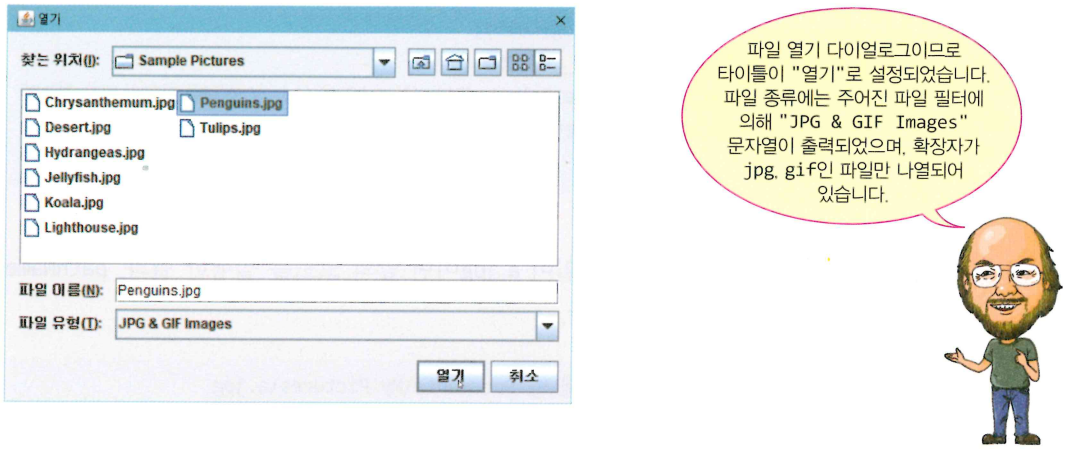
showOpenDialog()

이제, 파일 다이얼로그를 출력하고 가동시켜보자. 다음 코드와 같이 showOpenDialog()를 이용하여 파일 다이얼로그를 화면에 출력한다. 결과는 [그림 14-10]과 같다.

```
int ret = chooser.showOpenDialog(null);
```

showOpenDialog(Component parent) 메소드의 인자인 parent는 다이얼로그의 부모

컴포넌트를 지정하는 것으로, 다이얼로그를 출력할 위치를 정할 때, 기준이 되는 부모 컴포넌트이며, null을 주면 전체 화면을 기준으로 위치를 잡게 된다.



[그림 14-10] jpg와 gif 확장자를 가진 파일만 나열하는 파일 열기 다이얼로그 출력

6. 사용자가 선택한 파일 이름 알아내기

getSelectedFile()

사용자는 보통 [그림 14-10]에서 파일을 선택하고 "열기" 버튼을 누른다. 이때 파일 다이얼로그는 보이지 않게 될 뿐이지 JFileChooser 객체가 사라진 것은 아니다. JFileChooser 객체 내부에는 사용자가 선택한 파일 이름, 파일 경로명, 디렉터리명 등 다양한 정보가 남아 있다. 사용자가 선택한 파일 이름은 다음과 같이 JFileChooser의 getSelectedFile() 메소드(File 객체 리턴)를 이용하여 알아낸다.

```
String pathName = chooser.getSelectedFile().getPath(); // 완전경로명
```

7. showOpenDialog()의 리턴 값 처리

마지막으로 사용자가 취소 버튼을 선택하거나 강제로 다이얼로그를 닫는 경우에 대한 대책이 필요하다. showOpenDialog()는 사용자의 행위에 따라 3가지의 값 중 하나를 리턴한다.

- JFileChooser.APPROVE_OPTION: "열기" 버튼을 누른 경우
- JFileChooser.CANCEL_OPTION: "취소" 버튼을 누른 경우
- JFileChooser.ERROR_OPTION: 오류가 발생하거나 사용자가 다이얼로그를 닫은 경우

응용프로그램에서 사용자가 선택한 파일 이름을 정확하게 얻기 위해서는 정상적으로 "열기" 버튼을 누른 경우를 가려낼 필요가 있다. JFileChooser로부터 사용자가 선택한 파일의 경로명과 이름을 알아내는 코드는 다음과 같다.

```
int ret = chooser.showOpenDialog(null);
if(ret == JFileChooser.APPROVE_OPTION) {
    String pathName = chooser.getSelectedFile().getPath(); // 선택한 파일의 완전 경로명
    String fileName = chooser.getSelectedFile().getName(); // 파일의 이름
}
```

예를 들어, 사용자가 선택한 파일이 a.jpg이면 앞의 코드를 실행한 결과, pathName은 다음과 같이 완전 경로명이다.

```
C:\Documents and Settings\황기태\My Documents\My Pictures\a.jpg
```

그리고 fileName은 순수 파일 이름으로 다음과 같다.

```
a.jpg
```

파일 저장 다이얼로그 생성

showSaveDialog()

파일 저장 다이얼로그를 출력하기 위해서는 showOpenDialog() 대신 다음과 같이 showSaveDialog()를 호출하면 된다. 나머지 과정은 동일하다.

```
int ret = chooser.showSaveDialog(null);
```



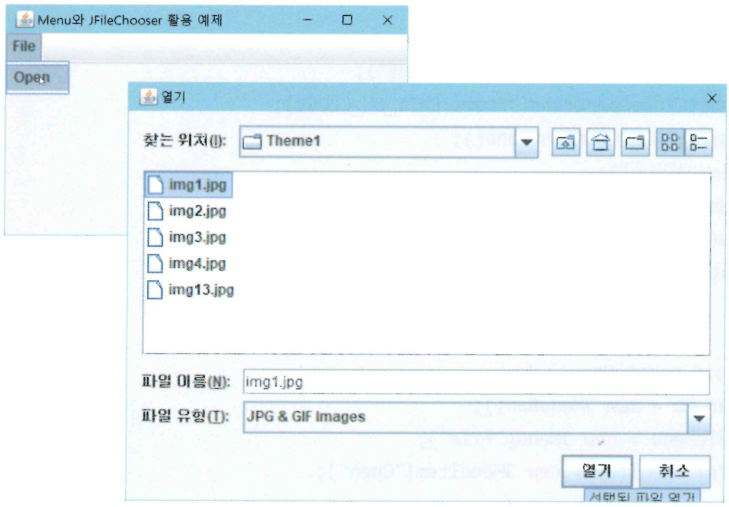
잠깐!

- 파일 다이얼로그가 실제 파일 열기까지 수행하는 것은 아니다. 다이얼로그는 사용자로부터 파일 이름을 얻어내는 목적에 국한된다. 응용프로그램은 파일 다이얼로그로부터 사용자가 선택한 파일의 이름을 알아 온 후 파일을 여는 작업을 따로 해야 한다.
- 파일 다이얼로그가 닫혔다고 해서 JFileChooser 객체가 사라졌거나 사용 불가능하게 된 것은 아니다. showOpenDialog()나 showSaveDialog()를 다시 호출하면 다이얼로그가 다시 화면에 출력된다. 재사용이 가능하다.

파일 열기 다이얼로그 생성 및 출력 예제 14-9

JFileChooser를 이용하여 파일 열기 다이얼로그를 출력하고 사용자가 선택한 이미지 파일을 화면에 그리는 응용프로그램을 작성하라.

다음 그림과 같이 메뉴를 만들고, Open 메뉴아이템이 선택되면 파일 열기 다이얼로그를 출력하라. 사용자로부터 이미지 파일의 이름을 입력받아 JLabel 컴포넌트를 이용하여 이미지를 출력한다. jpg나 gif 형식의 파일을 필터링하도록 파일 필터를 만들어서 사용하라.





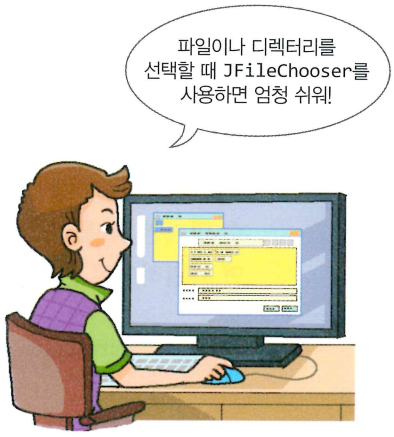
```java
import javax.swing.*;
import javax.swing.filechooser.*;
import java.awt.event.*;
import java.awt.*;

public class MenuAndFileDialogEx extends JFrame {
   private JLabel imageLabel = new JLabel();

   public MenuAndFileDialogEx() {
      setTitle("Menu와 JFileChooser 활용 예제");
      setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      Container c = getContentPane();
      c.add(imageLabel);
      createMenu();
      setSize(350,200);
      setVisible(true);
   }

   private void createMenu() {
      JMenuBar mb = new JMenuBar();
      JMenu fileMenu = new JMenu("File");
      JMenuItem openItem = new JMenuItem("Open");

      // Open 메뉴아이템에 Action 리스너를 등록한다.
      openItem.addActionListener(new OpenActionListener());
      fileMenu.add(openItem);
      mb.add(fileMenu);
      setJMenuBar(mb);
   }

   // Open 메뉴아이템이 선택되면 호출되는 Action 리스너
   class OpenActionListener implements ActionListener {
      private JFileChooser chooser;

      public OpenActionListener() {
         chooser = new JFileChooser(); // 파일 다이얼로그 생성
      }
      public void actionPerformed(ActionEvent e) {
         FileNameExtensionFilter filter = new FileNameExtensionFilter(
               "JPG & GIF Images", // 파일 이름난에 출력될 문자열
               "jpg", "gif"); // 파일 필터로 사용되는 확장자. *.jpg. *.gif만 나열됨

         chooser.setFileFilter(filter); // 파일 다이얼로그에 파일 필터 설정
```

```
43
44          int ret = chooser.showOpenDialog(null); // 파일 다이얼로그 출력
45          if(ret != JFileChooser.APPROVE_OPTION) { // 사용자가 창을 강제로 닫았거나
                                                     취소 버튼을 누른 경우
46             JOptionPane.showMessageDialog(null, "파일을 선택하지 않았습니다",
47                                           "경고", JOptionPane.WARNING_MESSAGE);
48             return;
49          }
50
51          // 사용자가 파일을 선택하고 "열기" 버튼을 누른 경우
52          String filePath = chooser.getSelectedFile().getPath(); // 파일 경로명 리턴
53          imageLabel.setIcon(new ImageIcon(filePath)); // 이미지 출력
54          pack(); // 이미지의 크기에 맞추어 프레임 크기 조절
55       }
56    }
57    public static void main(String [] args) {
58       new MenuAndFileDialogEx();
59    }
60 }
```

showOpenDialog()에 의
파일 다이얼로그를 출력한다

showOpenDialog()의 리턴
값이 APPROVE_OPTION이 아
닌 경우는 비정상적으로 다이
로그가 종료된 경우이다.

사용자가 선택한 이미지 파일
을 읽어 화면에 그린다.



14.7 컬러 다이얼로그

JColorChooser

JColorChooser
컬러 다이얼로그

JColorChooser를 이용하면 사용자가 색을 선택할 수 있는 컬러 다이얼로그를 출력할 수 있다. [그림 14-11]은 JColorChooser를 사용하여 컬러 다이얼로그를 출력한 예이다.



[그림 14-11] JColorChooser를 이용하여 생성된 컬러 다이얼로그

컬러 다이얼로그 생성 및 출력

JColorChooser를 이용하여 컬러 다이얼로그를 출력하고 색을 입력받는 과정을 살펴보자.

1. 컬러 다이얼로그 출력

showDialog()

JColorChooser 객체를 생성하여 컨텐트팬이나 패널에 컴포넌트로 삽입하여 사용할 수 있지만, 다음과 같이 JColorChooser의 static 메소드인 showDialog()를 호출하면 독립적으로 출력하여 동작하는 컬러 다이얼로그를 출력할 수 있다.

```
Color selectedColor = JColorChooser.showDialog(null, "Color", Color.YELLOW);
```

사용자가 선택한 색 리턴

showDialog() 메소드는 사용자가 선택한 색을 리턴한다. 그러므로 앞의 코드에서 selectedColor는 사용자가 선택한 색상이다. 이 코드는 다이얼로그의 타이틀을 "Color"로 하고, 다이얼로그가 보여줄 초기 색상을 YELLOW로 하였다. 그러므로 [그림 14-11]의 미리보기 부분에 색이 노란색(YELLOW)으로 표시되어 있다.

2. 사용자가 선택한 색 얻기

showDialog()는 [그림 14-11]에서 "확인" 버튼을 누르면 선택한 색을 리턴하며, "취소" 버튼이나 다이얼로그를 강제로 닫는 경우 null을 리턴한다. 그러므로 다음과 같이 판별하는 코드가 필요하다.

```
if(selectedColor != null) { // 사용자가 색을 선택하고 "확인" 버튼을 누른 경우
    // 사용자가 선택한 색 selectedColor 사용
}
```

JColorChooser를 이용한 컬러 다이얼로그 사용 예제 14-10

컬러 다이얼로그를 이용하여 글자 색을 선택하는 프로그램을 작성하라.

다음 그림과 같이 Color 메뉴아이템이 선택되면 컬러 다이얼로그를 출력하고 사용자가 선택한 색으로 "Hello"의 글자색을 변경한다.



```java
import javax.swing.*;
import java.awt.event.*;
import java.awt.*;

public class MenuAndColorChooserEx extends JFrame {
   private JLabel label = new JLabel("Hello");

   public MenuAndColorChooserEx() {
      setTitle("JColorChooser 예제");
      setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      Container c = getContentPane();
      label.setHorizontalAlignment(SwingConstants.CENTER);
      label.setFont(new Font("Ravie", Font.ITALIC, 30));
      c.add(label, BorderLayout.CENTER);
      createMenu();
      setSize(250,200);
      setVisible(true);
   }

   private void createMenu() {
      JMenuBar mb = new JMenuBar();
      JMenuItem colorMenuItem = new JMenuItem("Color");
      JMenu fileMenu = new JMenu("Text");

      // Color 메뉴아이템에 Action 리스너 등록
      colorMenuItem.addActionListener(new MenuActionListener());

      fileMenu.add(colorMenuItem);
      mb.add(fileMenu);
      this.setJMenuBar(mb);
   }

   // Color 메뉴아이템이 선택될 때 실행되는 Action 리스너
   class MenuActionListener implements ActionListener {
      public void actionPerformed(ActionEvent e) {
         String cmd = e.getActionCommand(); // 메뉴아이템의 이름 리턴
         if(cmd.equals("Color")) { // Color 메뉴아이템의 경우
            // 컬러 다이얼로그를 출력하고 사용자가 선택한 색을 알아온다.
            Color selectedColor = JColorChooser.showDialog(null,"Color",Color.YELLOW);

            // 취소 버튼을 누르거나 그냥 다이얼로그를 닫는 경우 selectedColor는 null이다.
            if(selectedColor != null)
               label.setForeground(selectedColor);
```

Color 메뉴아이템에 Action 리스너를 등록한다.

메뉴아이템이 선택되면 컬러 다이얼로그를 출력한다. 그리고 사용자로부터 받은 색으로 문자열의 색을 변경한다.

컬러 다이얼로그를 출력한다. selectedColor는 사용자가 선택한 색이다.

사용자가 선택한 색으로 문자열의 색을 변경한다.

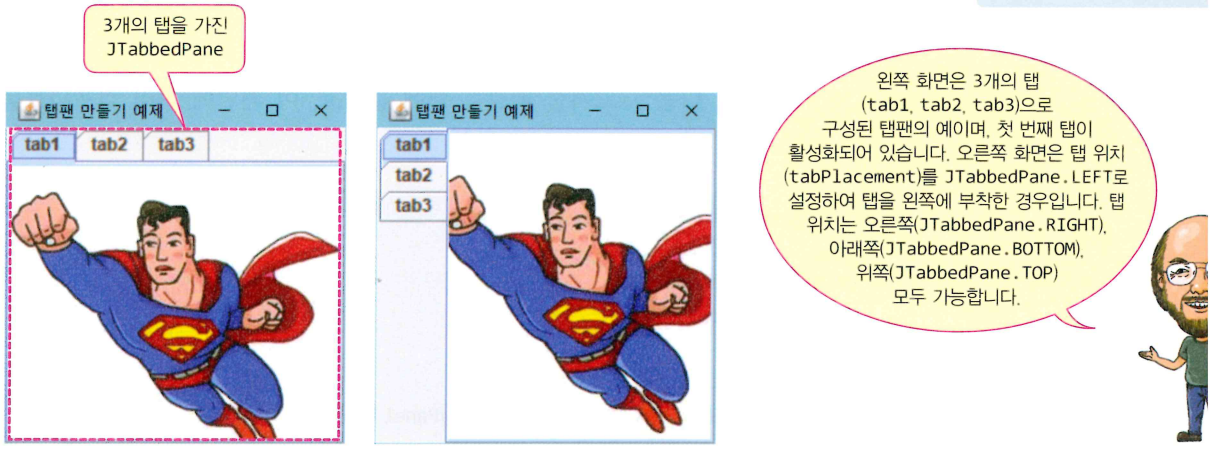
```
43            }
44         }
45      }
46
47      public static void main(String [] args) {
48         new MenuAndColorChooserEx();
49      }
50   }
```

14.8 탭팬

JTabbedPane

JTabbedPane은 [그림 14-12]와 같이 여러 개의 패널을 겹치게 하여 출력 공간을 공유하는 탭팬을 구현한다. 탭팬에 부착된 각 패널을 탭(tab)이라 부른다.

JTabbedPane
탭팬
탭



[그림 14-12] JTabbedPane을 이용하여 작성된 탭팬

탭팬 만들기

탭팬을 만드는 과정은 단순하다.

1. JTabbedPane 객체 생성

```
JTabbedPane pane = new JTabbedPane(); // 탭 위치는 디폴트로 JTabbedPane.TOP
```

탭은 디폴트로 탭팬의 위쪽에 위치한다. 탭 위치를 왼쪽에 부착하려면 다음과 같이 한다.

```
JTabbedPane pane = new JTabbedPane(JTabbedPane.LEFT); // 탭 위치를 왼쪽으로
```

2. 탭 만들어 붙이기

addTab()

탭팬에 탭을 붙이는 작업은 JTabbedPane의 addTab() 메소드를 이용한다. addTab()의 첫 번째 인자는 탭의 타이틀이며, 두 번째 인자는 탭에 출력될 컴포넌트이다. 다음은 타이틀이 "tab1"이고 img1.jpg 이미지를 가진 레이블을 컴포넌트로 가진 탭을 만들어 탭팬에 부착하는 코드이다.

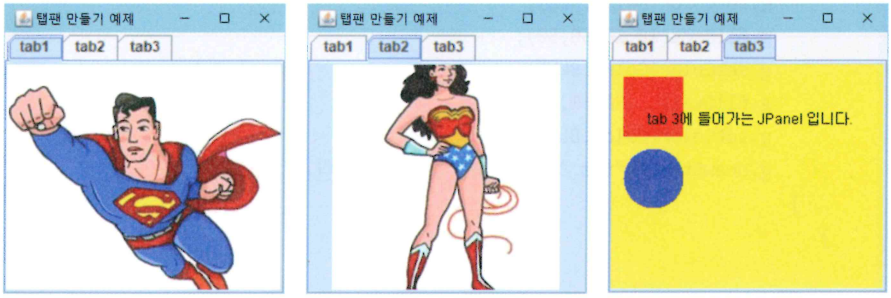
```
// img1.jpg 이미지 레이블을 가진 tab1
pane.addTab("tab1", new JLabel(new ImageIcon("images/img1.jpg")));
```

탭에는 기본적으로 컴포넌트를 하나만 붙일 수 있다. 여러 개의 컴포넌트를 하나의 탭을 담고자 하면, 다음 코드처럼 패널을 탭에 붙이고, 이 패널에 여러 컴포넌트를 담으면 된다.

```
class MyPanel extends JPanel {
    ......................
    // 여러 컴포넌트를 붙인다.
    ......................
}
.......................
pane.addTab("tab3", new MyPanel()); // MyPanel 객체를 컴포넌트로 가진 tab3
```

탭팬 만들기 예제 14-11

다음 그림과 같이 3개의 탭을 가진 탭팬을 작성하라.




```
1  import javax.swing.*;
2  import java.awt.*;
3
4  public class TabbedPaneEx extends JFrame {
5     public TabbedPaneEx() {
6        setTitle("탭팬 만들기 예제");
7        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
8        Container c = getContentPane();
9        JTabbedPane pane = createTabbedPane(); // 탭팬을 생성한다.
10       c.add(pane, BorderLayout.CENTER); // 탭팬을 컨텐트팬에 부착한다.
11       setSize(250,250);
12       setVisible(true);
13    }
14
15    // 탭팬을 생성하고 3개의 탭을 생성하여 부착한다.
16    private JTabbedPane createTabbedPane() {
17       JTabbedPane pane = new JTabbedPane(); // 탭팬 객체를 생성한다.
18       pane.addTab("tab1", new JLabel(new ImageIcon("images/img1.jpg")));
                                                          // 첫 번째 탭
19       pane.addTab("tab2", new JLabel(new ImageIcon("images/img2.jpg")));
                                                          // 두 번째 탭
20       pane.addTab("tab3", new MyPanel()); // 세 번째 탭
21       return pane;
22    }
23
24    // 3번째 탭의 컴포넌트로 사용될 패널
25    class MyPanel extends JPanel {
26       public MyPanel() {
27          this.setBackground(Color.YELLOW);
28       }
```

```
29        public void paintComponent(Graphics g) {
30           super.paintComponent(g);
31           g.setColor(Color.RED);
32           g.fillRect(10,10,50,50);
33           g.setColor(Color.BLUE);
34           g.fillOval(10,70,50,50);
35           g.setColor(Color.BLACK);
36           g.drawString("tab 3에 들어가는 JPanel 입니다. ", 30, 50);
37        }
38     }
39
40     public static void main(String [] args) {
41        new TabbedPaneEx();
42     }
43  }
```

14.9 자바 오디오 다루기

자바의 오디오 API

오디오 API

자바는 응용프로그램에서 오디오를 재생하고 제어할 수 있는 오디오 API를 제공하다
다음 2가지 종류의 오디오 데이터를 다룰 수 있다.

- 디지털 오디오(Digital Audio)
- 미디(MIDI: Music Instrument Digital Interface) 데이터

디지털 오디오

디지털 오디오란 연주되고 있는 음악이나 사람의 목소리 등 아날로그 소리를 샘플
링하여 디지털 데이터로 만든 오디오 데이터로, 음악 CD나 전화 목소리를 녹음한 WAV
파일 등이 이에 해당한다. 디지털 오디오를 다루는 자바 클래스와 인터페이스들은
javax.sound.sampled 패키지에 제공된다.

미디 데이터

한편, 미디 데이터란 피아노, 바이올린, 드럼 등 악기의 소리를 낼 수 있는 특별한
장치(MIDI Device)에게, 어떤 악기를, 어느 높이로, 얼마의 시간동안 연주할지를 지시

하는 데이터이다. 미디 제어를 위한 클래스와 인터페이스는 javax.sound.midi 패키지에서 제공된다.

자바의 사운드 API는 양이 방대하고 내용도 어려우므로 이 책에서는 디지털 오디오에 관해서만 알아본다.

