

Package와 Jar

남 광 우

강의자료 : head first java

애플리케이션 배포

- **로컬**
  - 애플리케이션 전체가 최종사용자의 컴퓨터에서 실행 가능한 JAR로 배포된 독립적인 프로그램으로 실행됩니다.

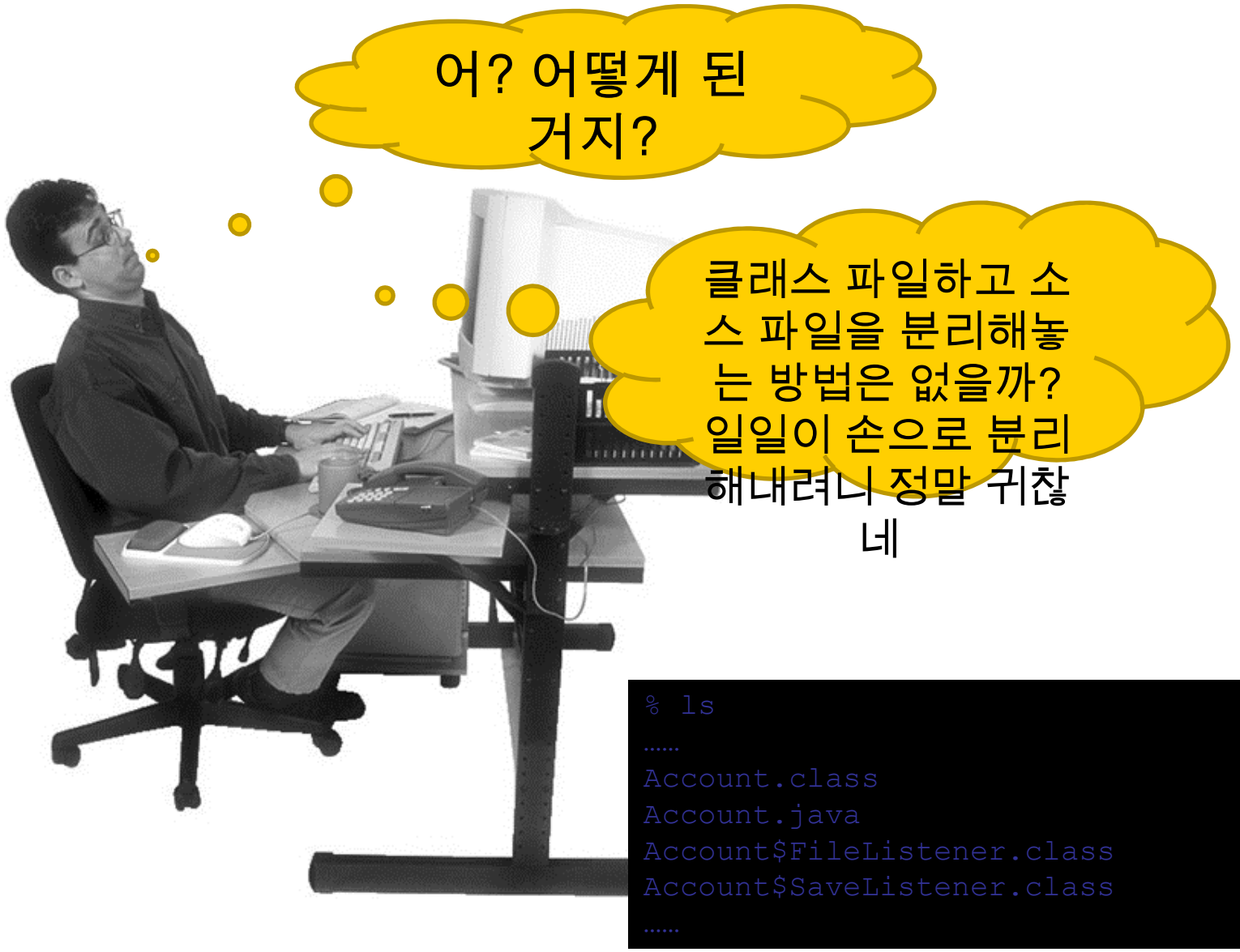
- **로컬과 원격의 혼합형 (세미로컬)**
  - 애플리케이션의 일부분이 돌아가고 있는 서버와 사용자의 로컬 시스템에서 실행되면서 서버에 연결되어 작업을 처리하는 클라이언트 부분으로 나뉘어져 있습니다.

- **원격**
  - 자바 애플리케이션 전체가 서버 시스템에서 돌아가고 클라이언트는 웹 브라우저와 같은 자바를 사용하지 않는 방법을 통해서 서버에 접근합니다.