디지털 오디오 포맷

자바 오디오 API로 재생 가능한 디지털 오디오 포맷은 다음과 같다.

```
WAV, AU, AIFF, AIFC
```

오디오 클립과 오디오 재생

지금부터 오디오를 재생하고 중단하는 코드를 작성해보자.

1. 오디오 클립 만들기

오디오 클립
Clip 인터페이스

오디오를 재생하기 위해서는 먼저 오디오 클립(audio clip)을 만들어야 한다. 오디오 클립의 기능은 자바의 Clip 인터페이스 객체를 통해 구현된다. 먼저 비어 있는 오디오 클립은 다음과 같이 생성한다.

```
Clip clip = AudioSystem.getClip(); // AudioSystem 클래스의 static 메소드 getClip()
```

오디오 클립은 오디오 파일을 재생하기 전에 오디오 데이터를 미리 로딩해두고 제어하는 객체로서, 오디오 클립을 이용하면 오디오 파일 내 어느 위치든 재생가능하고 오디오의 일부 혹은 전체를 반복 재생할 수도 있다.

2. 오디오 클립에 오디오 스트림 연결

스트림 객체 생성

오디오 파일로부터 오디오 데이터를 읽을 오디오 스트림 객체를 생성한다. 다음은 애국가.wav 파일을 읽어들일 오디오 스트림 객체를 생성하는 코드이다.

```
File audioFile = new File("애국가.wav");
AudioInputStream audioStream = AudioSystem.getAudioInputStream(audioFile);
```

그리고 나면, 다음과 같이 오디오 클립과 오디오 스트림을 연결한다.

```
clip.open(audioStream); // 오디오 클립과 오디오 스트림 연결
```

이제, clip은 오디오 스트림으로부터 오디오 데이터를 받아 재생할 수 있는 상태가 된다.

3. 오디오 재생

start()

Clip 클래스의 start() 메소드를 호출하면 오디오 재생을 시작한다.

```
clip.start(); // 오디오 재생 시작
```

오디오 재생, 중단을 제어하는 Clip 메소드는 다음과 같다.

void open(AudioInputStream stream)
오디오 클립은 stream에 존재하는 오디오의 형식과 데이터를 인식하고, 재생할 수 있는 준비를 갖춘다.
void start()
현재 프레임 위치에서 오디오 클립을 재생한다. 처음 실행될 때 프레임 위치는 0
void stop()
재생중인 오디오 클립의 재생을 중단한다.
void setFramePosition(int frames)
오디오 샘플 내에 재생할 프레임의 위치를 지정한다. 프레임의 시작 번호는 0이다.
void loop(int count)
현재 프레임의 위치에서 시작하여 count만큼 반복 재생한다. count 값이 LOOP_CONTINUOUSLY이면 무한 반복한다.
void close()
오디오 클립의 모든 자원을 반환한다.



잠깐!

오디오 클립은 네트워크 등으로부터 실시간 재생하는 스트리밍에는 사용할 수 없고, 오디오 파일과 같이, 재생 전에 전체 오디오를 미리 로딩할 수 있는 경우에 사용된다. 그러므로 큰 오디오 파일을 재생하기에는 많은 메모리가 필요한 단점이 있다.



잠깐!

자바의 오디오 API에는 MP3 오디오를 재생할 수 있는 기능을 제공하지 않는다. 미래에는 제공될 수 있을지도 모르겠다. MP3 오디오를 자바로 재생하려면 복잡한 코드를 작성해야 한다. 한편 JavaFX 라이브러리는 MP3 오디오를 쉽게 재생하는 클래스를 제공하니 참고하기 바란다.

오디오 재생/중지 예제 14-12

play, stop, play again의 3 버튼을 가진 응용프로그램을 작성하고, 애국가.wav 파일을 연주하라. 애국가.wav는 프로젝트 폴더 밑의 audio 폴더에 있어야 한다.




```
1  import javax.sound.sampled.*;
2  import javax.swing.*;
3  import java.awt.*;
4  import java.awt.event.*;
5  import java.io.*;
6
7  public class AudioEx extends JFrame {
8      private JButton btns[] = { new JButton("play"),
9                  new JButton("stop"),
10                 new JButton("play again")};
11     private Clip clip;
12
13     public AudioEx() {
14      setTitle("오디오 제어");
15         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
16         setSize(300,150);
17         Container c = getContentPane();
18         c.setLayout(new FlowLayout());
19
20         MyActionListener al = new MyActionListener();
21         for(int i=0; i<btns.length; i++) {
22             c.add(btns[i]);
23             btns[i].addActionListener(al); // 버튼에 Action 리스너 등록
24         }
25         setVisible(true);
26         loadAudio("audio/애국가1절.wav");
27     }
28
29     private void loadAudio(String pathName) {
30         try {
31             clip = AudioSystem.getClip(); // 비어있는 오디오 클립 만들기
32             File audioFile = new File(pathName); // 오디오 파일의 경로명
33             AudioInputStream audioStream =
                   AudioSystem.getAudioInputStream(audioFile); // 오디오 파일로부터
34             clip.open(audioStream); // 재생할 오디오 스트림 열기
35         }
36         catch (LineUnavailableException e) { e.printStackTrace(); }
37         catch (UnsupportedAudioFileException e) { e.printStackTrace(); }
38         catch (IOException e) { e.printStackTrace(); }
```

```
39      }
40
41      class MyActionListener implements ActionListener {
42          public void actionPerformed(ActionEvent e) {
43              switch(e.getActionCommand()) {
44                  case "play": clip.start(); // 오디오 재생 시작
45                      break;
46                  case "stop": clip.stop(); // 오디오 재생 중단
47                      break;
48                  case "play again":
49                      clip.setFramePosition(0); // 재생 위치를 첫 프레임으로 변경
50                      clip.start(); // 오디오 재생 시작
51                      break;
52              }
53          }
54      }
55
56      public static void main(String [] args) {
57          new AudioEx();
58      }
59  }
```

오디오 클립에 Line 이벤트 처리

Line 이벤트

오디오 클립(Clip 객체)은 오디오가 재생되는 도중 여러 상황에서 Line 이벤트가 발생한다. 이것은 Clip 클래스가 Line 클래스를 상속받았기 때문이다.

● Line 이벤트와 LineListener 인터페이스

LineListener

오디오가 재생을 시작할 때, 재생이 중단되었을 때, 오디오 클립이 닫혔을 때, 오디오 클립 객체에 Line 이벤트가 발생한다. Line 이벤트를 처리하는 리스너는 LineListener 인터페이스이며, Line 이벤트가 발생하면 다음 메소드가 호출된다.

> *public void update(LineEvent e)*
> 오디오가 재생을 시작할 때, 재생이 중단되었을 때, 오디오 클립이 닫혔을 때 호출된다.

● LineEvent 객체

LineEvent 객체

LineEvent 객체는 다음 메소드를 통해 이벤트에 대한 여러 정보를 제공한다.

Line getLine()
이벤트가 발생하는 오디오 클립 객체(Clip은 Line 인터페이스 상속받음)
long getFramePosition()
오디오 내에 이벤트가 발생한 프레임 위치(0부터 시작)
LineEvent.Type getType()
이벤트 종류로 다음 4가지 값 중 하나 리턴
- LineEvent.Type.OPEN: 오디오 클립이 열릴 때(Clip의 open() 호출시)
- LineEvent.Type.START: 재생이 시작될 때(Clip의 start() 호출시)
- LineEvent.Type.STOP: 재생이 중단될 때(Clip의 stop()이나 끝까지 재생되었을 때)
- LineEvent.Type.CLOSE: 오디오 클립이 닫히고 모든 자원이 반환되었을 때(Clip의 close() 호출시)

● 오디오 클립 객체에 Line 이벤트 리스너 달기

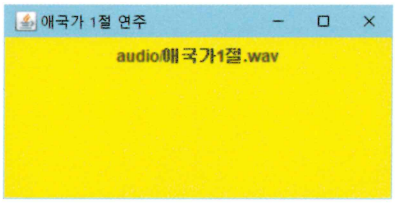
Clip의 addLineListener() 메소드를 이용하여 다음과 같이 리스너를 등록한다.

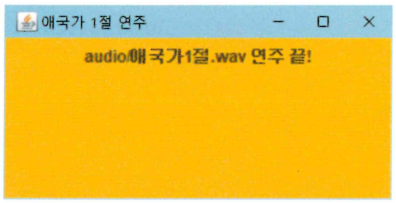
addLineListener()

```
clip.addLineListener(new MyLineListener());
```

예제 14-13 오디오 연주가 끝나면 컨텐트팬의 배경색을 ORANGE 색으로 만들기

애국가1절.wav 파일을 재생시키고 연주가 끝나면 컨텐트팬의 바탕색이 ORANGE 색으로 바뀌도록 하라.






```
1  import javax.sound.sampled.*;
2  import javax.swing.*;
3  import java.awt.*;
4  import java.io.*;
5
6  public class LineEventEx extends JFrame {
7     private Clip clip;
8     private String song="audio/애국가1절.wav";
9     private JLabel label=new JLabel(song);
10
11    public LineEventEx() {
12       setTitle("애국가 1절 연주");
13       setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

```
14        Container c = getContentPane();
15        c.setLayout(new FlowLayout());
16        c.setBackground(Color.YELLOW);
17        c.add(label);
18        setSize(300,150);
19        setVisible(true);
20        loadAudio(song);
21     }
22
23     private void loadAudio(String pathName) {
24        try {
25           File audioFile = new File(pathName); // 오디오 파일의 경로명
26           final AudioInputStream audioStream = AudioSystem.getAudioInputStream
                                                   (audioFile); // 오디오 파일로부터
27
28           clip = AudioSystem.getClip(); // 비어있는 오디오 클립 만들기
29           clip.addLineListener(new LineListener() {
30              public void update(LineEvent e) {
31                 if (e.getType() == LineEvent.Type.STOP) { // clip.stop()이 호출되
                                                              거나 재생이 끝났을 때
32                    try {
33                       getContentPane().setBackground(Color.ORANGE);
34                       label.setText(song + " 연주 끝!");
35                       audioStream.close();
36                    } catch (IOException e1) {
37                       e1.printStackTrace();
38                    }
39                 }
40              }
41           });
42           clip.open(audioStream); // 재생할 오디오 스트림 열기
43           clip.start(); // 재생 시작
44        }
45        catch (LineUnavailableException e) { e.printStackTrace(); }
46        catch (UnsupportedAudioFileException e) { e.printStackTrace(); }
47        catch (IOException e) { e.printStackTrace(); }
48     }
49
50
51     public static void main(String [] args) {
52        new LineEventEx();
53     }
54  }
```

요약



SUMMARY

- 메뉴를 만들기 위해 사용되는 클래스는 JMenuBar, JMenu, JMenuItem이며 JMenuBar는 메뉴바를, JMenu는 하나의 메뉴를, JMenuItem은 하나의 메뉴아이템을 구현한다. 메뉴바를 프레임에 붙이기 위해서는 JFrame.setJMenuBar(JMenuBar) 메소드를 이용한다.
- 사용자가 메뉴아이템을 선택하면 Action 이벤트가 발생한다.
- JToolBar는 바(bar) 모양으로서 툴바라고 불리며 다양한 종류의 스윙 컴포넌트를 담을 수 있는 컨테이너이다. 주로 사용자에게 아이콘 메뉴를 제공하기 위해 사용된다.
- 툴팁은 컴포넌트 위에 마우스를 올리면 출력되는 문자열로서, 모든 JComponent들이 가질 수 있으며, 툴팁을 달기 위해서는 JComponent의 setToolTipText(String msg)를 이용한다. msg는 툴팁 문자열이다.
- JDialog를 이용하여 독자가 원하는 형태의 다이얼로그를 제작할 수 있다.
- JOptionPane을 이용하여 다양한 팝업 다이얼로그를 출력할 수 있다.
- 입력 다이얼로그는 JOptionPane.showInputDialog() 메소드를 호출하면 출력되고, 한 줄의 입력창을 통해 문자열을 입력받을 때 사용된다.
- 확인 다이얼로그는 JOptionPane.showConfirmDialog() 메소드를 호출하면 출력되고, 사용자로부터 "예", "아니오" 등의 답을 얻는 데 사용된다.
- 메시지 다이얼로그는 JOptionPane.showMessageDialog() 메소드를 호출하면 출력되고, 사용자에게 임의의 메시지를 전하기 위해 사용된다.
- JFileChooser는 파일 열기 다이얼로그와 파일 저장 다이얼로그를 생성하고 출력하여 사용자로부터 파일이나 디렉터리를 선택하도록 한다.
- JColorChooser는 사용자에게 색상 팔레트를 보여주고 색을 선택하도록 하는 컬러 다이얼로그를 생성하고 출력하는 데 사용된다.
- JTabbedPane은 탭이라고 불리는 여러 개의 패널을 겹치게 하여 출력 공간을 공유하도록 하는 스윙 컴포넌트이다.
- Clip은 오디오 클립 객체로서, 오디오 파일로부터 오디오 데이터를 읽어오는 오디오 스트림과 연결하여 오디오를 재생하고 중단하는 기능을 제공한다.
- Clip 클래스는 Line 클래스를 상속받은 클래스로서, 오디오 클립의 오디오가 재생을 시작할 때, 중단될 때, 오디오 클립이 닫혔을 때, Line 이벤트를 받는다.



커피 자판기 시뮬레이터 만들기

Open Challenge

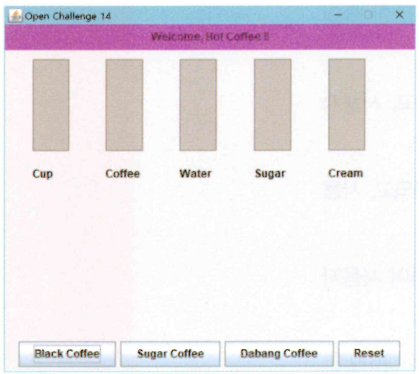
목적

실전적이고 종합적인 스윙 프로그램 작성 연습

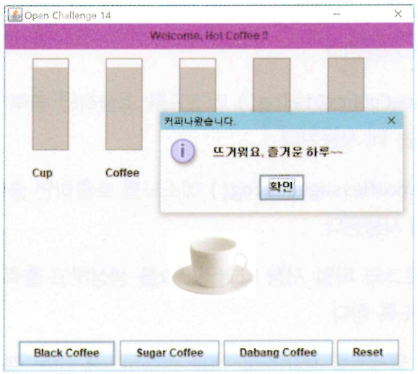
커피 자판기 시뮬레이터를 작성해보자. 커피 자판기 시뮬레이터는 실제 커피 자판기의 기능을 축소하여 다음 기능을 가진다. 난이도 8

- 커피의 종류는 커피만 들어 있는 Black Coffee, 커피와 설탕이 들어 있는 Sugar Coffee, 커피, 설탕, 크림이 모두 들어 있는 Dabang Coffee의 3가지로 한다.
- 화면에는 컵, 커피, 물, 설탕, 크림의 현재 양을 보여주며 커피를 뽑을 때마다 이 값들이 모두 조절된다.
- Reset 버튼을 두고, 이 버튼을 누르면 컵, 커피, 물, 설탕, 크림이 통에 가득 채워진다.
- 커피를 선택하였을 때 재료가 부족하면 커피를 먹을 수 없다는 경고 창을 출력한다.
- 커피를 선택하였을 때 커피 이미지를 출력하고 경고 창을 이용하여 알린다.

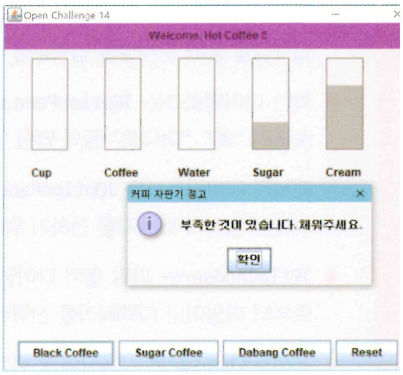
커피 자판기 시뮬레이터를 실행한 결과 화면은 다음과 같다.



초기 화면



Black Coffee 버튼을 누른 경우



여러 번 커피를 먹어서 Cup, Coffee, Water 통이 비어 있는 상태에서 버튼을 누른 경우

힌트 Hint

- 컨텐트팬의 NORTH에는 "Welcome, Hot Coffee!!" 레이블을 가진 패널을 만들어 붙인다.
- 컨텐트팬의 SOUTH에는 4개의 버튼을 부착한 패널을 만들어 붙인다.
- 컨텐트팬의 CENTER에는 커피 자판기 시뮬레이터의 핵심 기능을 가진 패널을 만들어 붙인다. 3개의 커피 버튼이 눌러지면 다음의 절차가 작동하도록 한다.
  1. 어떤 버튼이 눌러졌는지 판단한다.
  2. 버튼에 따라 현재 필요한 재료가 모두 있는지 판단한다. 재료가 모두 있으면 각 재료를 하나씩 감소시키고, 커피잔을 화면에 출력한다. JOptionPane.showMessageDialog()를 이용하여 메시지를 출력한다.

연습문제

EXERCISE



이론문제

• 홀수 문제는 정답이 공개됩니다.

1. 메뉴를 만드는 데 사용되는 클래스가 아닌 것은?

① JMenu ② JMenuItem ③ JMenuBar ④ Separator

2. 다음 프로그램의 빈칸을 채워라.

```
JMenuItem item1 = new JMenuItem("Open");
JMenuItem item2 = new JMenuItem("Save");
item1.addActionListener(new MyAction());
item2.addActionListener(new MyAction());

class MyAction implements _____________{
    public void actionPerformed(ActionEvent e) {
        String s = e. _____________________();
        if(s.equals("Open")) ......; // 열기 작업을 수행한다.
        else ....; // 닫기 작업을 수행한다.
    }
}
```

3. 툴바에 대해 잘못 설명한 것은?

① 툴바는 컨테이너이다.
② JToolBar를 이용하여 툴바를 만든다.
③ 툴바는 반드시 BorderLayout 배치관리자를 가진 컨테이너에만 붙는다.
④ 툴바의 핸들을 마우스로 드래깅할 수 없게 만드는 메소드는 JToolBar의 setEnabled (false)이다.

4. 툴팁에 대해 잘못 말한 것은?

① 툴팁은 JComponent를 상속받은 모든 스윙 컴포넌트에 달 수 있다.
② 툴팁은 반드시 문자열만 사용된다.
③ 툴팁이 나타나지 않도록 설정하기 위해서는 JToolTip의 setEnabled(false) 메소드를 사용한다.
④ 툴팁의 지속 시간을 제어할 수 있으며 설정된 시간은 모든 툴팁에 일괄적으로 적용된다.

5. 다음 버튼에 "안녕하세요"라는 툴팁을 만드는 코드를 작성하라.

```
JButton b = new JButton("Hello");
_____;
```

6. 다음 JLabel 컴포넌트에 마우스를 올리면 1초 후에 "자바"라는 툴팁이 나타나서 10초 동안 지속되도록 아래의 빈 줄에 코드를 삽입하라.

```
JLabel la = new JLabel("Java");
_____
_____
_____
_____
```

7. 일반적으로 파일 열기 다이얼로그를 모달 다이얼로그로 만드는 이유가 무엇인지 설명하라.

8. 다이얼로그와 관련이 제일 먼 것은 무엇인가?

① JDialog ② JOptionPane
③ JFileChooser ④ FileNameExtensionFilter
⑤ JTabbedPane ⑥ JColorChooser

9. 다음 소스에 의해 각각 어떤 창이 만들어지는지 그려라.

(1) `JOptionPane.showInputDialog("type ID");`

(2)
```
JOptionPane.showConfirmDialog(null, "Continue?", "확인하고자 합니다."
    JOptionPane.YES_NO_CANCEL_OPTION);
```

10. 다음 설명 중 틀린 것은 무엇인가?

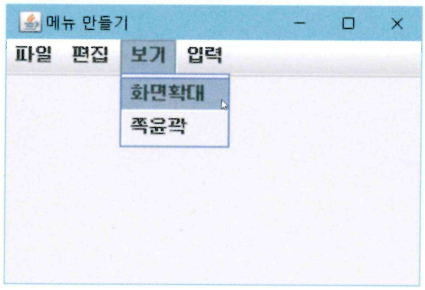
① 컬러 다이얼로그는 모달 타입이다.
② 사용자가 파일 열기 다이얼로그 창을 닫으면 다이얼로그가 화면에서 사라지지만 다이얼로그 객체는 사라지지 않기 때문에 재사용이 가능하다.
③ 파일 저장 다이얼로그에서 파일 이름을 선택한 후 "닫기" 버튼을 선택하여 정상적으로 닫기가 이루어진 경우, JFileChooser.saveFileDialog() 메소드는 JFileChooser.APPROVE_OPTION을 리턴한다.
④ 탭팬에서 탭의 위치는 디폴트 값이 왼쪽, 즉 JTabbedPane.LEFT이다.

실습문제

• 홀수 문제는 정답이 공개됩니다.

1. "파일", "편집", "보기", "입력" 등의 4가지 메뉴를 가진 스윙 프로그램을 작성하라. "보기" 메뉴에만 "화면확대", "쪽윤곽"의 2개의 메뉴아이템이 있다. 난이도 3

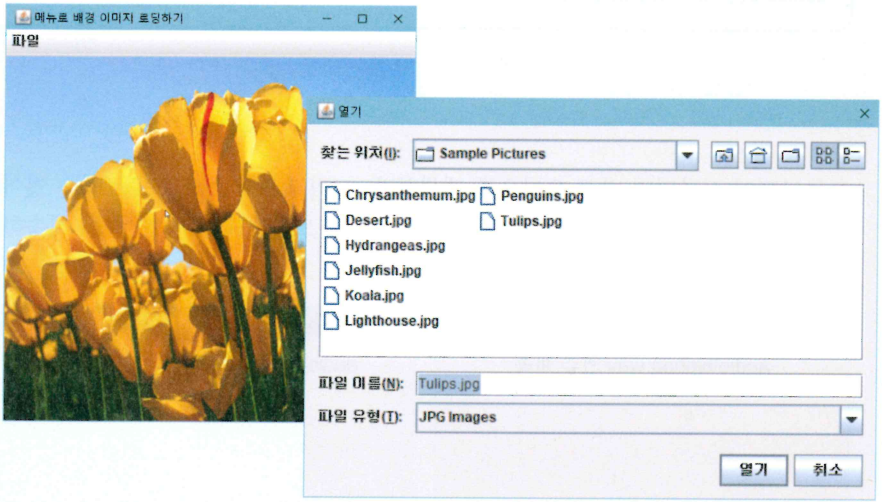
목적 간단한 메뉴 만들기



2. "파일" 메뉴에 "열기" 메뉴아이템을 하나 만든다. 이 메뉴아이템을 선택하면 파일 열기 다이얼로그를 출력하고 사용자가 JPG 이미지 파일을 선택하면 이 이미지를 프레임의 바탕화면 전체(컨텐츠팬)에 그리는 스윙 응용프로그램을 작성하라. 이미지를 그리기 위해서 JLabel을 이용하지 말고 Graphics의 drawImage() 메소드를 이용하라.

난이도 5

목적 메뉴 선택 시 이벤트 JFileChooser, 이미지 출 습

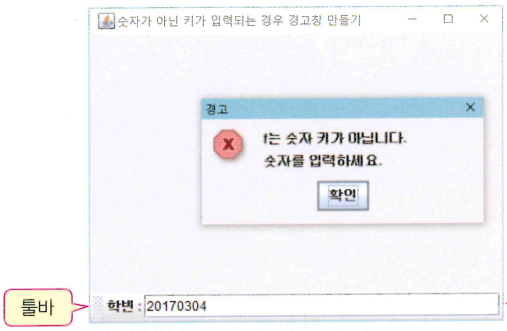


3. 툴바를 만들고 "종료" 버튼을 하나 단다. 이 버튼을 선택하면 JOptionPane을 이용하여 YES_NO_OPTION에 "정말 종료하시겠습니까?" 메시지를 출력하는 확인 다이얼로그를 출력하라. 그리고 사용자가 "예(Y)"로 답한 경우에만 응용프로그램을 종료하는 스윙 응용프로그램을 작성하라. 난이도 4

목적 JToolBar와 JOpt Pane 다루기 연습

목적 JToolBar와 JOption Pane 연습

4. 툴바에 그림과 같이 학번을 입력하는 텍스트필드 컴포넌트를 삽입하고 툴바를 프레임의 하단에 부착하라. 이 텍스트필드 컴포넌트에는 오직 숫자만이 입력되도록 하기 위해 사용자가 숫자가 아닌 키를 입력하면 그림과 같은 경고 메시지를 가진 경고 창을 출력하도록 하라. 그림에서는 20170304 뒤에 f 키가 입력된 경우 출력되는 경고 창을 보여준다. 난이도 5

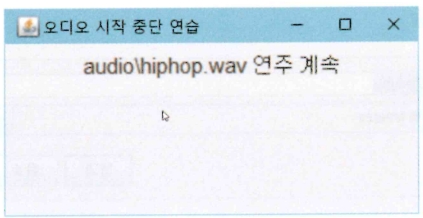


힌트 Hint

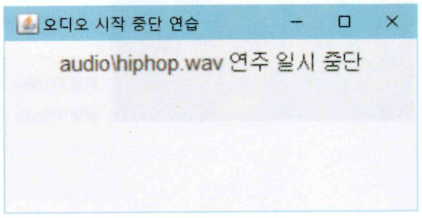
텍스트필드 컴포넌트에 Key 이벤트 리스너의 keyTyped() 메소드를 작성하여 키를 구분하라. 숫자 키가 아닌 키가 입력된 경우 이 키 이벤트를 무력화시키기 위해서는 KeyEvent의 consume() 메소드를 호출하면 된다.

목적 Clip, AudioInputStream, AudioSystem 클래스로 오디오 제어

5. 프로그램이 시작되면 바로 오디오를 재생하라. 그리고 마우스가 프로그램을 벗어나면 연주를 일시 중단시키고, 다시 마우스가 프로그램으로 올라오면 연주를 계속하도록 하라. 난이도 5



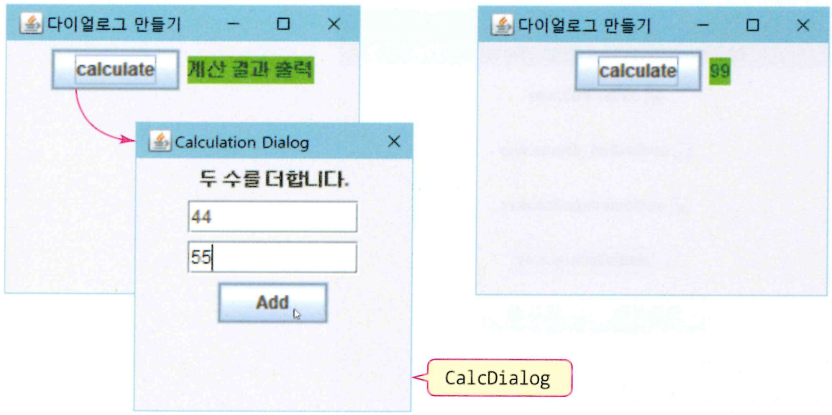
마우스가 컨텐트팬 내에 있는 경우



마우스가 컨텐트팬을 벗어난 경우

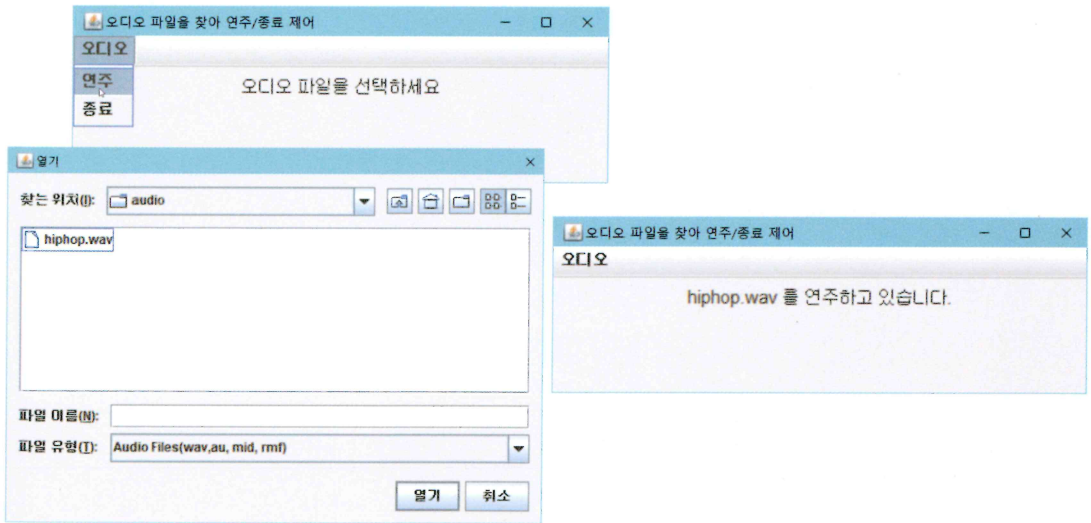
6. "Calculate" 버튼과 레이블을 가진 프레임을 작성하라. JDialog를 상속받아 CalcDialog 다이얼로그를 다음 그림과 같이 구현하고, "Calculate" 버튼을 클릭하면 CalcDialog가 출력되도록 하라. 사용자로부터 두 정수를 입력받고 "Add" 버튼을 클릭하면 그 결과 값이 레이블에 출력되도록 하라. CalcDialog는 모달 다이얼로그로 만들어라. 난이도 6

목적 JDialog로 다이얼로그를 만들고 다루기



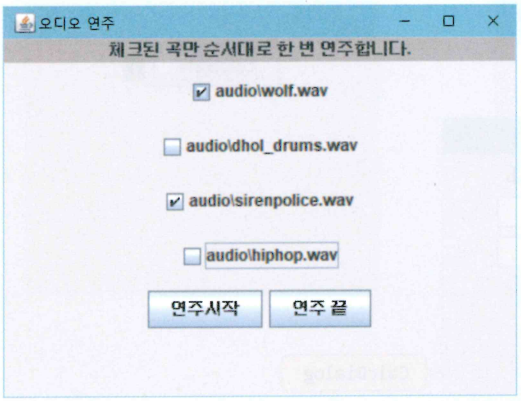
7. 메뉴를 이용하여 오디오를 연주하는 프로그램을 작성하라. "오디오" 메뉴에 "연주"와 "종료" 메뉴아이템을 두고, "연주" 메뉴아이템이 선택되면 JFileChooser를 이용하여 wav 파일을 선택하여 연주를 시작하고, "종료" 메뉴아이템이 선택되면 연주를 종료한다. 난이도 7

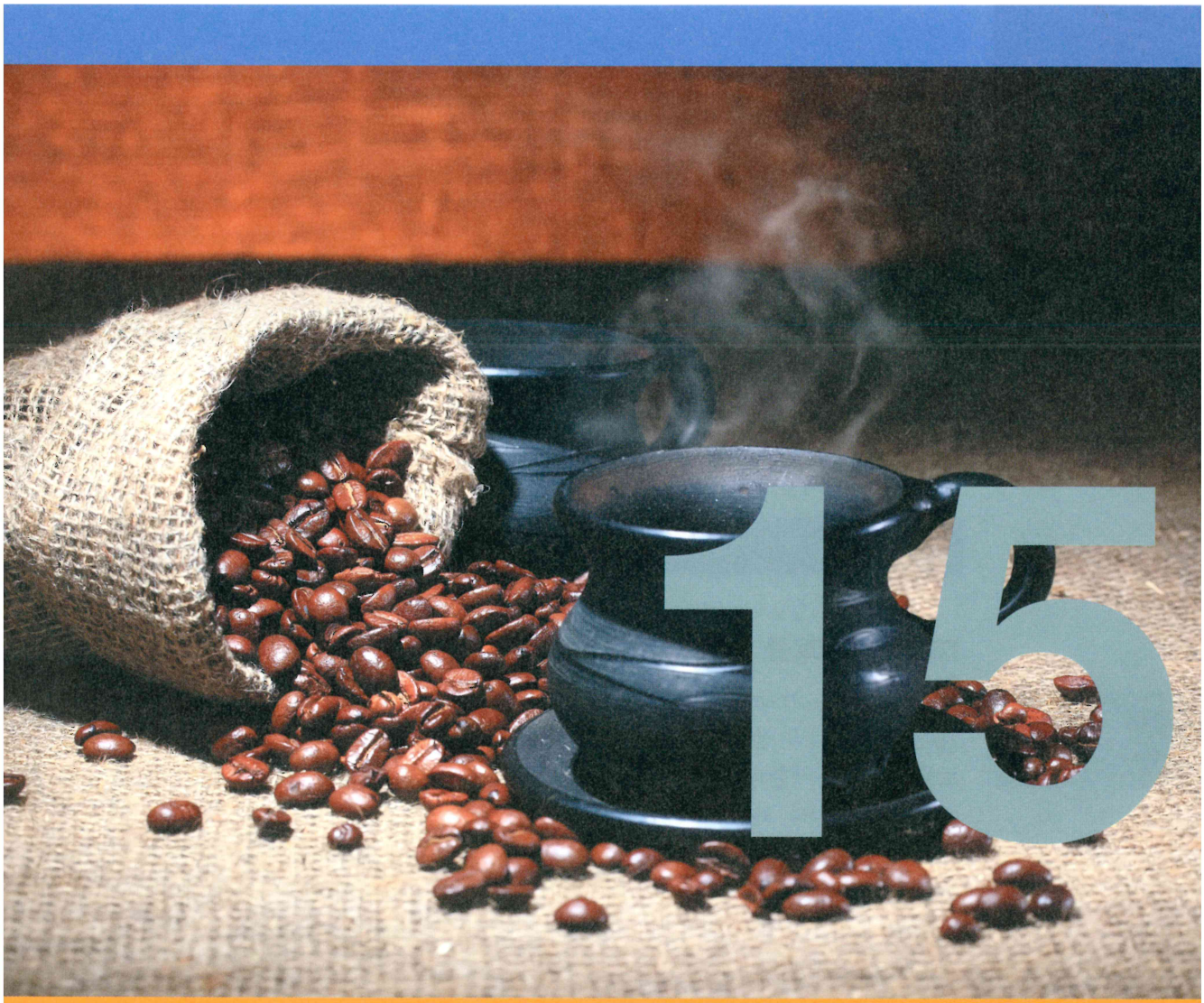
목적 Clip, AudioSystem으로 오디오 제어



목적 체크박스, 오디오 재생, LineListener 다루기

8. 다음 실행 예시와 같이 연주할 곡을 체크박스로 만들고 사용자가 체크한 곡만 순서대로 연주하는 프로그램을 작성하라. 연주시작 버튼을 누르면 연주가 시작되고 다음 곡에 대해서는 체크박스를 선택/해제할 수 있다. 연주되고 있는 곡명의 글자색을 달리하는 것은 옵션이다. 연주끝 버튼을 누르면 연주가 멈추고, 연주시작 버튼을 누르면 처음부터 시작한다. 연주될 wav 파일은 프로젝트의 audio 폴더에 두면 된다. 난이도 8



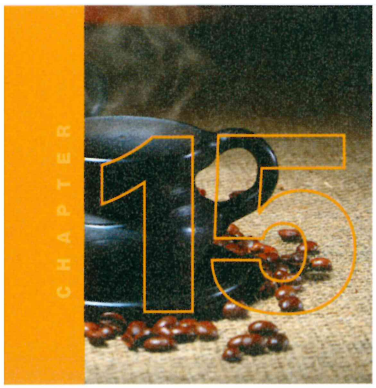


15

네트워크

Objectives

- TCP/IP 프로토콜의 개념을 이해한다.
- 자바의 소켓과 포트의 개념을 이해한다.
- 서버 클라이언트 통신 프로그램의 구조를 이해한다.
- 서버 소켓과 클라이언트 소켓을 구분하여 이해한다.
- 간단한 채팅 프로그램 예제를 통해 소켓 통신을 이해한다.
- 수식 계산 서버-클라이언트 통신 예제로 서버-클라이언트 통신을 이해한다.
- 소켓 프로그래밍을 할 수 있다.



CHAPTER 15

JAVA PROGRAMMING

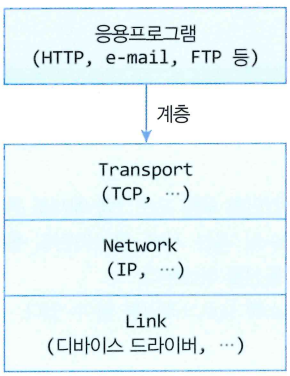
네트워크

15.1 TCP/IP

TCP/IP 프로토콜 소개

TCP 프로토콜
IP 프로토콜

TCP 프로토콜은 Transmission Control Protocol의 약자로 다른 두 시스템 간에 신뢰성 있는 데이터의 전송을 관장하는 통신 프로토콜로서 IP(Internet Protocol) 프로토콜 위에서 동작한다. TCP 프로토콜을 사용하는 응용프로그램으로는 e-mail, FTP, 웹(HTTP) 등이 있다. IP는 패킷 교환 네트워크에서 송신 호스트와 수신 호스트가 데이터를 주고받는 것을 관장하는 프로토콜로서 TCP의 하위 레벨 프로토콜이다. TCP는 IP 기능을 활용하여 두 시스템 사이에 데이터가 손상 없이 안전하게 전송되도록 하며, TCP와 IP를 묶어 TCP/IP로 표기한다. TCP/IP 프로토콜 및 e-mail, 웹 응용프로그램의 관계는 [그림 15-1]과 같다.



[그림 15-1] 네크워크 계층

IP 주소

IP 주소

네트워크 장치의 주소

IP 주소는 네트워크상에서 유일하게 식별될 수 있는 네트워크 장치의 주소로서, 예를 들면 192.156.11.15와 같이 4개의 숫자가 '.'으로 연결된다. 하나의 숫자 범위는 0~255로서 한 바이트로 표현이 가능하다. IP 주소는 마치 전화번호나 집주소와 같아 이 주소를 통해 네트워크에 연결된 장치를 식별할 수 있으며, 동일한 주소를 여러 네트워크 장치에 중복해서 사용할 수 없다. 숫자로 된 주소는 기억하기 어려우므로 www.naver.com과 같은 문자열로 구성된 도메인 이름으로 바꿔 사용한다. 사용자가 문자열로 구성된 도메인 이름을 사용하면 DNS(Domain Name Server) 서버에 의해 숫자로 구성된 IP 주소로 자동 변환되게 된다.

현재는 4개의 숫자로 구성된 IP 주소를 표현하기 위해 32비트의 IP 버전 4(IPv4)가 사용되고 있다. 그러나 세계적으로 네트워크 장치의 개수가 폭발적으로 증가하여 각 장치에 고유하게 부여할 수 있는 IP 주소가 고갈됨에 따라 128비트의 IP 버전 6(IPv6)이 점점 사용되는 추세이다.

localhost

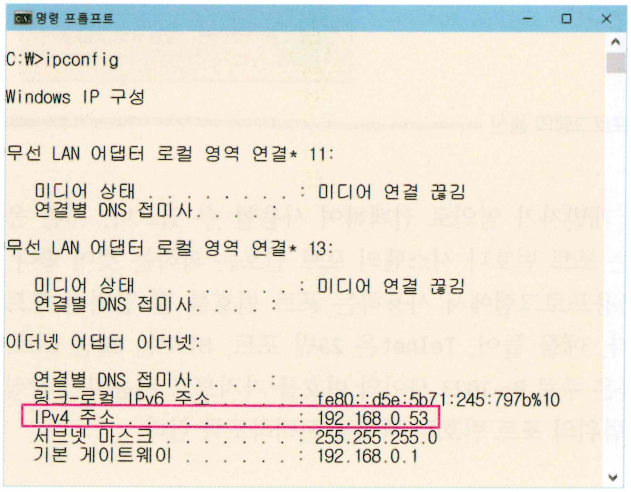
127.0.0.1

자신의 컴퓨터에서 자신의 IP 주소를 간단히 localhost라는 이름으로 사용해도 된다. localhost의 IP 주소는 127.0.0.1로 정해져 있다.

● 내 컴퓨터의 IP 주소 확인하기

ipconfig

윈도우 PC에서는 명령창을 열어 ipconfig 명령을 수행하면 [그림 15-2]와 같이 컴퓨터의 IP 주소를 확인할 수 있다. 대학이나 연구소 등에서는 한 컴퓨터에 항상 동일한 IP(고정 IP)를 설정하는 경우가 많지만, 가정에서는 대개 무선 공유기가 자동으로 할당해 주는 IP 주소를 부여받는다. 이를 유동 IP라고 부른다.



[그림 15-2] 내 컴퓨터 IP 주소 확인

포트

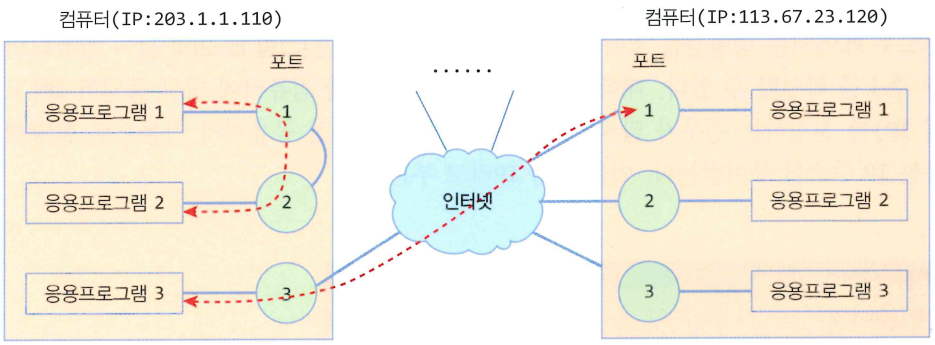


포트

IP 주소는 네트워크상에 있는 한 컴퓨터를 유일하게 식별한다 하지만, 한 컴퓨터에는 여러 응용프로그램이 네트워크를 사용하고 있기 때문에, IP 주소만 가지고는 통신하고자 하는 응용프로그램을 식별할 수 없다. 이를 위해 한 컴퓨터 내의 각 응용프로그램은 통신을 위해 가상의 연결단인 포트(port)를 생성하고 이 포트 번호로 상대방이 자신을 식별하게 한다.

IP 주소는 아파트의 동 번호와 같고, 포트 번호는 그 동에 있는 호 번호에 비유할 수 있다. 다른 예로 은행의 사례를 들면, IP 주소는 은행 지점의 주소이고, 포트 번호는 은행 내의 고객 창구 번호와 같다.

따라서 통신을 수행하는 모든 응용프로그램은 IP 주소와 포트를 이용하여 상대편 통신 프로그램을 인지하며 데이터를 교환한다. 물론 이때 상대편 응용프로그램은 자신의 IP 주소와 포트 번호를 알고 통신 접속이나 데이터가 오기를 기다리고 있어야 한다. 응용프로그램과 포트 사용 사례는 [그림 15-3]과 같다.



[그림 15-3] 포트를 이용한 두 응용프로그램의 통신

잘 알려진 포트

포트 번호는 응용프로그램 개발자가 임의로 선택하여 사용할 수 있으나, 기존 응용프로그램에서 사용하고 있는 포트 번호나 시스템의 포트 번호는 피하는 것이 좋다. 시스템이나 기존에 알려진 응용프로그램에서 사용하는 포트 번호를 잘 알려진 포트(well-known ports)라고 한다. 예를 들어, Telnet은 23번 포트, HTTP는 80번 포트, FTP는 21번 포트 등이며, 이들은 주로 0~1023 사이의 번호를 가지므로 사용자가 작성하는 응용프로그램에서는 이 범위의 포트 번호는 피해서 선택하도록 한다.

CHECK TIME

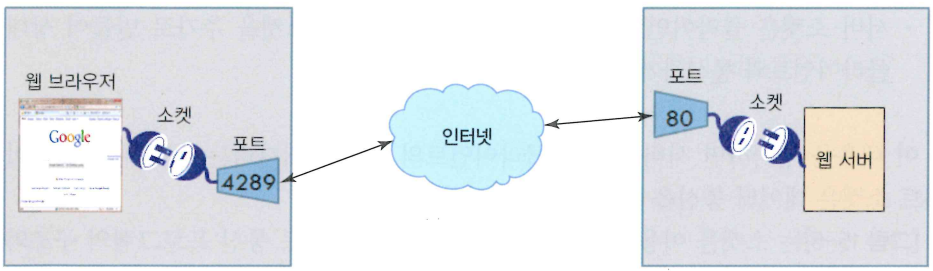
1. IP 주소와 포트에 대해 설명하라.
2. 현재 자신의 컴퓨터의 IP 주소가 얼마인지 확인하라.
3. 한 컴퓨터에서 2개의 통신 응용프로그램이 동일한 포트 번호를 사용한다면 어떤 문제가 발생할 수 있는가?
4. 하나의 통신 응용프로그램은 반드시 하나의 포트만 사용하여야 하는가?

15.2 소켓 프로그래밍

소켓(socket)

소켓 통신
소켓

소켓 통신은 개발자가 TCP/IP 네트워크를 이용하여 쉽게 통신 프로그램을 작성하도록 지원하는 기반 기술이다. 여기서 소켓은 통신하는 두 응용프로그램 간의 통신 링크의 각 끝단(endpoint)으로서, TCP/IP의 네트워크 기능을 활용하여 다른 컴퓨터의 소켓과 데이터를 주고받는다. 소켓을 활용하는 통신의 모양은 [그림 15-4]와 같으며, 소켓은 특정 포트에 연결되어 데이터를 보내거나 받을 때 해당 응용프로그램을 식별한다.



[그림 15-4] 소켓을 이용하는 통신 사례

응용프로그램은 소켓과 연결한 후 소켓에 데이터를 주기만 하면, 소켓이 상대방 -
용프로그램에 연결된 소켓에 데이터를 보낸다. 또는 응용프로그램은 연결된 소켓으
부터 도착한 데이터를 단순히 받기만 하면 된다. 인터넷을 경유하여 데이터를 주고
는 기능은 순전히 소켓의 몫이다. 데이터를 주고받는 동안 전송받은 데이터에 오류
없는 지 검사하고, 만일 손상된 데이터가 오면 다시 받기를 요청하는 등 온전한 데
터를 받는 과정은 모두 소켓의 몫이다.

소켓과 서버 클라이언트 통신

서버
클라이언트

소켓을 이용하는 통신에서는 반드시 서버 응용프로그램과 클라이언트 응용프로그램
로 구분된다. 정보를 제공하는 쪽을 서버(server)라고 부르며, 정보를 이용하는 쪽
클라이언트(client)라고 부른다. 통신은 서버가 먼저 클라이언트의 접속을 기다리고
클라이언트에서 서버에 접속하면, 그 때부터 서버나 클라이언트가 데이터를 서로
고받을 수 있다. 서버나 클라이언트가 보내는 순서를 정하거나 순서에 상관없이 데
터를 전송하는 것은 개발자가 프로그램을 작성하기에 달려 있다.

● 서버 소켓과 클라이언트 소켓

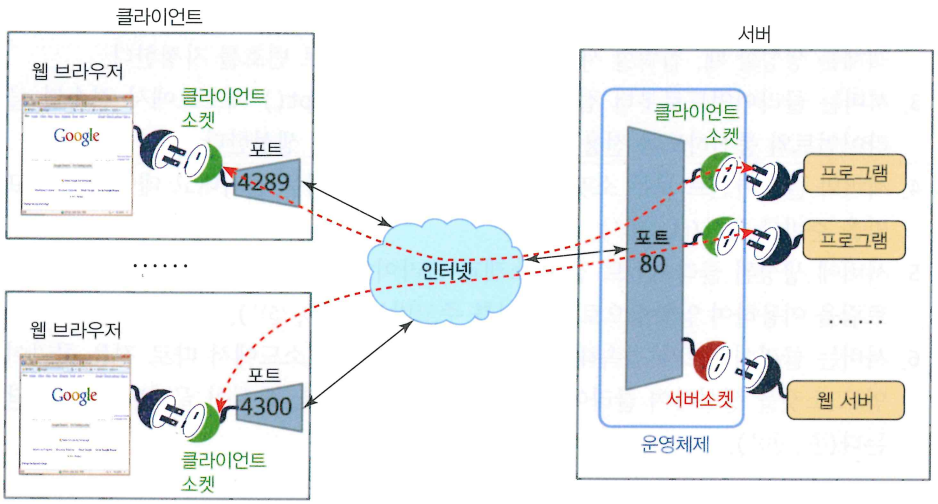
서버 소켓
클라이언트 소켓

소켓에는 서버 소켓과 클라이언트 소켓의 2가지 종류가 있다. 이 둘은 다음과 같이
도가 서로 다르다.

- 서버 소켓은 서버 응용프로그램이 사용자의 접속을 기다리는(listen) 목적으
만 사용된다.
- 클라이언트 응용프로그램에서는 클라이언트 소켓을 이용하여 서버에 접속한다.
- 서버 소켓은 클라이언트가 접속해오면, 클라이언트 소켓을 추가로 만들어 상
클라이언트와 통신하게 한다.

이 내용을 정리하면 서버 소켓은 클라이언트의 접속을 기다리는 소켓이며, 클라
언트 소켓은 데이터 통신을 실시하는 소켓이다.

[그림 15-5]는 소켓을 이용한 전형적인 웹 서버 클라이언트 통신 프로그램의 구조
함께, 클라이언트 응용프로그램과 클라이언트 소켓, 그리고 서버 응용프로그램에
서버 소켓, 클라이언트 소켓의 관계를 보여준다.



[그림 15-5] 소켓을 이용한 웹 서버와 웹 클라이언트 사이의 통신

● 서버에서 클라이언트 소켓들의 포트 공유

포트 공유
운영체제

[그림 15-5]에서 서버 쪽의 통신프로그램은 각각 독립된 소켓을 이용하여 클라이언트와 통신을 수행한다. 한편, 그림의 서버 쪽을 자세히 들여다보면, '동일한 포트(80)를 여러 클라이언트 소켓들이 공유하고 있으면, 여러 클라이언트들로부터 전송받은 데이터를 서버 내 어떤 소켓으로 전달해야 하는지 어떻게 판단할까?' 하는 의문이 들 것이다. 이것은 운영체제에 의해 처리된다. 클라이언트가 처음 서버 소켓에 연결될 때, 운영체제는 연결된 클라이언트 IP 주소와 포트 번호를 저장하고 기억해둔다. 그 후 서버 컴퓨터의 운영체제는 클라이언트로부터 데이터 패킷을 받게 되면, 패킷 속에 들어 있는 클라이언트의 IP 주소와 포트 번호를 참고하여, 서버에 있는 클라이언트 소켓을 찾아 그 곳으로 데이터를 보낸다.

● 소켓을 이용한 서버 클라이언트 통신 프로그램 구성

자바로 작성하는 서버 응용프로그램과 클라이언트 응용프로그램의 구조를 전체적으로 알아보자. 구체적인 코드들은 다음 절에서 설명한다. 서버 클라이언트의 전형적인 구조는 [그림 15-6]과 같으며 동작하는 과정은 다음과 같다.

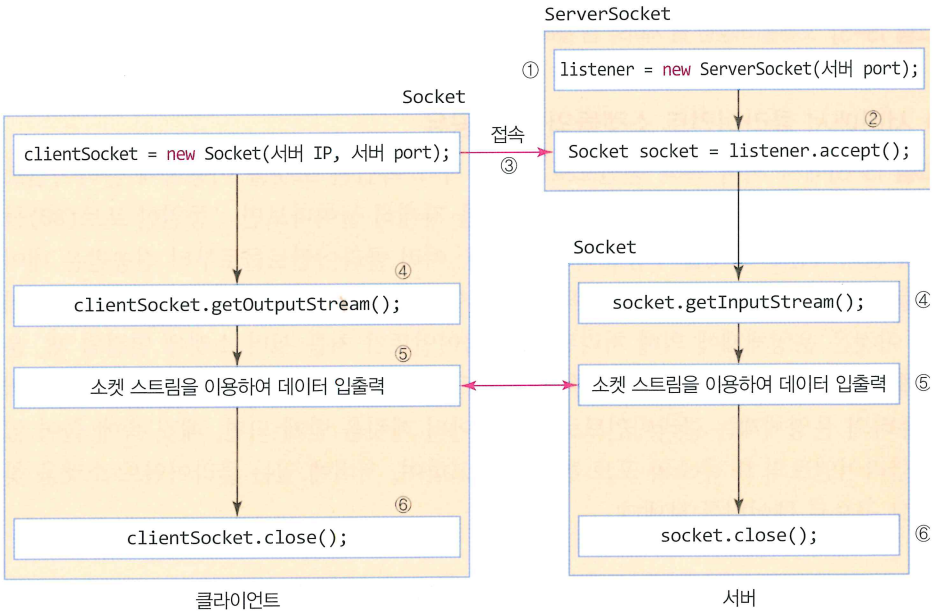
서버 소켓 객체
접속 기다린다

1. 서버 응용프로그램은 ServerSocket 클래스를 이용하여 서버 소켓 객체를 생성하고(new SeverSocket(서버 port)①) 클라이언트의 접속을 받기 위해 기다린다(②). 서버 소켓을 생성할 때 포트 번호를 주어 해당 포트로 접속해 오는 클라이언트 기다리게 한다.
2. 클라이언트 응용프로그램은 Socket 클래스를 이용하여 클라이언트 소켓 객체를

서버에 접속
클라이언트 소켓을 따로 생성
소켓을 닫는다

생성하고(new Socket(서버 IP, 서버 port)) 서버에 접속을 시도한다(③). 소켓 객체를 생성할 때, 접속할 서버 소켓의 IP 주소와 포트 번호를 지정한다.

3. 서버는 클라이언트로부터 접속 요청을 받으면, accept() 메소드에서 접속된 클라이언트와 통신하도록 전용 클라이언트 소켓을 따로 생성한다.
4. 서버와 클라이언트 모두 소켓으로부터 입출력 스트림을 얻어내고 데이터를 주고받을 준비를 한다(④, ④').
5. 서버에 생성된 클라이언트 전용 소켓과 클라이언트의 소켓이 상호 연결된 채 스트림을 이용하여 양방향으로 데이터를 주고받는다(⑤, ⑤').
6. 서버는 클라이언트가 접속해 올 때마다 accept() 메소드에서 따로 전용 클라이언트 소켓을 생성하여 클라이언트와 통신하도록 한다. 통신이 끝나면 소켓을 닫는다(⑥, ⑥').



[그림 15-6] 소켓을 이용한 서버 클라이언트 통신 프로그램의 전형적인 구조

Socket 클래스, 클라이언트 소켓

Socket
클라이언트 소켓

Socket은 java.net 패키지에 포함되어 있는 클래스로서 클라이언트 소켓을 구현한다. 즉, 서버와 통신하기 위해 클라이언트 응용프로그램에서 사용하는 소켓이다. Socket의 생성자와 메소드는 각각 〈표 15-1〉, 〈표 15-2〉와 같다.

Socket의 생성자는 연결할 서버의 IP 주소(또는 도메인 주소)와 포트 번호를 인자로 받아서 Socket 객체를 생성한다. 클라이언트 자신의 주소와 포트 번호가 아님에 주

의하라. 이제 Socket 클래스를 이용하여 클라이언트 응용프로그램을 작성하는 방법을 하나씩 알아보자.

● 클라이언트 소켓 생성 및 서버 접속

IP 주소가 128.12.1.1이고 포트 번호가 5550인 서버에 연결하기 위해 다음과 같이 클라이언트 소켓 객체를 생성한다. 이때 클라이언트의 포트(local port)는 사용되지 않는 포트 중에서 자동으로 선택된다.

```
Socket clientSocket = new Socket("128.12.1.1", 5550); // 128.12.1.1 서버에 접속
```

Socket 객체가 생성되면 곧바로 128.12.1.1 주소의 5550번 포트로 자동 접속이 이루어진다. 다음과 같이 빈 소켓 객체를 생성하고 서버에 접속해도 된다.

```
Socket clientSocket = new Socket(); // 연결되지 않는 소켓 생성
clientSocket.bind(new InetSocketAddress("192.168.1.21", 1234));
                    // 소켓에 자신의 IP 주소(192.168.1.21)와 로컬 포트(1234)를 결합한다.
clientSocket.connect(new InetSocketAddress("128.12.1.1", 5550));
                      // IP 주소가 128.12.1.1이고 포트가 5550인 서버 응용프로그램에 접속
```

● 네트워크 입출력 스트림 생성

소켓이 만들어지고 서버와 연결이 된 후에는 Socket 클래스의 getInputStream()과 getOutputStream() 메소드를 이용하여 서버와 데이터를 주고받을 소켓 스트림을 얻어내고 이를 버퍼 스트림에 연결한다.

getInputStream()
getOutputStream()

```
BufferedReader in = new BufferedReader(new
                        InputStreamReader(clientSocket.getInputStream()));
BufferedWriter out = new BufferedWriter(new
                        OutputStreamWriter(clientSocket.getOutputStream()));
```

지금부터는 in, out 스트림 객체를 이용하여 네트워크 데이터를 보내고 받으면 된다. 이 코드에서 만들어진 in, out은 문자만 보내고 받을 수 있는 문자 입출력 스트림이다.

● 서버로 데이터 전송

이제 버퍼 출력 스트림 out을 통해 데이터를 전송해보자. 다음은 "hello" 문자열을 서버로 전송하는 코드이다.

```
out.write("hello"+"\n");
out.flush();
```

out.flush()

"hello"에 "\n"을 덧붙여 보내는 이유는 서버 쪽에서 라인 단위('\n' 문자가 올 ㄴ까지 한 번에 읽는)로 수신한다고 가정하였기 때문이다. 스트림 out은 버퍼 입출력 스트림이므로 버퍼가 차기 전까지 데이터를 보내지 않기 때문에 강제로 out.flush()를 ㅈ출하여 스트림 속의 데이터를 모두 즉각 전송하도록 하였다.

● 서버로부터 데이터 수신

버퍼 입력 스트림 in을 이용하면 서버로부터 문자 데이터를 수신할 수 있다. 다음ㅅ 클라이언트로부터 한 개의 문자를 입력받는 코드이다.

```
int x = in.read(); // 클라이언트로부터 한 개의 문자 수신
```

한 행의 문자열을 입력받는 코드는 다음과 같다.

```
String line = in.readLine(); // 클라이언트로부터 한 행의 문자열 수신
```

in.readLine() 메소드는 '\n' 문자가 올 때까지 계속 읽고 '\n'이 도착하면 그때ㄲ지 읽은 문자열을 리턴한다. 이 문자열 속에는 '\n'이 삽입되지 않는다.

● 데이터 송수신 종료

데이터 송수신을 모두 수행하고 소켓 연결을 끊고자 하면 다음과 같이 한다.

```
socket.close();
```

〈표 15-1〉 Socket 클래스의 생성자

생성자	설명
Socket	연결되지 않은 상태의 소켓을 생성
Socket(InetAddress address, int port)	소켓을 생성하고, 지정된 IP 주소(addresss)와 포트 번호(port)에서 대기하는 원격 응용프로그램의 소켓에 연결
Socket(String host, int port)	소켓을 생성하여 지정된 호스트(host)와 포트 번호(port)에 연결한다. 호스트 이름이 null인 경우는 루프백(loopback) 주소로 가정

〈표 15-2〉
Socket 클래스의 주요 메

메소드	설명
void bind(SocketAddress bindpoint)	소켓에 로칼 IP 주소와 로컬 포트 지정(결합)
void close()	소켓을 닫는다.
void connect(SocketAddress endpoint)	서버에 연결
InetAddress getInetAddress()	소켓에 연결된 서버 IP 주소 반환
InputStream getInputStream()	소켓의 입력 스트림 반환. 이 스트림을 이용하여 소켓이 상대편으로부터 받은 데이터를 읽을 수 있음
InetAddress getLocalAddress()	소켓의 로컬 주소 반환
int getLocalPort()	소켓의 로컬 포트 번호 반환
int getPort()	소켓에 연결된 서버의 포트 번호 반환
OutputStream getOutputStream()	소켓의 출력 스트림 반환. 이 스트림에 출력하면 소켓이 서버로 데이터 전송
boolean isBound()	소켓이 로컬 주소와 결합되어 있으면 true 반환
boolean isConnected()	소켓이 서버에 연결되어 있으면 true 반환
boolean isClosed()	소켓이 닫혀있으면 true 반환
void setSoTimeout(int timeout)	데이터 읽기 타임아웃 시간 지정. 0이면 타임아웃 해제

ServerSocket 클래스, 서버 소켓

ServerSocket 클래스는 서버 소켓을 구현한다. ServerSocket 클래스는 java.net 패키지에 포함되어 있으며 ServerSocket 클래스의 생성자와 메소드는 각각 〈표 15-3〉, 〈표 15-4〉와 같다. ServerSocket은 클라이언트로부터 연결 요청을 기다리는 목적으로만 사용되며, 서버가 클라이언트의 연결 요청을 수락하면 Socket 객체를 별도로 생성하고, 이 Socket 객체가 클라이언트와 데이터를 주고받는다. ServerSocket은 데이터의 송수신에 사용되지 않는다.

ServerSocket
서버 소켓
연결 요청을 기다리는 목

● 서버 소켓 생성

ServerSocket 생성자는 포트 번호를 인자로 받아서 ServerSocket 객체를 생성한다. 이 포트 번호는 클라이언트의 접속을 기다릴 자신의 포트 번호이다. 이미 사용 중인 포트 번호를 지정하면 오류가 발생한다. 9999번 포트를 사용하는 서버 소켓을 생성하는 예를 들면 다음과 같다.

```
ServerSocket listener = new ServerSocket(9999);
```

● 클라이언트로부터 접속 대기

accept()

ServerSocket 클래스의 accept() 메소드를 이용하여 클라이언트로부터의 연결 요청을 기다린다. accept() 메소드가 연결을 수락하면 다음과 같이 Socket 객체를 하나 별도로 생성하여 리턴한다.

```
Socket socket = listener.accept();
```

서버에서 지금 접속된 클라이언트와의 데이터 통신은 새로 만들어진 socket을 이용하여 이루어진다. 새로 만들어진 socket은 서버 소켓과 동일하게 9999번 포트를 통해 데이터를 주고받는다.

● 네트워크 입출력 스트림 생성

getInputStream()
getOutputStream()

클라이언트로 데이터를 주고 받기 위한 스트림 객체는, ServerSocket의 accept() 메소드로부터 얻은 socket 객체의 getInputStream()과 getOutputStream() 메소드를 이용하여 얻어낸다. 다음과 같이 소켓 스트림을 버퍼 입출력 스트림에 연결하여 사용한다.

```
BufferedReader in = new BufferedReader(new
                        InputStreamReader(socket.getInputStream()));
BufferedWriter out = new BufferedWriter(new
                        OutputStreamWriter(socket.getOutputStream()));
```

버퍼 입출력 스트림 in, out을 이용하여 클라이언트와 데이터를 주고받으면 된다. in, out은 모두 문자만 입출력하는 스트림이다.

● 클라이언트로부터 데이터 수신

앞서 만들어진 버퍼 스트림 in을 이용하여 클라이언트로부터 문자 데이터를 수신할 수 있다. 다음은 클라이언트로부터 한 개의 문자를 입력받는 코드이다.

```
int x = in.read(); // 클라이언트로부터 한 개의 문자 수신
```

한 행의 문자열을 입력받는 코드는 다음과 같다.

```
String line = in.readLine(); // 클라이언트로부터 한 행의 문자열 수신
```

in.readLine() 메소드는 '\n' 문자가 올 때까지 계속 읽고 '\n'이 도착하면 그때까지 읽은 문자열을 리턴한다. 이 문자열 속에는 '\n'이 삽입되지 않는다. 현재 서버에서 라인 단위로 읽기 때문에 클라이언트에서는 송신하는 데이터의 끝에 '\n'을 덧붙여 보내야 한다.

● 클라이언트로 데이터 전송

앞서 만들어진 문자 버퍼 스트림 out을 통해 클라이언트로 데이터를 전송할 수 있다. 다음은 "Hi!, Client" 문자열을 클라이언트로 전송하는 코드이다.

```
out.write("Hi!, Client"+"\n");
out.flush();
```

"Hi!, Client"에 "\n"을 덧붙여 보내는 이유는 클라이언트 쪽에서 라인 단위('\n' 문자가 올 때까지 한 번에 읽는)로 수신한다고 가정하였기 때문이다. out.flush()를 호출하면 버퍼 스트림 속의 데이터를 모두 즉각 클라이언트로 전송한다.

out.flush()

● 데이터 송수신 종료

데이터 송수신을 모두 수행하고 소켓 연결을 끊고자 하면 다음과 같이 한다.

```
socket.close();
```

● 서버 응용프로그램 종료

더 이상 클라이언트의 접속을 받지 않고 서버 응용프로그램을 종료하고자 하는 경우 다음과 같이 ServerSocket을 종료시킨다.

```
serverSocket.close();
```

〈표 15-3〉 ServerSocket 클래스의 주 생성자

생성자	설명
ServerSocket(int port)	지정된 포트 번호(port)와 결합된 소켓 생성

〈표 15-4〉
ServerSocket 클래스의 주요 메소드

메소드	설명
Socket accept()	클라이언트로부터 연결 요청을 기다리다 요청이 들어오면 수락하고 클라이언트와 데이터를 주고받을 새 Socket 객체를 반환
void close()	서버 소켓을 닫는다.
InetAddress getInetAddress()	서버 소켓의 로컬 IP 주소 반환
int getLocalPort()	서버 소켓의 로컬 포트 번호 반환
boolean isBound()	서버 소켓이 로컬 주소와 결합되어 있으면 true 반환
boolean isClosed()	서버 소켓이 닫혀있으면 true 반환
void setSoTimeout(int timeout)	accept()가 대기하는 타임아웃 시간 지정. 0이면 무한정 대기

15.3 서버-클라이언트 채팅 프로그램 만들기

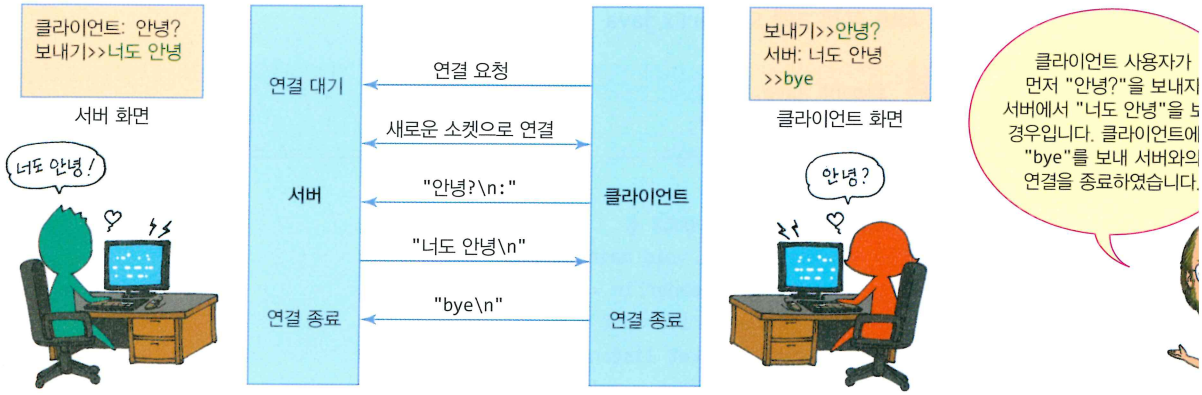
예제 개요

채팅 예제

서버와 클라이언트가 번갈아 한 번씩 채팅 문자를 보내는 간단한 채팅 예제를 만들어 보자. 채팅 프로그램의 개념은 다음과 같다.

- 서버와 클라이언트가 1:1로 채팅한다.
- 클라이언트와 서버가 서로 한번씩 번갈아 가면서 문자열 전송 및 수신한다. 클라이언트가 먼저 문자열을 보내면, 서버가 받아 출력하고 서버가 다시 문자열을 보내는 식이다.
- 서버나 클라이언트는 사용자로부터 문자열을 입력받아 보낸다. 이때 문자열 끝에 "\n"을 덧붙여 보내고 받는 쪽에서는 라인 단위로 수신한다.
- 클라이언트가 "bye"를 보내면 서버 클라이언트 모두 종료한다.

간단한 채팅 프로그램 예제를 통해 자바에서 소켓을 이용하여 어떻게 클라이언트와 서버 간에 데이터를 주고받는 지 실습해보기로 한다. 채팅 프로그램의 실행 과정은 [그림 15-7]과 같다.



[그림 15-7] 채팅 실행 과정

코드 부분 설명

채팅 프로그램의 전체 소스 코드는 다음 절에서 보이며, 여기서는 소스 코드 중 두 부분만 간단히 설명한다. 나머지는 15.2절에서 설명한 바와 거의 같다.

● 클라이언트의 소켓 생성

서버와 클라이언트를 동일한 컴퓨터에서 실행해보고자 한다. 그러므로 클라이언트에서 소켓을 생성할 때, 다음과 같이 서버의 IP 주소 부분을 "localhost"로 하면 된다. 서버가 다른 컴퓨터라면 서버의 IP 주소를 주어야 한다.

```
socket = new Socket("localhost", 9999); // 클라이언트 소켓 생성. 서버의 9999번 포트에 연결
```

● 사용자로부터 라인 입력

다음 절의 소스 코드에는 15.2절에서 설명한 내용에 클라이언트나 서버에서 사용자가 입력한 문자열을 읽어 출력하는 부분이 추가되어 있다. 다음은 서버 쪽에서 사용자가 입력한 문자열을 전송하는 부분 코드이다.

```
Scanner scanner = new Scanner(System.in);
String outputMessage = scanner.nextLine(); // 키보드로부터 한 줄을 읽는다.
out.write(outputMessage + "\n");
out.flush();
```

서버-클라이언트 채팅 소스 코드

전체 예제 코드는 다음과 같다.

• 서버 프로그램 ServerEx.java

```
1  import java.io.*;
2  import java.net.*;
3  import java.util.*;
4
5  public class ServerEx {
6      public static void main(String[] args) {
7          BufferedReader in = null;
8          BufferedWriter out = null;
9          ServerSocket listener = null;
10         Socket socket = null;
11         Scanner scanner = new Scanner(System.in); // 키보드에서 읽을 scanner 객체 생성
12         try {
13             listener = new ServerSocket(9999); // 서버 소켓 생성
14             System.out.println("연결을 기다리고 있습니다.....");
15             socket = listener.accept(); // 클라이언트로부터 연결 요청 대기
16             System.out.println("연결되었습니다.");
17             in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
18             out = new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()));
19             while (true) {
20                 String inputMessage = in.readLine(); // 클라이언트로부터 한 행 읽기
21                 if (inputMessage.equalsIgnoreCase("bye")) {
22                     System.out.println("클라이언트에서 bye로 연결을 종료하였음");
23                     break;  // "bye"를 받으면 연결 종료
24                 }
25                 System.out.println("클라이언트: " + inputMessage);
26                 System.out.print("보내기>>"); // 프롬프트
27                 String outputMessage = scanner.nextLine(); // 키보드에서 한 행 읽기
28                 out.write(outputMessage + "\n"); // 키보드에서 읽은 문자열 전송
29                 out.flush(); // out의 스트림 버퍼에 있는 모든 문자열 전송
30             }
31         } catch (IOException e) {
32             System.out.println(e.getMessage());
33         } finally {
34             try {
35                 scanner.close(); // scanner 닫기
36                 socket.close(); // 통신용 소켓 닫기
37                 listener.close(); // 서버 소켓 닫기
38             } catch (IOException e) {
39                 System.out.println("클라이언트와 채팅 중 오류가 발생했습니다.");
40             }
41         }
42     }
43 }
```

소켓 입력 스트림

소켓 출력 스트림

받은 메시지를 화면에 출력

• 클라이언트 프로그램 ClientEx.java

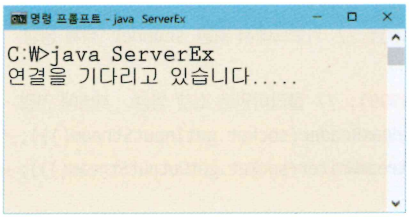
```
1  import java.io.*;
2  import java.net.*;
3  import java.util.*;
4
5  public class ClientEx {
6      public static void main(String[] args) {
7          BufferedReader in = null;
8          BufferedWriter out = null;
9          Socket socket = null;
10         Scanner scanner = new Scanner(System.in); // 키보드에서 읽을 scanner 객체 생성
11         try {
12             socket = new Socket("localhost", 9999); // 클라이언트 소켓 생성. 서버에 연결
13             in = new BufferedReader(new InputStreamReader(socket.getInputStream())); // 소켓 입력 스트림
14             out = new BufferedWriter(new OutputStreamWriter(socket.getOutputStream())); // 소켓 출력 스트림
15             while (true) {
16                 System.out.print("보내기>>"); // 프롬프트
17                 String outputMessage = scanner.nextLine(); // 키보드에서 한 행 읽기
18                 if (outputMessage.equalsIgnoreCase("bye")) {
19                     out.write(outputMessage+"\n"); // "bye" 문자열 전송
20                     out.flush();
21                     break; // 사용자가 "bye"를 입력한 경우 서버로 전송 후 실행 종료
22                 }
23                 out.write(outputMessage + "\n"); // 키보드에서 읽은 문자열 전송
24                 out.flush(); // out의 스트림 버퍼에 있는 모든 문자열 전송
25                 String inputMessage = in.readLine(); // 서버로부터 한 행 수신
26                 System.out.println("서버: " + inputMessage); // 서버로부터 받은 메시지를 화면에 출력
27             }
28         } catch (IOException e) {
29             System.out.println(e.getMessage());
30         } finally {
31             try {
32                 scanner.close();
33                 if(socket != null) socket.close(); // 클라이언트 소켓 닫기
34             } catch (IOException e) {
35                 System.out.println("서버와 채팅 중 오류가 발생했습니다.");
36             }
37         }
38     }
39 }
```

서버-클라이언트 채팅 예제 실행

예제를 실행하기 전에 컴파일된 클래스 파일 ServerEx.class와 ClientEx.class를 드라이브 c:\ 밑에 옮겨 놓아야 한다. 서버와 클라이언트는 따로 명령창을 열어 실행한다. 실행은 다음 순서로 진행한다.

연결 요청을 기다린다

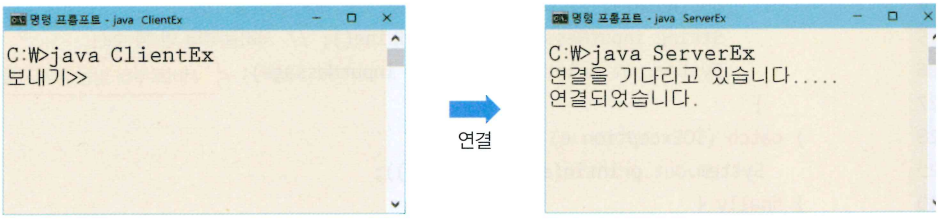
1. 명령창을 열고 서버 프로그램을 먼저 실행시켜 클라이언트의 연결 요청을 기다린다. [그림 15-8]과 같다.



[그림 15-8] 서버가 먼저 실행되어 클라이언트로부터 연결을 기다리고 있는 서버 명령창

서버와 연결

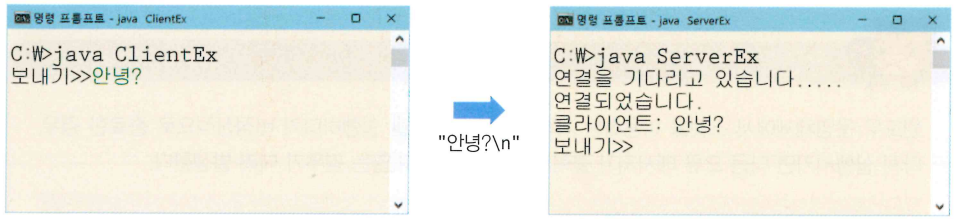
2. [그림 15-9](a)와 같이 다른 명령창을 열어 클라이언트 프로그램을 실행한다. 클라이언트 프로그램이 실행되면 바로 서버로 연결을 시도한다. 서버와 연결이 이루어지면 [그림 15-19](b)와 같이 서버의 명령창에는 "연결되었습니다."가 출력된다.



(a) 클라이언트 명령창 (b) 서버의 명령창

[그림 15-9] 클라이언트에서 서버로 문자열 보내기

3. 채팅은 클라이언트에서 먼저 시작하며, 번갈아 메시지를 주고받는다. [그림 15-10](a)에서 클라이언트는 사용자로부터 "안녕?"과 <Enter> 키가 입력되면 서버로 전송한다. [그림 15-10](b)에서 서버는 받은 문자열을 출력한다.

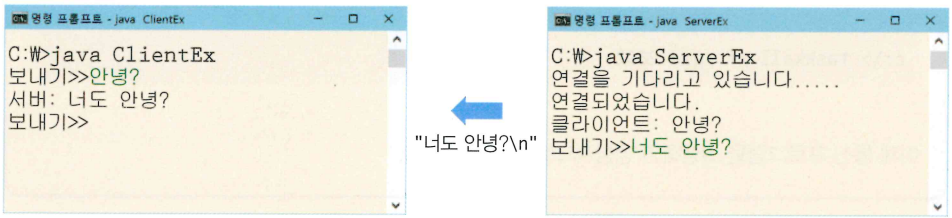


(a) 클라이언트는 사용자로부터 "안녕?"을 입력받아 전송

(b) 서버는 클라이언트로부터 받은 문자열 출력

[그림 15-10] 클라이언트에서 서버로 문자열 보내기

4. 이제, 서버는 사용자로부터 "너도 안녕?"을 입력받아 클라이언트로 보낸다. [그림 15-11](b)에서 서버는 사용자로부터 입력받은 "너도 안녕?"을 보낸다. [그림 15-11](a)에는 클라이언트가 서버로부터 받은 문자열을 출력한다. 이런 식으로 번갈아 메시지를 주고받는다.



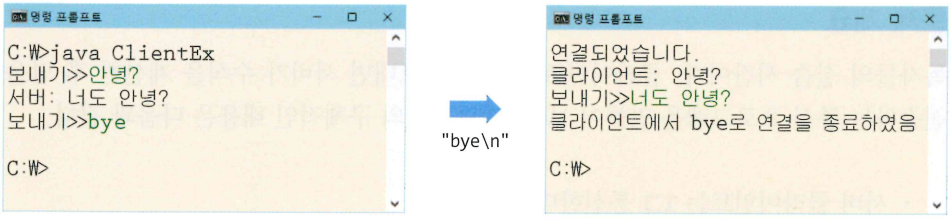
(a) 클라이언트는 서버로부터 받은 문자열 출력

(b) 서버는 사용자로부터 입력받은 "너도 안녕?"을 전송

[그림 15-11] 서버에서 클라이언트로 문자열 보내기

5. 클라이언트에서 "bye"를 서버에게 전송하면 채팅이 종료된다. [그림 15-12](a)는 클라이언트에서 "bye"를 입력받아 서버로 보내고 스스로 종료한 화면이고, [그림 15-12](b)는 "bye" 문자열을 받고 서버가 종료하는 화면이다.

채팅 종료



(a) 클라이언트가 "bye"를 서버로 전송하고 스스로 종료함

(b) 서버는 "bye"를 받고 종료함

[그림 15-12] 클라이언트가 "bye"를 전송하면 채팅 종료



잠깐!

윈도우 운영체제에서 소켓을 이용하는 자바 응용프로그램을 실행하다가 비정상적으로 종료한 경우, 다시 실행시키면 다음 오류 메시지가 출력되고 실행이 되지 않는 경우가 더러 발생한다.

```
java.net.BindException: Address already in use: JVM_Bind
```

이것은 소켓이 포트를 바인드(소유)하여 사용하다가 바인드를 풀어 놓지 않고 (비정상)종료된 상황에서, 그 포트를 바인드할 때 나타나는 오류이다. 이런 경우, 명령창에 다음 명령을 입력하면 현재 윈도우에서 사용되고 있는 TCP 포트와 포트를 소유하고 있는 프로세스의 번호(PID) 리스트를 출력한다.

```
c:\> netstat -a -n -o
```

명령창에 출력된 리스트에서 문제가 된 포트와 이를 소유하고 있는 프로세스 번호(PID)를 찾아 프로세스를 죽이면 된다. 만일 프로세스 번호가 2692라고 하면 명령창에 다음 명령을 입력한다.

```
c:\> taskkill /f /pid 2692
```

이제 통신 프로그램을 실행하면 정상적으로 실행될 것이다.

15.4 수식 계산 서버-클라이언트 만들기 실습

문제 개요

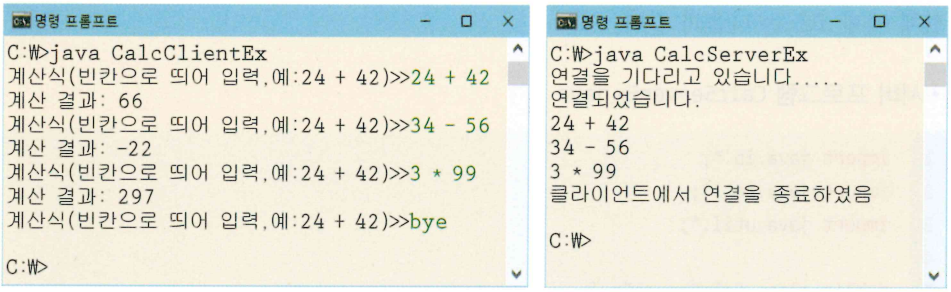
독자들의 실습 시간이다. 클라이언트가 수식을 보내면 서버가 수식을 계산한 후 답을 전송하는 통신 프로그램을 만들어 보자. 프로그램의 구체적인 내용은 다음과 같다.

- 서버 클라이언트는 1:1 통신한다.
- 서버를 먼저 실행시키고 클라이언트를 실행시켜 서버에 접속한다.
- 클라이언트는 사용자로부터 수식을 입력받아 서버로 전송한다. 연산자는 +, -, *의 3가지만 허용하고 정수 연산만 가능하다.

- 서버가 식을 받으면 식을 서버의 화면에 출력하고, 계산하여 결과를 클라이언트로 전송한다.
- 클라이언트는 서버로부터 받은 답을 화면에 출력한다.
- 클라이언트와 서버는 전송할 데이터를 문자열로 만들고 "\n"을 덧붙여 전송하며, 받는 쪽에서는 라인 단위로 수신한다.
- 클라이언트가 "bye"를 보내면 양쪽 모두 종료한다.

실행 예시

[그림 15-13]은 수식 계산 통신 프로그램의 실행 과정 예시를 보여준다. 실행 전에 CalcClientEx.class와 CalcServerEx.class 파일은 미리 c:\에 옮겨져 있어야 한다. 서버가 먼저 실행되어 기다리며 클라이언트가 실행되면 연결이 이루어진다. 클라이언트가 사용자로부터 24 + 32의 수식을 입력받아 서버로 전송하고, 서버는 식을 계산하여 결과를 전송한다. 클라이언트는 결과 값을 받아 출력하고 다시 사용자로부터 수식을 입력받는다. 이 과정이 반복된다. 24 + 32에서 숫자와 + 기호 사이에는 반드시 빈칸으로 띈다.



(a) 계산 클라이언트의 실행 (b) 계산 서버의 실행

[그림 15-13] 수식 계산 서버-클라이언트의 실행 과정

코드 부분 설명

이 실습 예제는 앞의 채팅 프로그램과 거의 유사하여 작성하는데는 큰 어려움이 없을 것으로 본다. 다만 서버 쪽에서는 전달받은 수식을 계산하는 calc() 메소드가 추가 작성되었는데 이 메소드에 대해서만 알아보자. 클라이언트부터 넘어오는 수식은 다음과 같은 모양의 문자열이다.

```
"24 + 32" 또는 "36 - 5", "5 * 6"
```

숫자와 연산기호 사이에 빈칸(' ')을 삽입하여 전송하도록 제한을 가한 이유는 StringTokenizer 클래스를 이용하여 숫자와 연산기호를 분리하기 쉽게 함이다(예제가 쉽도록). 예를 들어 exp에 "24 + 32"의 문자열이 들어 있다면, 다음 코드를 이용하여 빈칸(' ')을 기준으로 exp의 문자열을 분리하면,

```
StringTokenizer st = new StringTokenizer(exp, " ");
```

다음과 같이 3개의 토큰으로 분리된다.

```
토컨 0 : "24"
토컨 1 : "+"
토컨 2 : "32"
```

calc() 메소드는 토큰 1의 연산기호(+)에 따라 계산을 수행하고 결과를 문자열로 만들어 리턴한다.

수식 계산 서버-클라이언트 소스 코드

전체 예제 코드는 다음과 같다.

• 서버 프로그램 CalcServerEx.java

```
1  import java.io.*;
2  import java.net.*;
3  import java.util.*;
4
5  public class CalcServerEx {
6      public static String calc(String exp) {
7          StringTokenizer st = new StringTokenizer(exp, " ");
8          if (st.countTokens() != 3) return "error";
9
10         String res="";
11         int op1 = Integer.parseInt(st.nextToken());
12         String opcode = st.nextToken();
13         int op2 = Integer.parseInt(st.nextToken());
14         switch (opcode) {
15             case "+": res = Integer.toString(op1 + op2);
16                 break;
17             case "-": res = Integer.toString(op1 - op2);
18                 break;
19             case "*": res = Integer.toString(op1 * op2);
```

```
20                  break;
21              default : res = "error";
22          }
23          return res;
24      }
25
26      public static void main(String[] args) {
27          BufferedReader in = null;
28          BufferedWriter out = null;
29          ServerSocket listener = null;
30          Socket socket = null;
31          try {
32              listener = new ServerSocket(9999); // 서버 소켓 생성
33              System.out.println("연결을 기다리고 있습니다.....");
34              socket = listener.accept(); // 클라이언트로부터 연결 요청 대기
35              System.out.println("연결되었습니다.");
36              in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
                                                                    // 소켓 입력스트림
37              out = new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()));
                                                                    // 소켓 출력 스트림
38              while (true) {
39                  String inputMessage = in.readLine(); // 클라이언트로부터 한 행 읽기. 수식
40                  if (inputMessage.equalsIgnoreCase("bye")) {
41                      System.out.println("클라이언트에서 연결을 종료하였음");
                                                        // 받은 메시지를 화면에 출력
42                      break; // "bye"를 받으면 연결 종료
43                  }
44                  System.out.println(inputMessage); // 받은 메시지를 화면에 출력
45                  String res = calc(inputMessage); // 계산. 계산 결과는 res
46                  out.write(res + "\n"); // 계산 결과 문자열 전송
47                  out.flush();
48              }
49          } catch (IOException e) {
50              System.out.println(e.getMessage());
51          } finally {
52              try {
53                  if(socket != null) socket.close(); // 통신용 소켓 닫기
54                  if(listener != null) listener.close(); // 서버 소켓 닫기
55              } catch (IOException e) {
56                  System.out.println("클라이언트와 채팅 중 오류가 발생했습니다.");
57              }
58          }
59      }
60  }
```

• 클라이언트 프로그램 CalcClientEx.java

```
1  import java.io.*;
2  import java.net.*;
3  import java.util.*;
4
5  public class CalcClientEx {
6      public static void main(String[] args) {
7          BufferedReader in = null;
8          BufferedWriter out = null;
9          Socket socket = null;
10         Scanner scanner = new Scanner(System.in); // 키보드에서 읽을 scanner 객체 생성
11         try {
12             socket = new Socket("localhost", 9999);
                                                    // 클라이언트 소켓 생성. 서버와 바로 연결
13             in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
                                                                    // 소켓 입력 스트림
14             out = new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()));
                                                                    // 소켓 출력 스트림
15             while (true) {
16                 System.out.print("계산식(빈칸으로 띄어 입력,예:24 + 42)>>"); // 프롬프트
17                 String outputMessage = scanner.nextLine(); // 키보드에서 수식 읽기
18                 if (outputMessage.equalsIgnoreCase("bye")) {
19                     out.write(outputMessage+"\n"); // "bye" 문자열 전송
20                     out.flush();
21                     break; // 사용자가 "bye"를 입력한 경우 서버로 전송 후 연결 종료
22                 }
23                 out.write(outputMessage + "\n"); // 키보드에서 읽은 수식 문자열 전송
24                 out.flush();
25                 String inputMessage = in.readLine(); // 서버로부터 계산 결과 수신
26                 System.out.println("계산 결과: " + inputMessage);
                                                    // 서버로부터 받은 계산 결과 화면 출력
27             }
28         } catch (IOException e) {
29             System.out.println(e.getMessage());
30         } finally {
31             try {
32                 scanner.close();
33                 if(socket != null) socket.close(); // 클라이언트 소켓 닫기
34             } catch (IOException e) {
35                 System.out.println("서버와 채팅 중 오류가 발생했습니다.");
36             }
37         }
38     }
39 }
```

요약



SUMMARY

- TCP는 다른 두 시스템 간에 신뢰성 있는 데이터의 전송을 관장하는 프로토콜이며 IP는 패킷 교환 네트워크에서 송신 호스트와 수신 호스트가 데이터를 주고받는 것을 관장하는 프로토콜이다.
- IP 주소는 231.1.127과 같이 네트워크상에서 유일하게 식별될 수 있는 4개의 숫자로 구성된다.
- TCP 포트(port)는 프로그램 간에 TCP 통신을 위한 가상의 연결 단으로서 통신할 프로그램을 식별한다.
- 소켓은 네트워크상에서 통신하는 두 프로그램 간의 양방향 통신 링크의 한쪽 끝단을 의미하며 포트에 연결된다.
- 소켓 통신은 통신 프로그램의 기반 기술로서, 소켓 통신 라이브러리를 이용하여 쉽게 통신 프로그램을 개발할 수 있다.
- 소켓 통신은 서버와 클라이언트 모형으로 통신하는 기반 기술이므로, 어느 하나는 서버가 되고 나머지 하나는 클라이언트가 된다.
- 자바 소켓은 2종류로서 서버 소켓인 ServerSocket 클래스와 클라이언트 소켓인 Socket 클래스가 있다.
- 소켓 통신에서 클라이언트로 Socket 객체를 생성하고, Socket 객체에게 서버의 IP 주소(혹은 도메인 주소)와 포트 번호를 주어 서버에 접속하도록 한다.
- 소켓 통신에서 서버는 서버 소켓인 ServerSocket 객체를 생성하고 특정 포트와 결합시키고, accept() 메소드를 호출하여 클라이언트의 연결 요청을 기다리도록 한다.
- ServerSocket 객체는 클라이언트의 요청이 들어오면 연결을 완료하고, 데이터 통신을 이어갈 Socket 객체를 생성하여 반환한다. 그리고 다시 다른 클라이언트의 연결을 기다린다. 클라이언트와의 데이터 통신은 모두 이 Socket 객체에 의해 이루어진다. 그러므로 ServerSocket은 연결을 기다리는(listen) 역할만 하고, 연결된 후의 데이터의 통신은 Socket 클래스의 역할이다.
- Socket 클래스는 응용프로그램이 네트워크에 데이터를 전송하거나 수신을 쉽게 하도록, 입출력 스트림을 제공한다. 응용프로그램은 Socket 객체로부터 스트림을 알아내고 이 스트림에 입출력함으로서 네트워크로 데이터를 송수신할 수 있다.
- 연결을 종료할 때는 Socket 클래스의 close() 메소드를 호출한다.
- 서버 소켓을 닫으면 클라이언트는 더 이상의 연결 요청을 할 수 없다.

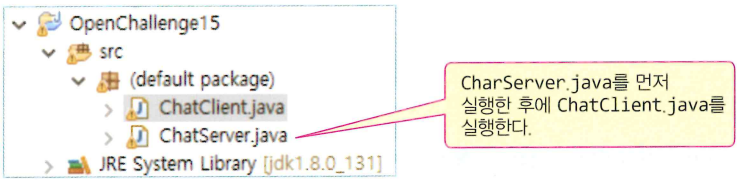


스레드를 이용한 서버-클라이언트 채팅 Open Challenge

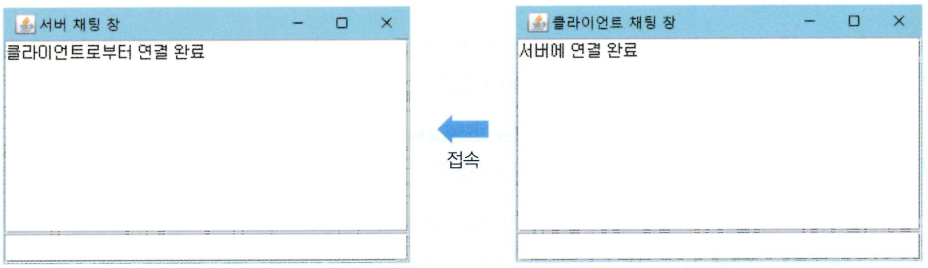
목적

소켓 통신, 스레드를 이용하여 양방향 통신 작성

본문에서 다루었던 서버-클라이언트 1:1 채팅 예제는 클라이언트에서부터 시작하여 클라이언트와 서버가 순서대로 번갈아 가면서 메시지를 주고받았다. 이 예제를 순서에 상관없이 자유롭게 서버와 클라이언트가 메시지를 주고받을 수 있도록 스레드를 이용하여 스윙 프로그램으로 재작성하라. 서버와 클라이언트는 JTextArea를 이용하여 상대방으로부터 받은 메시지를 출력하고, JTextField 창을 이용하여 상대방에게 보낼 메시지를 사용자로부터 입력받고 <Enter>키를 입력하면 상대에게 바로 전송하도록 하라. 본문의 예제와 달리 어느 한쪽이 접속을 끊으면 프로그램이 종료되도록 한다. 난이도 7

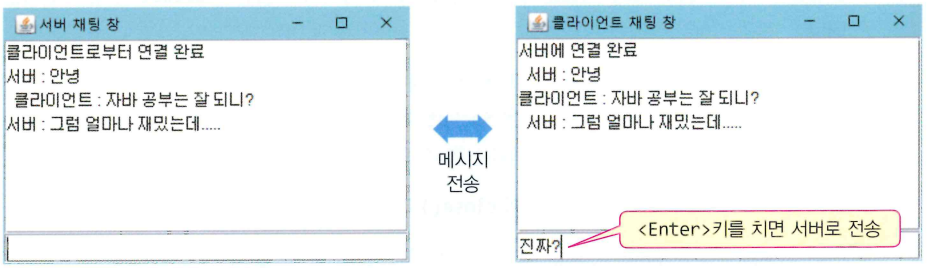


프로젝트 구성



서버를 먼저 실행하고, 클라이언트 접속 기다림

클라이언트가 실행되면 바로 서버에 접속함



서버, 클라이언트가 순서에 상관없이 메시지를 주고 받는 화면

JTextField 창에 서버로 보낼 메시지를 입력하는 화면

연습문제

EXERCISE



이론문제

• 홀수 문제는 정답이 공개됩니다.

1. 윈도우 PC의 명령창에서 내 PC의 IP 주소를 확인하는 명령은 무엇인가? 독자들인 명령창에서 그 명령을 입력하여 독자 PC의 IP 주소를 확인해보라.

2. 잘 알려진 포트(well known port)에 대해 잘못 설명한 것은?
   ① 기존의 많이 알려진 통신프로그램들이 전용으로 사용하는 포트를 일컫는다.
   ② 잘 알려진 포트의 번호는 주로 0~1023 사이에 있다.
   ③ Telnet 응용프로그램은 23번 포트를 사용한다.
   ④ 개발자는 응용프로그램에서 잘 알려진 포트를 사용해야 자신의 프로그램을 알리는 데 도움이 된다.

3. 소켓 통신에 대해 잘못 설명한 것은?
   ① 통신 소프트웨어는 서버 클라이언트로 나누어져 개발되어야 한다.
   ② 소켓 주소는 IP 주소와 포트 주소로 구성된다.
   ③ 소켓은 서버 소켓과 클라이언트 소켓의 두 종류가 있다.
   ④ 서버 소켓은 서버 쪽에서 데이터를 전송하는데 사용되고 클라이언트 소켓은 클라이언트 쪽에서 데이터를 전송하는데 사용된다.

4. 통신 프로그램이 상대 통신 프로그램에 접속할 때 필요한 것은?
   ① IP 주소
   ② IP 주소와 포트 주소
   ③ 호스트 이름
   ④ 프로토콜과 호스트 이름

5. 다음 코드에 대한 설명으로 틀린 것은?

```
Socket socket = new Socket("192.168.0.115", 5050);
```

   ① 클라이언트 소켓 객체를 생성한다.
   ② 클라이언트가 접속할 서버의 주소는 192.168.0.115이다.
   ③ 생성된 클라이언트 소켓의 포트 번호는 5050이다.
   ④ 소켓 객체가 생성될 때 서버에 바로 접속한다.

6. 다음은 통신 서버 코드의 일부분이다. 주석에 따라 빈칸에 코드를 작성하라.

```
ServerSocket ss = _____ // 5550포트와 결합하는 서버 소켓 생성
Socket  s = _____ // 클라이언트로부터의 접속을 기다린다.
BufferedWriter out = new BufferedWriter(new
    OutputStreamWriter(s._____); // 소켓 s로 데이터를 전송할 출력 스트림을 만든다.
out._____ // 소켓으로 하여금 "안녕"을 전송하도록 한다.
s._____ // 소켓 s를 닫는다.
_____ // 서버 소켓을 닫는다.
```

7. ServerSocket에 대한 다음 설명으로 틀린 것은?

```
ServerSocket ss = new ServerSocket(5050);
```

① 서버 컴퓨터에서 서버 소켓을 생성하는 코드이다.
② 서버 소켓이 클라이언트로부터 접속을 받는 포트 번호는 5050이다.
③ 생성된 ss 소켓은 서버에서 클라이언트와 데이터를 주고받기 위해 사용된다.
④ ServerSocket은 Socket 클래스에는 없는 accept() 메소드를 가지고 있다.

8. 다음은 통신 클라이언트 코드의 일부분이다. 주석에 따라 빈칸에 코드를 작성하라.

```
Socket s = _____ // 203.1.1.1주소의 6000번 포트에 접속을 시도한다.
BufferedReader in = new BufferedReader(new
    InputStreamReader(s._____); // 소켓 s로부터 데이터를 수신할 입력 스트림을 만든다.
in._____ // 소켓으로부터 한 라인의 텍스트를 입력받는다.
s._____ // 소켓 s를 닫는다.
```

실습문제

• 홀수 문제는 정답이 공개됩니다.

1. 클라이언트에서는 한 줄씩 텍스트를 입력 받아 서버로 보내고, 서버는 받은 텍스트를 출력하는 소켓 프로그램을 작성하라. 클라이언트가 "끝"을 전송하면 클라이언트와 서버 모두 연결을 끊고 종료하라. 실행 예시는 다음과 같다. 난이도 6

목적 클라이언트에서는 서버에는 받기만 하는 1: 만들기

```
서버입니다. 클라이언트를 기다립니다...
연결되었습니다.
... 자바 프로그램을 짜고 있어.
... 굉장히 재밋어.
... 소켓 프로그램이 더 그래
... 다했다. 이제 자야겠어
접속을 종료합니다.
```

서버 화면

접속, 텍스트 전송

```
서버에 접속하였습니다...
텍스트 입력 >> 자바 프로그램을 짜고 있어.
텍스트 입력 >> 굉장히 재밋어.
텍스트 입력 >> 소켓 프로그램이 더 그래
텍스트 입력 >> 다했다. 이제 자야겠어
텍스트 입력 >> 끝
연결을 종료합니다.
```

클라이언트 화면

2. 서버는 클라이언트가 접속하자마자 500밀리초마다 클라이언트에게 1부터 시작하여 1씩 증가하는 숫자를 전송하여 시간을 알려주는 소켓 통신 프로그램을 작성하라. 서버는 10초가 지나면 연결을 끊고 종료한다. 난이도 6

목적 서버는 보내기만 하. 라이언트는 받기만 하는 1 신 만들기

```
타임서버입니다.
연결되었습니다.
500ms마다 수를 보냅니다.
종료합니다.
```

서버 화면

접속

숫자 전송

```
서버에 접속하였습니다...
서버의 시간 : 0 1 2 3 4 5 6 7 8 9 10
11 12 13 14 15 16 17 18 19
연결을 종료합니다.
```

클라이언트 화면

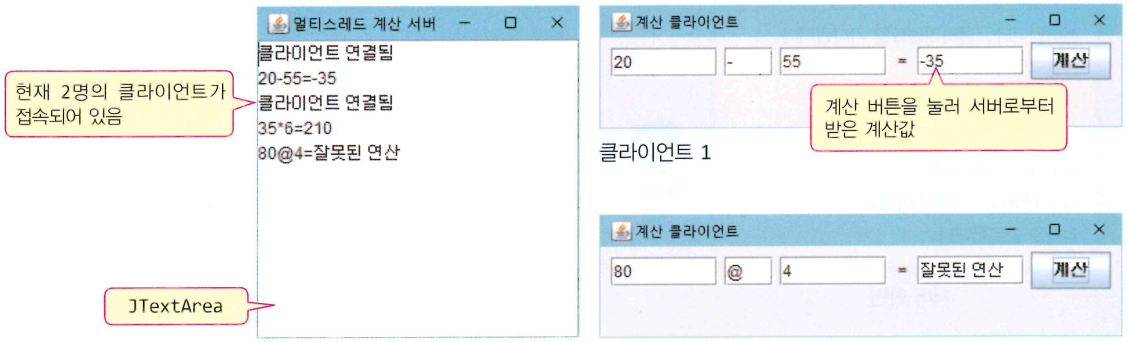
힌트 Hint

서버에서는 500ms마다 수를 보내는 TimerThread를 만들고 main()에서는 다음과 같이 스레드 객체를 생성한다. TimerThread는 run()에서 20번 수를 보낸 후 종료한다.

```
out = new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()));
TimerThread th = new TimerThread(out); // 스레드 생성자에 스트림 out 전달
th.start();
try {
    th.join(); // 스레드 th가 종료하기를 기다린다.
} catch (InterruptedException e) { }
```

목적 다중 클라이언트를 서비스하는 소켓 통신 응용

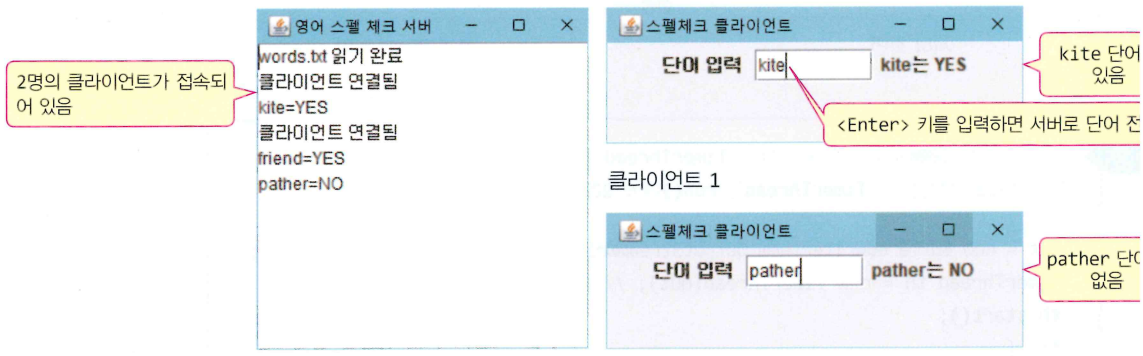
3. 하나의 서버에 다중 클라이언트가 동시에 접속하여 통신하는 프로그램을 작성해보자. 본문의 15.4의 코드를 스윙을 이용하여 실행 예시와 같이 수정 작성하라. 계산 서버는 JTextArea 창을 두고 클라이언트가 접속했음과 클라이언트가 보내온 수식, 그리고 계산 결과를 출력하여 서버 사용자에게 알리고, 클라이언트는 계산 버튼을 누르면 계산식을 서버로 전송하고 결과를 받아 JTextField 결과 창에 출력한다. 난이도 7



서버 클라이언트 1 클라이언트 2

목적 다중 클라이언트를 서비스하는 소켓 통신 응용

4. 하나의 서버에 다중 클라이언트가 동시에 접속하여 통신하는 프로그램을 작성해보자. 서버는 영어 단어 25143개를 가지고 있는 words.txt 파일을 읽고, 클라이언트는 사용자로부터 영어 단어를 입력받아 서버로 보낸다. 서버는 클라이언트로부터 받은 단어가 words.txt에 있는지 검사하고 있으면 "YES", 없으면 "NO"를 전송한다. 클라이언트는 서버로부터 받은 결과를 출력한다. words.txt 파일은 독자들에게 배포하는 자료에 포함되어 있다. 난이도 7



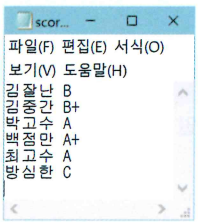
서버 클라이언트 1 클라이언트 2

힌트 Hint

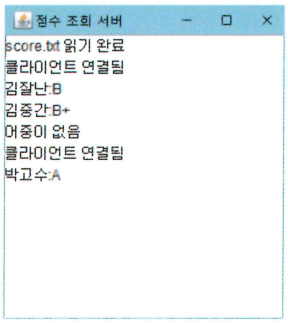
words.txt 파일을 읽어 Vector<String> 컬렉션에 저장하라. 8장 open challenge 참고.

목적 서버의 파일을 검색
다중 클라이언트 서비스

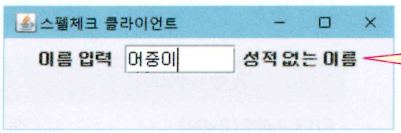
5. 학생 이름과 학점이 저장된 텍스트 파일을 미리 c:\temp\score.txt에 마련해두고, 서버는 이 파일을 읽는다. 그리고 클라이언트로부터 학생 이름을 입력받으면 학점을 전송하는 통신 프로그램을 작성하라. 여러 클라이언트가 서버에 접속할 수 있도록 하라. 난이도 8



score.txt 파일

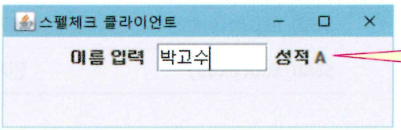


서버



서버로부터 검색한 결과

클라이언트 1



서버로부터 검색한 결과

클라이언트 2

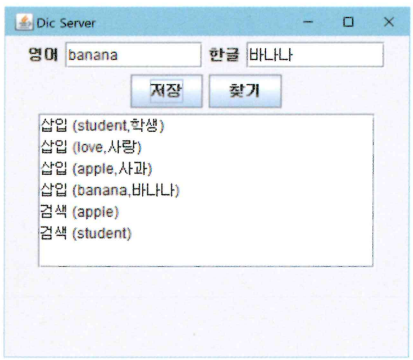
힌트 Hint

score.txt 파일을 읽어 HashMap<String, String> 컬렉션에 저장하라.

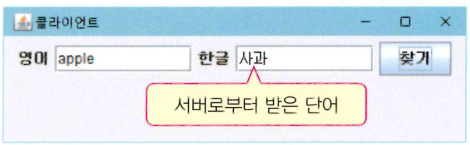
목적 소켓 통신, HashMap
복합적인 응용프로그램 작성

6. 서버의 관리자는 서버 프로그램의 화면에서 영한사전의 데이터를 저장할 수 있고, 클라이언트는 서버에 영어 단어를 보내 한글 단어를 검색할 수 있다. 다음은 실행 화면으로 서버에서 (student, 학생), (love, 사랑), (apple, 사과), (banana, 바나나)의 4 단어를 입력하였다. 그리고 클라이언트 프로그램을 실행하여 love 단어와 apple을 검색하는 화면이다. 사용자가 클라이언트에서 "apple"를 입력하고 "찾기" 버튼을 누르면 클라이언트는 서버에 "apple"을 보낸다. 서버는 현재 입력된 단어 중에서 "apple"을 찾고 "사과"를 전송한다. 클라이언트는 받은 "사과"를 출력한다.

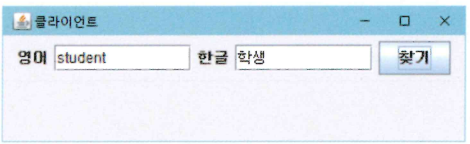
난이도 8



서버



클라이언트 1



클라이언트 2

목적 소켓 통신으로 파일 전송 응용 만들기 연습. 바이너리 데이터 전송

7. GUI 없이 콘솔에서 실행되는 파일 전송 프로그램을 작성해보자. 클라이언트에서 서버로 바이너리 파일을 전송하여 서버에 똑같은 사본을 저장하는 프로그램을 작성하라. 클라이언트와 서버는 다음과 같은 간단한 프로토콜을 이용하라. 편의상 클라이언트는 서버에게 명령과 인자를 오류 없이 전송하는 것으로 간주하고 오류 복구를 위한 재전송은 없는 것으로 한다. 난이도 8

명령(1바이트)	명령 인자
FILE_NAME(0x00)	첫 4바이트(파일 이름 바이트 크기), 파일 이름
FILE_SIZE(0x01)	파일 크기(8바이트)
SEND_BEGIN(0x02)	인자 없음
SEND_END(0x03)	인자 없음

```
연결됨
전송받은 파일 이름 길이 :8
전송받은 파일 이름:back.jpg
저장할 파일 이름:copy_back.jpg
전송받은 파일 크기:561276
.....
.....
.....
.....
파일 수신 성공. 현재 폴더에 저장되었습니다.
```

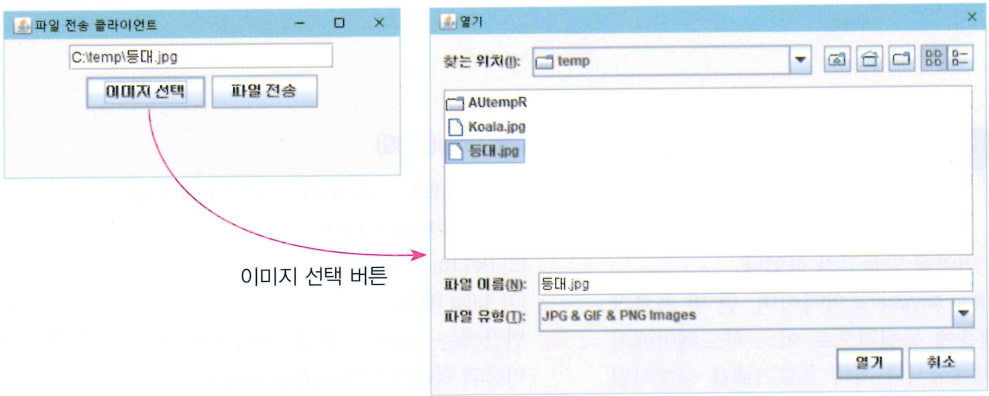
서버

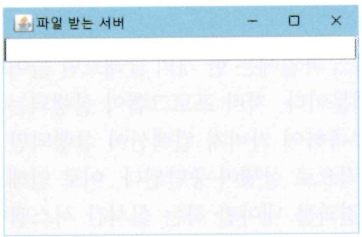
```
보낼 파일 이름을 입력하세요>>c:\temp\back.jpg
파일 전송이 완료되었습니다.
```

클라이언트

목적 스윙으로 바이너리 ... 전송 프로그램 작성. 종합 ...

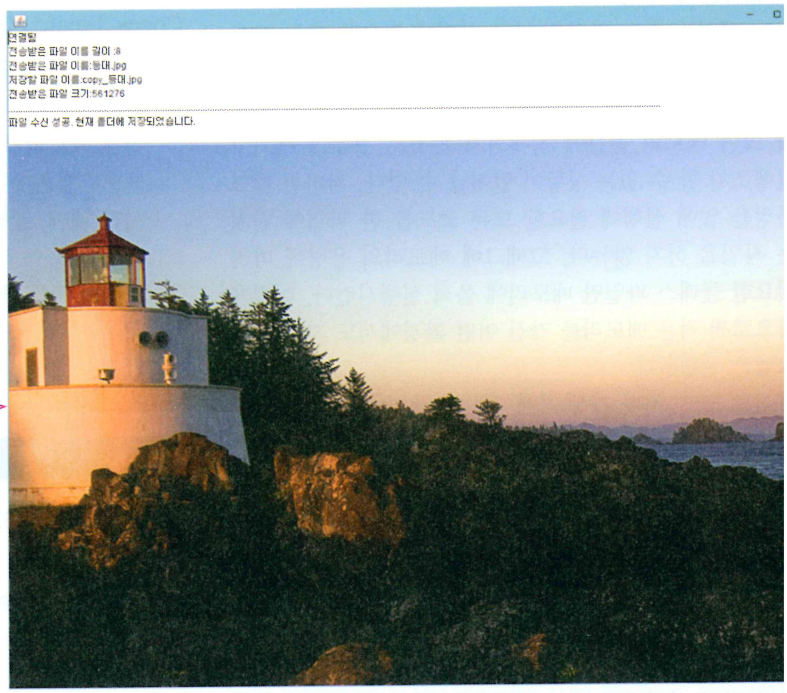
8. 문제 7의 프로그램을 스윙을 이용하여 재작성하라. 서버를 먼저 실행시키고, 클라이언트에서 '이미지 선택' 버튼을 누르면 JFileChooser를 이용하여 사용자로부터 이미지 파일을 선택받는다. 그리고 '파일 전송' 버튼을 누르면 이미지 파일을 서버로 전송한다. 서버는 받은 이미지를 화면에 출력한다. 난이도 9





초기 서버 화면

클라이언트로부터 이미지 파일을 받으면 서버에 출력된다.



이미지를 받은 후 서버 화면

Check Time 정답

1장 Check Time 정답

Check Time(p.29)

1. ④ 효율적인 절차적 언어를 만들고자 하였다.
2. WORA는 Write Once Run Anywhere 약자이며, 한 번 프로그램을 작성하면 플랫폼에 독립적으로 어느 하드웨어에서나 또는 어느 운영체제에서나 자바 프로그램을 실행시킬 수 있다는 뜻이다.
3. C/C++에서는 링크 과정을 거쳐 실행에 필요한 모든 코드를 합쳐 하나의 exe(실행 파일)을 만든다. 그리고 이 실행 파일을 메모리에 적재하여 실행한다. 만일 실행 파일이 크면 메모리 공간에 적재되지 못하는 경우가 생기고 실행조차 할 수 없는 상황이 일어날 수 있다. 자바는 링크 과정을 없애 실행에 필요한 모든 코드를 한 파일에 합치는 작업을 하지 않는다. 그때그때 메모리의 용량에 따라 필요한 클래스 파일만 메모리에 올려 실행시킨다. 이렇게 함으로써 적은 메모리를 가진 어떤 환경에서도 자바 프로그램은 실행된다.
4. 바이트 코드, 자바 가상 기계

Check Time(p.37)

1. JRE이다. JRE에는 자바 실행 환경만 포함되어 있다.
2. 자바 핵심 API를 구성하는 클래스들에 대한 자바 소스가 포함되어 있다.
3. JDK 설치 디렉터리 밑의 bin 디렉터리의 javac.exe가 자바 컴파일러이다.
4. 자바 명령 실행기의 파일 이름은 java.exe이며 JDK 설치 디렉터리의 bin 디렉터리나 또는 JRE 설치 디렉터리 밑의 bin 디렉터리에 있다.

Check Time(p.39)

1. public static void main(String[] args) 메소드
2. (1) javac Hello.java
   (2) Hello.class
   (3) java Hello
3. 빈칸에는 class가 필요하며, 자바 파일의 이름은 클래스 이름과 동일하게 JavaApp.java로 저장

Check Time(p.50)

1. 하나의 자바 클래스 파일에는 한 개의 클래스만 들어간다
2. 가비지 컬렉션 때문이다. 자바 프로그램이 실행되는 동안 가용 메모리가 부족하여 가비지 컬렉션이 실행되면 자바 프로그램은 일시적으로 실행이 중단된다. 이로 인해 일정 시간 내에 실행 결과를 내야만 하는 실시간 시스템에 부적합하다.
3. 자바의 모든 변수나 메소드는 반드시 클래스 내에 선언하게 되어 있으므로 클래스에 속하지 않은 변수와 메소드는 있을 수 없다. 이것은 캡슐화의 특징이다.

2장 Check Time 정답

Check Time(p.62)

1. ⑥
2. public 지정이 빠져 있다. main() 메소드는 반드시 public static void로 선언되어야 한다.
3.

```
public class Hi {
   public static void main(String[] args) {
      System.out.println("Hi");
   }
}
```

Check Time(p.65)

1. ② int 1stVar; // 숫자로 시작할 수 없기 때문
2. ③ MyDocument
3. final int ALPHA = 30;

Check Time(p.75)

1. ①
2.

```
final int YEAR = 365; // 또는 static final int YEAR = 365;
final int MONTH = 30; // 또는 static final int MONTH = 30;
final int WEEK = 7; // 또는 static final int WEEK = 7;
```

3. 256. byte 타입은 -128~127까지 가능
4. "a"
5. 22

Check Time(p.80)

1. import java.util.Scanner;
2.

```
Scanner scanner = new Scanner(System.in);
int i = scanner.nextInt();
int j = scanner.nextInt();
System.out.println(i+j);
```

Check Time(p.93)

1. 11
2. 4
   -4
3. 4

Check Time(p.104)

1. if (i=0)이 잘못됨
   i=0이 i==0으로 바뀌어야 한다.
2. 홀수. 소스는 다음과 같이 수정되어야 한다.

```
int i = 4;
if (i % 2 == 0) {
   if (i % 3 == 0)
      System.out.println("6의 배수");
}
else
   System.out.println("홀수");
```

3. 출력은 다음과 같다.
   옵션 1
   옵션 2
   옵션 3
4. ④. 실수는 case 문의 값이 될 수 없다.

3장 Check Time 정답

Check Time(p.127)

1.

```
public class ForNestedLoop {
   public static void main(String[] args) {
      int i;
      for (i=5;i>0;i--) {
         for (int j=0; j<i; j++)
            System.out.print("*");
         System.out.println();
      }
   }
}

public class WhileNestedLoop {
   public static void main(String[] args) {
      int i;
      i=5;
      while (i>0) {
         int j = 0;
         while (j++<i)
            System.out.print("*");
         System.out.println();
         i--;
      }
   }
}

public class DoWhileNestedLoop {
   public static void main(String[] args) {
      int i;
      i=5;
```

```
        do {
            int j = 0;
            while (j++<i)
                System.out.print("*");
            System.out.println();
            i--;
        } while(i>0);
    }
}
```

2. 무한 반복에 빠진다.
3.

```
public class Sum1to100 {
    public static void main(String[] args) {
        int sum = 0;
        for (int i=1; i<=100; i++)
            sum = sum + i;
        System.out.print(sum);
    }
}
```

Check Time(p.137)

1. `int tenArray[] = new int[10];`
   또는
   `int[] tenArray = new int[10];`
2. `tenArray.length`
3.

```
public class ArrayCheck {
    public static void main(String[] args) {
        int tenArray[] = new int[10];

        for (int i=0; i<tenArray.length; i++)
            tenArray[i] = i+1;

        int sum = 0;

        for (int i=0; i<tenArray.length; i++)
            sum = sum + tenArray[i];

        System.out.print(sum);
    }
}
```

Check Time(p.145)

1. ③
2. `int i[][] = new int[2][3];`
   또는
   `int[][] i = new int[2][3];`
3.

```
char [] makeCharArray() { // 리턴 타입 작성
    char [] c = new char[4];
    return c; // return 문 작성
}
```

4. `int n[][] = {{0,1,2,3},{4},{5},{6,7,8,9}};`
   또는

```
public class SkewedArray {
    public static void main(String[] args) {
        int k=0;

        // 비정방형 배열 만들기
        int n[][] = new int [4][];
        n[0] = new int [4];
        n[1] = new int [1];
        n[2] = new int [1];
        n[3] = new int [4];

        for(int i=0; i<n[0].length; i++) {
            n[0][i] = k++;
        }
        n[1][0] = k++;
        n[2][0] = k++;
        for(int i=0; i<n[3].length; i++) {
            n[3][i] = k++;
        }

        // 값 확인하기
        for(int i=0; i<n.length; i++) {
            for(int j=0; j<n[i].length; j++)
                System.out.print(n[i][j]+" ");
            System.out.println();
```

```
    }
  }
}
```

Check Time(p.149)

1. 자바에서 main()의 매개변수 타입은 문자열 배열(String []])이다.
2. 명령행 인자의 개수는 main() 메소드의 문자열 배열 인자의 length 필드를 읽어서 알아낸다.
3.

```
public class PrintCommandLineArgs {
   public static void main(String[] args) {
      for(int i=0; i<args.length; i++)
         System.out.println(args[i]);
   }
}
```

Check Time(p.157)

1. try 블록
2.
   (1) java.lang.ArrayIndexOutOfBoundsException
   (2) java.lang.NumberFormatException
   (3) java.lang.NullPointerException

4장 Check Time 정답

Check Time(p.178)

1. ② 흐름도
2. 상속성
3. (1) 상속
   (2) 캡슐화
   (3) 다형성

Check Time(p.198)

1. 수정된 소스는 다음과 같다.

```
class Samp {
   int id;
   public Samp(int x) { // void 삭제
      this.id = x;
   }
   public Samp() { // void 삭제
      this(0); // this(0);는 생성자의 제일 첫 문장이 되어야 한
      System.out.println("생성자 호출");
   }
}
```

2. ConstructorExample 클래스는 int 타입을 인자로 하 성자가 작성되어 있어 컴파일러가 매개변수 없는 기 성자를 자동으로 생성해주지 않는다. 그러나 main(소드에서 객체를 생성할 때 기본 생성자를 호출하 ConstructorExample 클래스에 기본 생성자를 작성하 야 한다.

```
class ConstructorExample{
   int x;
   public  void setX(int x) {this.x = x; }
   public  int getX() {return x; }
   public ConstructorExample(int x) {
      this.x = x;
   }

   // 다음의 기본 생성자를 추가한다.
   public ConstructorExample() {
      this.x = 0;
   }
   public static void main(String [] args) {
      ConstructorExample a = new ConstructorExample(
      int n = a.getX();
   }
}
```

3. this는 현재 실행되는 메소드가 속한 객체에 대한 레 스이고, this()는 클래스의 생성자에서 다른 생성자 출할 때 사용한다.

Check Time(p.204)

1.

```
class Human {
   public String name;
   public int age;
   public double height, weight;
   public Human() {
      name = null;
   }
   public Human(String s) {
      name = s;
   }
}
```

2.

```
public static void main(String[] args) {
   Human aHuman = new Human();
   aHuman.name = "홍길동";
   aHuman.age = 21;
   aHuman.height = 180.5;
   aHuman.weight = 73.2;
}
```

3.

```
Human[] aHuman = new Human[5];
for (int i=0;i<aHuman.length;i++)
   aHuman[i] = new Human();
```

Check Time(p.210)

1.

```
double getSum(double[] d) {
   double sum=0;
   for (int i=0; i<d.length; i++)
      sum = sum + d[i];
   return sum;
}
```

2. 메소드 오버로딩 성공 조건
   (1) 메소드 이름이 동일하여야 한다.
   (2) 메소드의 매개변수 개수나 타입이 서로 달라야 한다.
   (3) 메소드의 이름이 같고 인자의 개수나 타입이 모두 같은데 메소드의 리턴 타입이 다르면 메소드 오버로딩이 성립되지 않는다.
3. 메소드 호출 시 배열의 레퍼런스만 복사되므로 메소드에서 원본 배열의 값을 변경할 수 있다.

Check Time(p.214)

1. 자바에서는 new로 할당받은 객체나 배열 메모리를 사용한 후 시스템에 반환하는 기능이 없다. 그러므로 프로그램에서 더 이상 사용되지 않는 메모리, 즉 가비지를 자동으로 회수하는 가비지 컬렉션이 필요하다. 가비지 컬렉션의 장점은 프로그래머가 메모리를 관리하지 않아도 되는 편리함이며, 단점은 가비지 컬렉션이 진행되는 동안 응용프로그램이 일시 중단된다는 점이다.
2. ④. main()에서 printHello() 메소드를 호출하면 String("hello") 객체가 생성되지만 printHello() 메소드가 종료하고 main()으로 돌아가는 순간 String("hello") 객체는 가비지가 된다.
3. 가비지가 발생한다. for 문에 의해 10번 new Scanner() 객체가 생기지만 이전에 생긴 9개의 객체들은 모두 가비지가 된다.
4. System.gc();

Check Time(p.221)

1. 다음 문장에서 컴파일 오류가 발생한다.

```
new SampleClass();
```

   SampleClass 클래스는 디폴트로 선언되었으므로 다른 패키지에서는 접근이 불가능하다.
2. 다음 문장에서 컴파일 오류가 발생한다.

```
fa.field4 = 3;
```

   SampleClass의 private 필드 field4는 다른 클래스에서 접근할 수 없다.
3. 필드 멤버는 private으로 선언하여 보호하는 것이 바람직하다.

Check Time(p.231)

1. 인스턴스 멤버
2. static 메소드는 객체가 생성되지 않은 상황에서도 사용이 가능하므로 객체에 속한 인스턴스 멤버를 접근할 수 없다.
3. 잘못된 부분은 다음 라인이다.

```
this.a = x;
```

setA()는 static 메소드이므로 this 키워드를 사용할 수 없다.

4. 틀린 부분은 다음과 같다.

```
StaticCheck.s = 20;
```

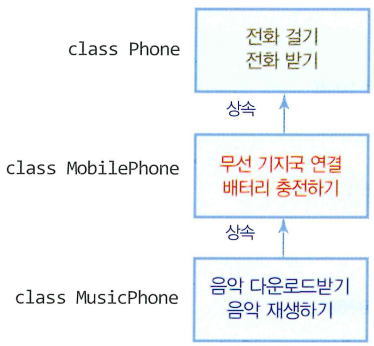
static 메소드에서 non-static 멤버인 s를 접근할 수 없기 때문이다.
다음과 같이 2가지 방법으로 수정할 수 있다.
(1) main() 메소드의 StaticCheck.s = 20;를 삭제
(2) non-static 멤버인 s를 static 멤버로 변경, 즉 static int s;로 변경
예상되는 출력 결과는

```
30 40
```

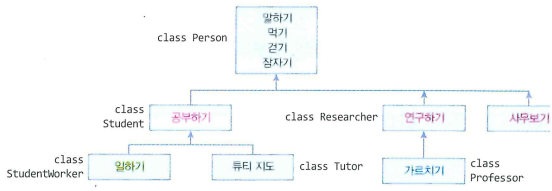
5장 Check Time 정답

Check Time(p.252)

1.



2.



클래스 재사용을 통해 클래스 작성의 용이한 장점을 준다.

Check Time(p.257)

1. extends
2. java.lang.Object
3. 서브 클래스에서 슈퍼 클래스의 private 멤버는 접 수 없다.

Check Time(p.261)

1. private 접근 지정자
2. protected나 public 접근 지정자

Check Time(p.268)

1. super()
2. 결과는 다음과 같다.

```
매개변수생성자A30
생성자B
```

3. super(10)는 다음과 같이 생성자의 가장 첫 행에서 되어야 한다.

```
public B() {
    super(10);
    System.out.println("생성자B");
}
```

Check Time(p.276)

1. 틀린 부분은 아래와 같다.

```
if (a instanceof b)
```

instanceof 연산자의 오른쪽에는 클래스 타입이 와야 ㅎ

2. ④

Check Time(p.289)

1. private 메소드는 오버라이딩할 수 없다.
2. 컴파일은 문제가 없다. 클래스 B의 f(char c) 메소드는 오버로딩이다.
3. 출력은 다음과 같다.

```
Sub Object
Super Object
```

4. 출력은 다음과 같다.

```
Sub Object
Super Object
```

Check Time(p.295)

1.

(1) f() 메소드의 코드가 없으므로 abstract void f(); 로 고쳐야 하며 class A도 abstract class A로 수정되어야 한다.

```
abstract class A {
   abstract void f();
}
```

(2) 틀린 부분이 없다.

(3) 틀린 부분은 다음과 같다.

```
A a = new A(); // 오류
a.f(); // 오류
```

클래스 A는 추상 클래스이므로 인스턴스를 생성할 수 없기 때문이다. 이 두 문장을 삭제하든지 클래스 A에서 abstract 키워드를 제거하면 된다.

2. 추상 클래스의 인스턴스는 생성할 수 없다.

Check Time(p.303)

1. 인터페이스의 객체는 생성할 수 없다.
2. ① implements
3. 인터페이스는 필드를 가질 수 없으므로 다음은 오류이다.

```
int VERSION;
```

또한 static 타입의 다음 메소드는 오류이다.

```
static public int drop();
```

6장 Check Time 정답

Check Time(p.337)

1.

```
package Book; // 이 문장을 추가한다.
public class Cost {
   public static int sum(int a, int b) {return a + b;}
}
```

2.

```
package Using;
import Book.Cost; // 오류가 발생하지 않도록 import 문을 삽입한다.
public class UsingCost {
   public static void main(String[] args) {
      Cost c = new Cost();
      System.out.println(c.sum(1, 2));
   }
}
```

Check Time(p.349)

1. java.lang.Object
2.

```
a==c
a is equal to b
```

Check Time(p.354)

1. Character
2. 64
3. 10000000000
4. 110

Check Time(p.361)

1.

```
public class CountWhiteSpace {
   public static void main(String[] args) {
      String s = "My name is Tom.";
      int numberOfWhiteSpace = 0;
      for (int i=0; i<s.length(); i++) {
         char c = s.charAt(i);
         if (c == ' ' || c == '\t' || c == '\n' || c == '\r')
                                          // 공백문자인 경우
            numberOfWhiteSpace++;
      }
      System.out.println("공백문자는 " + numberOfWhite
                           Space + "개 입니다.");
   }
}
```

2. abcdefg
3. d
4. String str4 = 'a';가 잘못되었다. 'a'는 char 타입이므로 String 객체가 아니다.
5. 자바3.14falsea

Check Time(p.365)

1. ②. StringBuffer 객체는 내부에 문자열을 저장하는 가변 크기의 버퍼를 가지고 있어 스트링 버퍼의 문자열 길이를 객체 생성 후 변경할 수 있다.
2. My name is Tom.
3. sb.insert(4, " not");

Check Time(p.368)

1.

```
import java.util.StringTokenizer;

public class StringTokenizerSample {
   public static void main(String[] args) {
      StringTokenizer st = new StringTokenizer("냉장고,
          세탁기,에어컨,TV,오디오", ","); // 구분자로 , 사용
      while (st.hasMoreTokens())
         System.out.println(st.nextToken());
   }
}
```

2.

```
import java.util.StringTokenizer;

public class StringTokenizerSample {
   public static void main(String[] args) {
      int sum = 0;
      StringTokenizer st = new StringTokenizer("2+3
                  66+88+323", "+"); // 구분자로 +
      while (st.hasMoreTokens())
         sum = sum + Integer.parseInt(st.nextToken(
                              // 문자열을 정수로 변환하여
      System.out.println("총 합은 " + sum);
   }
}
```

Check Time(p.371)

1. 3.0
2. -11.0

7장 Check Time 정답

Check Time(p.392)

1. 배열은 고정 크기로 요소를 다루며, 개발자가 필요라 배열의 크기를 스스로 조절하여야 한다. 또한 삽입, 삭제 등의 연산을 수행하면 요소를 앞으로 혹로 이동시키는 작업을 개발자가 프로그래밍하여야 그러나 컬렉션은 크기 조절이나 요소의 삽입, 삭제 든 것을 처리하기 때문에 개발자의 부담이 적다.
2. ③ StringBuffer
3. Stack<E>는 스택을 일반화시킨 컬렉션으로서 E를 타 개변수라고 부르며, E에 Integer, String, Point 등 타입을 지정하여 특정 타입만 다루는 스택을 만들 다. 예를 들어, Stack<E>가 실수만 다루는 스택이 도 구체화하면 Stack<Double>로 표현한다.

Check Time(p.419)

1. 3. 정확히 말하면 Integer(3)
2. v.add("100");이다. 벡터 v에는 문자열을 저장할 수
3. HashMap<Integer, String> m = new HashMap<Inte String>();

4.

```
Vector<Integer> v = new Vector<Integer>();
Iterator<Integer> it = v.iterator(); // Iterator 객체를
                                     얻어온다.
while(it.hasNext()) { // 요소를 모두 방문할 때까지 루프
   System.out.println(it.next()); // 다음 요소 출력
}
```

Check Time(p.426)

1. 타입 매개 변수에 자바의 기본 타입은 사용할 수 없다. int의 사용은 부적절하다. Integer로 수정하면 된다.
2.

```
interface MyList<T> {
      public T get();
      public void add(T ob);
}
```

3. 타입 매개변수가 나타내는 타입의 객체 생성은 불가능하므로 E a = new E();이 잘못되었다.
4.

```
public static <T> void printStack(GStack<T> a) {
   GStack<T> gs = reverse(a); // 원소의 출력은 인덱스 순서대
                              로 하므로 스택을 거꾸로 해야 한다.
   while(true) {
      T tmp;
      tmp = gs.pop();
      if(tmp == null)
         break;
      System.out.println(tmp);
      a.push(tmp); // 스택을 원래대로 돌려놓기 위해 다시 역순으
                   로 push
   }
}
```

8장 Check Time 정답

Check Time(p.446)

1. (1) 동영상 파일(.avi) - 바이트 스트림
   (2) 메모장으로 작성한 파일(.txt) - 문자 스트림(바이트 스트림도 무관함)
   (3) 자바 클래스 파일(.class) - 바이트 스트림
   (4) HTML 파일(.html) - 문자 스트림(바이트 스트림도 무관함)
2. Reader와 Writer

Check Time(p.463)

1. ④ 없다.
2. FileOutputStream 클래스
3.

```
import java.io.*;

public class CheckTimeFileOutputStream {
    public static void main(String[] args) {
      try {
         FileOutputStream fout = new FileOutputStream
                                 ("C:\\Temp\\test.dat");
         FileInputStream fin = null;
         fout.write(3);
         fout.write(5);
         fout.close();
         fin = new FileInputStream("C:\\Temp\\test.dat");
         int c=0;
         c = fin.read();
         System.out.print(c + " ");
         c = fin.read();
         System.out.print(c + " ");
         fin.close();
      } catch (IOException e) {
         System.out.println("입출력 오류");
      }
    }
}
```

Check Time(p.468)

1. 버퍼를 가진 입출력이 시스템 관점에서 이롭다. 버퍼를 가지고 있으면 운영체제에 대한 호출 횟수를 줄이고, 하드 디스크 등 하드웨어에 대한 작동 횟수를 줄이기 때문이다.
2. flush()

3.

```
import java.io.*;

public class FileReaderEx {
    public static void main(String[] args) {
      BufferedReader in = null
      try {
         in = new BufferedReader(new
         FileReader("c:\\windows\\system.ini"), 2048);
                              // 버퍼 크기 2048의 스트림 생성
         int c;
         while ((c = in.read()) != -1) {
            System.out.print((char)c);
         }
         in.close();
      } catch (IOException e) {
         System.out.println("입출력 오류");
      }
    }
}
```

Check Time(p.476)

1. ③. File 클래스가 파일을 읽고 쓰는 기능이 없으므로 파일의 내용은 읽을 수 없다.
2. 한글 파일은 바이너리 파일이므로 바이트 스트림인 FileInputStream/FileOutputStream을 이용해야 한다.
3.

```
import java.io.*;

public class FileExistEx {
    public static void main(String[] args) {
      File f = new File("c:\\Temp\\test.txt");
      if (f.exists())
         System.out.println("exist");
      else
         System.out.println("no");
    }
}
```

4.

```
import java.io.*;

public class FileDeleteEx {
    public static void main(String[] args) {
      File f = new File("c:\\Temp\\test.txt");
      f.delete();
    }
}
```

5.

```
import java.io.*;

public class FileLengthEx {
    public static void main(String[] args) {
      File f = new File("c:\\Temp\\test.txt");
      System.out.println(f.length());
    }
}
```

6.

```
import java.io.*;

public class FileRenameEx {
    public static void main(String[] args) {
      File f = new File("c:\\Temp\\test.txt");
      f.renameTo(new File("c:\\Temp\\res.txt"));
    }
}
```

9장 Check Time 정답

Check Time(p.497)

1. AWT
2. AWT이다. 그러나 스윙은 자바 응용프로그램의 GUI 도
   맥(Mac), 윈도우(Windows), 리눅스(Linux)의 3 가지
   서 선택할 수 있도록 지원하는 패키지를 가지고 있다
3. AWT의 실행 속도가 빠르다. 그 이유는 GUI를 그리기
   운영 체제를 직접 이용하기 때문이다.

Check Time(p.507)

1. ① JPanel
2. JFrame 내에 컴포넌트가 달리는 특별한 영역을 컨테

이라고 부른다.

```
Container c = getContentPane();
c.add(new JButton("Hello"));
```

Check Time(p.523)

1. 컨테이너는 컴포넌트이기도 하지만 다른 컴포넌트를 포함할 수 있는 기능을 가지고 있다. 그러나 컴포넌트는 다른 컴포넌트를 포함할 수 없다.
2. `container.setLayout(new FlowLayout(10, 20));`
3. `container.setLayout(null);`
4. 배치관리자의 기능은 컨테이너에 부착되는 혹은 이미 부착된 컴포넌트들의 위치와 크기를 적절히 설정하는 것이다. 배치관리자가 없다면 개발자가 프로그램에서 컴포넌트의 위치와 크기를 설정하여야 한다. 그렇지 않으면 컴포넌트는 보이지 않게 된다.

10장 Check Time 정답

Check Time(p.551)

1. `ActionListener`와 같이 이벤트 리스너의 메소드가 한 개뿐인 경우 구현할 메소드가 적기 때문에 효과적이다.
2. 이벤트 리스너를 독립된 클래스로 작성하는 것보다 내부 클래스로 작성하면 내부 클래스를 둘러싸고 있는 외부 클래스의 멤버를 마음대로 접근할 수 있기 때문에 프로그램 작성이 쉬운 장점이 있다.

Check Time(p.554)

1. 어댑터는 클래스이다.
2. 리스너 인터페이스를 이용하여 이벤트 리스너를 작성하면 인터페이스의 모든 메소드를 구현하여야 한다. 그러나 때에 따라 메소드 중에 일부만 구현하고자 하는 경우 어댑터 클래스를 이용하면 프로그래밍이 간단해진다.
3. `ActionListener`와 `ItemListener` 인터페이스는 오직 한 개의 메소드만 가지고 있기 때문에 굳이 어댑터 클래스가 필요 없다. 개발자는 항상 이 유일한 메소드를 구현하여야 하기 때문이다.

Check Time(p.565)

1. ② `getKeyCode()`
2. # 키가 입력되면 프로그램을 종료시키는 `MyKeyListener` 클래스는 다음과 같다.

```
class MyKeyListener extends KeyAdapter {
   public void keyPressed(KeyEvent e) {
      if(e.getKeyChar() == '#')
         System.exit(0);
   }
}
```

3. ③ `keyTyped()`
4. ③ `KeyListener`의 `keyPressed()` 메소드를 반드시 구현하여야 한다.
5. ①과 ③

Check Time(p.572)

1. 동일한 컴포넌트에서 마우스가 두 번 클릭되어야 하기 때문
2.

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class But extends JFrame {
   public But() {
      setTitle("Left Click and Right Click");
      setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      JButton b = new JButton("버튼");
      getContentPane().add(b, BorderLayout.CENTER);
      b.addMouseListener(new MouseAdapter() {
         public void mousePressed(MouseEvent e) {
            Component c = (Component)e.getSource();
            if(e.getButton() == MouseEvent.BUTTON1) {
               c.setBackground(Color.RED);
            }
            else if(e.getButton() == MouseEvent.BUTTON3) {
               c.setBackground(Color.BLUE);
            }
         }
      });
```

```
        setSize(300,200);
        setVisible(true);
    }

    public static void main(String [] args) {
        new But();
    }
}
```

12장 Check Time 정답

Check Time(p.642)

1. ③ paintComponent(Graphics g)
2. java.awt.Graphics
3. ④ JPanel

Check Time(p.647)

1. 자바에서 사용할 수 있는 색은 최대 256x256x256=224=16,581,375개의 색이다.
2.

```
g.setColor(Color.BLUE);
g.drawString("BLUE", 20, 20);
```

3.

```
class MyPanel extends JPanel {
    public void paintComponent(Graphics g) {
        super.paintComponent(g);
        g.setFont(new Font("Times New Roman", Font.PLAIN, 30));
        g.drawString("We win!!", 100,100);
    }
}
```

Check Time(p.652)

1.

(1) (10, 10)에서 (30,30)까지의 노란색 선 그리기

```
g.setColor(Color.YELLOW);
g.drawLine(10,10,30,30);
```

(2) 중심이 (20, 20)이고 높이가 10, 폭이 20인 사
내접하는 타원 그리기

```
g.drawOval(10,15,20,10);
```

(3) (10,10), (5,15), (15,20)의 3 점으로 구성되는
칠하기. 삼각형의 내부 색은 빨간색으로 한다.

```
int x[]={10,5,15};
int y[]={10,15,20};
g.setColor(Color.RED);
g.fillPolygon(x,y,3);
```

(4) 다음과 같은 3색 원을 그려라. 각 색이 차지하는
은 동일하다.

```
import java.awt.*;
import javax.swing.*;

public class ThreeColorCircle extends JFrame {
    public ThreeColorCircle() {
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE
        setContentPane(new MyPanel());
        setSize(300,300);
        setVisible(true);
    }
    class MyPanel extends JPanel {
        public void paintComponent(Graphics g) {
            super.paintComponent(g);
            g.setColor(Color.RED);
            g.fillArc(0, 0, 100, 100, 90, 120);
            g.setColor(Color.BLUE);
            g.fillArc(0, 0, 100, 100, 210, 120);
            g.setColor(Color.YELLOW);
            g.fillArc(0, 0, 100, 100, 330, 120);
        }
    }
    public static void main(String[] args) {
        new ThreeColorCircle();
    }
}
```

Check Time(p.660)

1.

```
import java.awt.*;
import javax.swing.*;

public class TwoImageFrame extends JFrame {
   public TwoImageFrame() {
      setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      setContentPane(new MyPanel());
      setSize(300,300);
      setVisible(true);
   }
   class MyPanel extends JPanel {
      Image imgA = new ImageIcon("images/a.jpg").getImage();
      Image imgB = new ImageIcon("images/b.jpg").getImage();

      public void paintComponent(Graphics g) {
         super.paintComponent(g);
         int width=getWidth()/2;
         g.drawImage(imgA, 0, 0, width, getHeight(), this);
         g.drawImage(imgB, width, 0, width, getHeight(), this);
      }
   }
   public static void main(String[] args) {
      new TwoImageFrame();
   }
}
```

Check Time(p.663)

1. ③
2.

```
g.setClip(100,100,100,100); // (100,100)에서 100x100 크기
                            를 클리핑 영역으로 지정
```

Check Time(p.669)

1.

(1) component.repaint() 메소드

(2) container.revalidate();

13장 Check Time 정답

Check Time(p.687)

1. 자전거 타면서 음악 듣기
2. 멀티스레딩
3. ③. 1에서 100000까지 더하기를 2개의 스레드를 작성하여 한 스레드는 짝수만 더하고, 다른 스레드는 홀수만 더하여 최종적으로 두 합을 더하여 결과를 내는 응용프로그램.
그 이유는 다음과 같다. 컴퓨터가 두 개의 스레드를 동시에 처리한다고 하더라도 하나의 CPU만 가진 경우 시간을 분할하여 두 개의 스레드를 돌아가면서 실행할 수밖에 없다. 그러므로 실행하고자 하는 스레드가 입출력 작업이 많아 대기하는 시간이 많은 경우, 대기하는 시간 동안 다른 스레드가 CPU를 사용할 수 있도록 하면 멀티스레드 사용이 보다 효과적이다. 그러나 이 보기의 경우처럼 홀수만 합하는 스레드와 짝수만 합하는 두 개의 스레드가 실행되는 시간의 합은 1에서 100000까지 더하는 시간뿐만 아니라 스레드의 문맥 교환에 따른 부가적인 시간이 소요되므로 하나의 스레드가 1에서 100000까지 더하는 경우보다 소요 시간이 커지게 되어 오히려 비효율적이다.

Check Time(p.689)

1. 하나의 자바 가상 기계는 오직 하나의 자바 응용프로그램만을 실행한다.
2. 하나의 자바 가상 기계는 여러 개의 자바 스레드를 실행할 수 있다.
3. 스레드의 실행을 책임지는 것은 자바 가상 기계(JVM)이다. 스레드를 스케줄링하여 실행하고, 중단하고, 잠재우고, 종료시키는 것은 전적으로 자바 가상 기계의 역할이다. 다만 개발자는 스레드가 실행을 시작하도록 초기에 start() 메소드를 호출할 뿐이다.

Check Time(p.702)

1. public void run() 메소드
2. InterruptedException
3.

```
public class ExitAfter10MinApp {
   public static void main(String[] args) {
      TenSecondThread th = new TenSecondThread();
```

```
        th.start();
    }
}

class TenSecondThread extends Thread {
    public void run() {
        System.out.println("스레드가 시작됩니다.");
        try {
            Thread.sleep(10000);
        }
        catch(InterruptedException e) {}
        System.out.println("스레드 종료합니다.");
    }
}
```

4.

```
public class ExitAfter10MinApp2 {
    public static void main(String[] args) {
        Thread th = new Thread(new TenSecondRunnable());
        th.start();
    }
}
class TenSecondRunnable implements Runnable {
    public void run() {
        System.out.println("스레드가 시작됩니다.");
        try {
            Thread.sleep(10000);
        }
        catch(InterruptedException e) {}
        System.out.println("스레드 종료합니다.");
    }
}
```

Check Time(p.707)

1. 화면에 출력하는 것은 입출력 행위이므로 자바의 경우 입출력이 발생하면 입출력을 수행하는 스레드는 BLOCK 상태가 된다. 그리고 입출력이 모두 종료하면 RUNNABLE 상태로 돌아오며 자바 가상 기계에 의해 스케줄링되면 다시 실행된다.
2. yield() 메소드
3. 1에서 10 사이
4. 자바에서는 철저히 우선순위를 기반으로 스케줄링된다. 그러므로 우선순위가 높은 스레드가 존재하는 동안 우선순위가 낮은 스레드는 절대로 실행될 수 없다.
5. 응용프로그램이 스레드의 우선순위를 조절할 수 있다

Check Time(p.714)

1. 한 번 종료한 스레드의 Thread 객체는 더 이상 새로 레드 생성에 이용할 수 없다.
2. Thread 클래스의 run() 메소드에서 return 문은 스레 종료시키는 결과를 낳는다.

Check Time(p.721)

1. ①
2. synchronized

Check Time(p.727)

1. wait(), notify(), notifyAll() 메소드는 모두 Obje 멤버이다. 그러므로 모든 자바 객체들이 이 메소드 지고 있다.
2. 멀티스레드의 동기화이다. wait() 메소드는 객체를 사용 중인 스레드가 깨워주기를 기다리면서 대기하며, notify()는 객체를 기다리는 스레드 중 하나만 고, notifyAll()은 모든 스레드를 깨운다.
3. 모든 객체는 Object를 상속받으므로 모든 객체가 wa notify(), notifyAll() 메소드를 가지고 있다. 그러 모든 객체가 동기화 객체로 사용될 수 있다.

15장 Check Time 정답

Check Time(p.803)

1. IP 주소는 네트워크상에서 유일하게 식별될 수 있는 워크 장치의 주소이며, 한 컴퓨터 내의 각 응용프로은 통신을 위해 가상의 연결단인 포트를 생성하고 트 번호로 자신을 식별하게 한다.
2. 윈도우 PC에서 명령창을 열어 ipconfig 명령을 수행한
3. 컴퓨터가 데이터를 수신하게 되었을 때 어떤 프로그 게 전달해 주어야 할지 판단할 수 없게 된다.
4. 하나의 통신 프로그램은 여러 개의 포트를 사용할 수

찾아보기

가비지 50, 197, 211
가비지 컬렉션 50
가비지 컬렉션 스레드 213
가비지 컬렉터 211
값 410
값에 의한 호출 204
강제 타입 변환 73
객체 172, 178
객체 배열 199
객체 비교 345
객체 소멸 210
객체 전달 206
객체 지향 언어 172, 176
객체 치환 197
경량 컴포넌트 493
구분 문자 365
구체화 390, 420
그래픽 기반 GUI 프로그래밍 584, 642
기계어 19
기본 생성자 190
기본 타입 66

나열 135, 136
내부 클래스 547
논리 리터럴 70
논리적 오른쪽 시프트 91

다운캐스팅 271
다이얼로그 754
다중 if-else 문 97
다중 상속 256, 298
다차원 배열 138
다형성 175, 279
대입 연산자 83
데몬 스레드 702
데이터 타입 65
도형 그리기 648
도형 칠하기 651
동적 바인딩 277, 279, 284, 285
디렉터리 생성 469
디지털 오디오 784
디폴트 멤버 219
디폴트 배치관리자 510
디폴트 생성자 190
디폴트 클래스 217
디폴트 패키지 336

라디오버튼 602
레퍼런스 65, 130, 182
레퍼런스 배열 200
레퍼런스 치환 132
리스너 인터페이스 541
리터럴 68
리터럴 테이블 355
링크 26, 27, 28

멀티스레드 49
멀티스레드 동기화 402
멀티스레딩 685
멀티태스킹 682
메뉴 740
메모장 452
메소드 60, 173, 180, 204
메소드 오버라이딩 277
메소드 오버로딩 208
메소드 중복 208
메인 스레드 706
모달 다이얼로그 760
모달리스 다이얼로그 760
무한 반복 116
문자 리터럴 69, 102
문자 버퍼 스트림 464
문자 스트림 443
문자 집합 45
문자열 66, 67
문자열 그리기 643
문자열 리터럴 71, 102
문장 6
미디 데이터 784

바이트 87
바이트 버퍼 스트림 464
바이트 스트림 443, 457
바이트 코드 24, 25
박싱 353
반복문 114
반복자 408
배열 128
배열 공유 132
배열 리턴 143
배열 생성 130
배열 선언 129
배열 전달 207
배열의 크기 134
배치관리자 508
버튼 그룹 602, 603
버퍼 463
버퍼 스트림 464
버퍼 입출력 463
벡터 393
벡터의 용량 397
벡터의 크기 397

변수 60, 67
부모 클래스 250
비교 연산자 85
비정방형 배열 140
비트 87
비트 논리 연산 87, 88
비트 시프트 연산 87, 89

산술 연산자 81
산술적 오른쪽 시프트 91
산술적 왼쪽 시프트 91
삼항 연산자 86
상속 174, 250
상수 64, 72
생성자 180, 186, 261
서버 804
서버 소켓 804, 809
서브 클래스 174, 252
서블릿 46
소켓 803
소프트웨어 18
수직 정렬 594
수평 정렬 594
순차 검색 408
슈퍼 클래스 174, 252
스레드 683
스레드 동기화 714, 716
스레드 스케줄링 689
스레드 우선순위 705
스레드 종료 707
스윙 컴포넌트 585
스윙 프레임 501
스윙의 페인팅 664
스트림 442
스트림 닫기 448, 455
스트림 생성 447
스트링 리터럴 355
식별자 62
실수 리터럴 69
실시간 50
실행시간 오류 150

안드로이드 47
어댑터 클래스 551
어셈블리어 19
언더스코어 71
언박싱 353
업캐스팅 269
연산자 80
연산자 우선순위 81
예외 150
예외 처리 152
예외 클래스 153
오디오 API 784
오디오 스트림 785
오디오 클립 785
오버라이딩 175, 288
오버로딩 175, 288
오크 21
오픈 소스 30
원호 650
유니코드 66, 70, 447
이미지 그리기 652
이미지 레이블 589
이미지 로딩 654
이미지 버튼 591
이미지 체크박스 597
이벤트 534
이벤트 객체 537, 538
이벤트 기반 프로그래밍 534
이벤트 리스너 534, 535, 537, 541
이벤트 분배 스레드 536, 537
이벤트 소스 537
이벤트 처리 스레드 507
이스케이프 시퀀스 70
이클립스 36, 40
익명 클래스 548
인덱스 128
인스턴스 178
인스턴스 멤버 223
인자 전달 204
인터페이스 296
인터페이스 상속 298
일반화 390
입출력 스트림 442

자동 박싱 353, 395, 399
자동 언박싱 353, 399
자동 타입 변환
자바
자바 API
자바 가상 기계
자바 스레드
자바 컴파일러
자바 키워드
자바스크립트
자식 클래스
잘 알려진 포트
전역 변수
전역 함수
절대 위치
절대 크기
절차 지향 언어
접근 지정자 59, 180
정렬
정방형 배열
정수 리터럴 68
제네릭
제네릭 메소드
제네릭 클래스
제네릭 타입
제임스 고슬링
조건 연산
조건문
주석문
중량 컴포넌트
중첩 if-else 문
중첩 반복
증감 연산자
지역변수

채팅
최상위 컨테이너 499
추상 메소드
추상 클래스

캐스팅 74,
캡슐화 48, 172,
컨테이너
컨텐트팬 501,
컬러 다이얼로그
컬렉션

컴파일 20, 26
컴파일러 190, 209, 263
컴포넌트 499
컴포넌트 기반 GUI 프로그래밍 584
클라이언트 804
클라이언트 소켓 804, 806
클래스 173, 178
클래스 로더 28
클래스 로딩 27, 227
클래스 만들기 59
클래스 멤버 223
클래스 파일 48
클리핑 660
키 410

타이틀 504
타입 변환 73, 269
태스크 684
탭 781
탭팬 781
템플릿 390
토큰 365
통합 개발 환경 36
툴바 746
툴팁 750
특수문자 리터럴 70

파일 고속 복사 475
파일 다이얼로그 770
파일 리스트 469
파일 복사 472
파일 삭제 469
파일 쓰기 455, 459
파일 열기 다이얼로그 770
파일 읽기 447, 452, 461
파일 저장 다이얼로그 770
파일 크기 469
파일 필터 772
팝업 다이얼로그 765
패키지 32, 49, 216, 322, 324
패키지 선언 329
폐다각형 650
포커스 554
포트 802
표준 입력 스트림 75
프레임 종료 507
플랫폼 21
플랫폼 독립성 48
플랫폼 독립적 21
플랫폼 종속성 22
피연산자 80
필드 173, 180

하드웨어 18
해시맵 410
해시맵 생성 411
확장 251
힙 메모리 355

Abstract 290
accept() 810, 812
ActionEvent 540
ActionListener 541, 542
actionPerformed 543
add() 395, 404
addActionListener 544
AdjustmentEvent 540
AdjustmentListener 542
AND 85
API 도큐먼트 34
append() 363
ArithmeticException 151, 153, 154
ArrayIndexOutOfBoundsException 153, 155
ArrayList<E> 402
auto boxing 353
auto unboxing 353
AWT 492

BinarySearch() 417
Boolean 349
boolean 65, 70
BorderLayout 509, 514
boxing 353
break 100
break 문 102, 124, 126
BufferedInputStream 475
BufferedOutputStream 465, 475
ButtonGroup 603

Byte 34
byte 6

Calendar 37
call-by-value 20
capacity() 363, 39
CardLayout 50
case 문 100, 10
catch 15
Change 이벤트 62
char 6
Character 34
charAt() 35
class 59, 18
ClassCastException 15
clear() 40
Clip 인터페이스 78
close() 78, 455, 808, 81
Collections 41
Color 64
compareTo() 35
Component 497, 49
ComponentEvent 54
ComponentListener 54
concat() 35
Container 49
ContainerEvent 54
ContainerListener 54
contains() 35
continue 문 12

Default constructor 19
default 문 10
delete 21
delete() 363, 46
delimiter 36
Double 34
double 6
do-while 문 12
downcasting 27
drawImage() 65
drawString() 64

ElementAt() 396, 40

enumeration 135
EOF 461
equals() 345, 346
Exception 150
exists() 469
extends 252

F 69
false 70
File 468
File 클래스 468
FileInputStream 460
FileNotFoundException 449
FileOutputStream 458
FileReader 447, 472
FileWriter 455, 472
final 72
finalize() 211
finally 152
Float 349
float 65
FlowLayout 509, 511
flush() 466, 808
focus 554
FocusEvent 540
FocusListener 542
Font 645
for 문 115
for-each 문 135

Garbage 197
gc() 213
generics 388
get() 371, 372, 396, 405, 410, 412
getButton() 568
getClass() 342
getClickCount() 568
getContentPane() 504
getInputStream() 809, 810
getInstance() 372
getKeyChar() 557
getKeyCode() 557
getKeyText() 559
getName() 343, 469
getOutputStream() 809, 810
getParent() 469, 586
getPath() 469
getPoint() 567
getSource() 539
getTopLevelAncestor() 586
getX() 567
getY() 567
Graphics 643
GridLayout 509, 517
GUI 492
GUI 컴포넌트 492

HashCode() 342
HashMap<K, V> 410

IDE 36
if 94
if-else 95
IllegalArgumentException 153
import 76, 325
InputMismatchException 153, 156
InputStreamReader 451
insert() 363
instanceof 271, 274
int 65
Integer 349
Integer.parseInt 157
interface 296
interrupt() 708
InterruptedException 708
IOException 153, 449
IP 주소 801
ipconfig 801
isDirectory() 469
isFile() 469
Item 이벤트 599, 605
ItemEvent 540, 599
ItemListener 542, 599
Iterator 408
iterator() 408

Jar 28
java 39
javac
javadoc
javap
JButton
JCheckBox
JColorChooser
JComboBox<E>
JComponent 497, 499
JDialog
JDK 29, 31
JFileChooser
JFrame
JLabel
JList<E>
JMenu
JMenuBar
JMenuItem
JOptionPane
JPanel
JRadioButton
JRE
JScrollPane 610
JSlider
JSP
JTabbedPane
JTextArea
JTextField
JToolBar

Key 이벤트
KeyEvent
KeyListener
keyPressed()
keyReleased()
keySet()
keyTyped()

LastModified()
length
length() 357,
Line 이벤트
LineEvent
LineListener
list()

listFiles() 469
ListSelectionEvent 540
ListSelectionListener 542
localhost 801
Long 349
long 65

Main 스레드 507, 705, 706
main() 59, 503
main() 메소드 49, 146
Math 369
method overloading 208
method overriding 277
mkdir() 469
Mouse 이벤트 565
MouseAdapter 552
MouseEvent 540
MouseListener 541, 542
MouseMotionListener 542
MouseWheelEvent 572
MouseWheelListener 565, 572
mouseWheelMoved 572
MS949 452

New 130, 138, 181, 182, 211
next() 78
nextDouble() 78
nextInt() 78
nextLine() 78
nextToken() 367
nodify() 722
non-static 멤버 222
NOT 85
notifyAll() 723
null 71
NullPointerException 153
NumberFormatException 153, 157

Object 256, 341
OR 85
OutOfMemoryError 153
overloading 175
overriding 175

Package 216, 329
paint() 664
paintBorder 664
paintChildren 664
paintComponent 664
paintComponent() 638
Polymorphism 175
port 802
private 멤버 218
protected 멤버 219, 259
public 180
public 멤버 218
public 클래스 217
put() 410

Random 370
random() 369
read() 461
Reader 445
remove() 397, 405, 412
removeAllElements() 397
repaint() 665
replace() 357, 363
requestFocus 555
revalidate() 666
reverse() 363, 417
rt.jar 324, 339
run() 691
Runnable 인터페이스 695
Runtime 213

Scanner 76
ServerSocket 809
ServerSocket 클래스 809
Set 컬렉션 412
set() 371, 372
setBorderPainted 597
setContentPane() 505
setEnabled() 586
setFocusable 555
setIcon 592
setLayout() 511, 520
setLength() 363
setPressedIcon 592
setRolloverIcon 59
setSelectedIcon 59
setTitle() 50
setVisible() 58
Short 34
short 6
showConfirmDialog() 76
showInputDialog() 76
showMessageDialog() 76
size() 397, 41
socket 80
Socket 80
Socket 클래스 80
sort() 41
split() 357, 36
start() 693, 69
static 22
static 멤버 22
Stream 44
String 66, 35
StringBuffer 36
StringTokenizer 36
sub class 174, 25
subString() 35
super 286, 28
super class 174, 25
super() 265, 287, 50
super.paintComponent() 64
Swing 49
switch 문 100, 10
synchronized 71
System 21
System.exit(0) 15
System.gc() 21
System.in 75, 44

TCP/IP 80
this 192, 230, 28
this() 195, 28
Thread 클래스 69
token 36
toLowerCase() 35
ToolTipManager 75
Top Level Container 499, 50

toString() 342, 343
toUpperCase() 357
trim() 359
true 70
try 152
try-catch-finally 152

Unboxing 353
upcasting 269

Vector<E> 393
void close() 809

Wait() 722
well-known ports 802
while 문 119
WindowEvent 540
WindowListener 542
WORA 22, 23
Wrapper 349
write() 455, 459
Writer 445

XOR 85

@Override
@SuppressWarnings
\u
<<
== 연산자
>>
>>>
0x000000FF
2차원 배열