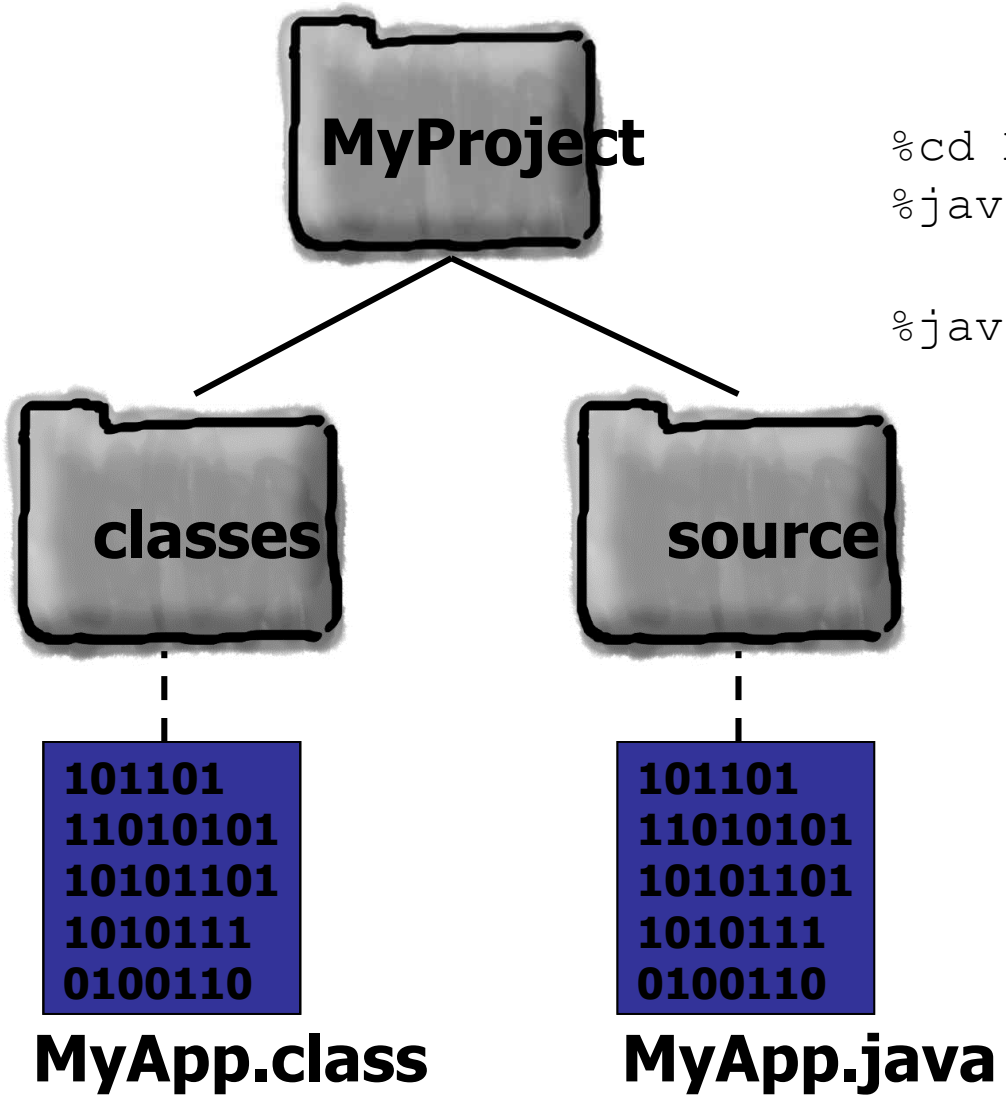
-d 컴파일 옵션



-d 컴파일 옵션


어? 어떻게 된 거지?
클래스 파일하고 소스 파일을 분리해놓는 방법은 없을까? 일일이 손으로 분리해내려니 정말 귀찮네
% ls
……
Account.class
Account.java
Account$FileListener.class
Account$SaveListener.class
……

-d 컴파일 옵션




```
%cd MyProject/source
%javac -d ../classes MyApp.java

%javac -d ../classes *.java
```

```
%cd MyProject/classes
%java MyApp
```

JAR(Java ARchive)

- pkzip 파일 유형을 기반으로 합니다.
- 모든 클래스를 하나로 묶을 수 있게 해 주기 때문에 수십 개의 파일을 그대로 전달하는 대신 JAR 파일 하나만 배포할 수도 있습니다.
- manifest 파일을 통해서 main() 메소드가 들어있는 클래스를 알려주면 실행 가능한 JAR 파일을 만들 수 있습니다.

실행 가능한 JAR을 만드는 방법

1. 클래스 파일이 모두 classes 디렉토리에 들어있는지 확인

2. main 메소드가 어떤 클래스에 들어있는지 알려주기 위한 manifest.txt 파일 만들기

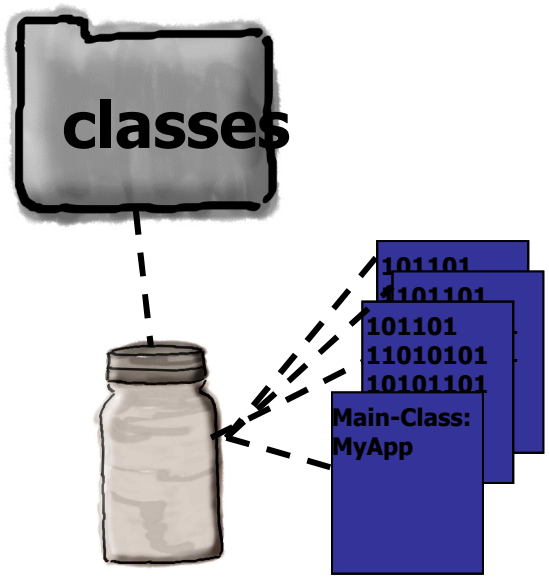
```
Main-Class: MyApp
```

3. jar를 실행시켜서 classes 디렉토리에 있는 모든 파일이 저장된 JAR 파일 만들기

```
%cd MyProject/classes
%jar -cvmf manifest.txt app1.jar *.class
%jar -cvmf manifest.txt app1.jar MyApp.class
```

JAR를 실행시키는 방법

- JAR 파일의 압축을 풀지 않고도 프로그램을 실행할 수 있습니다.

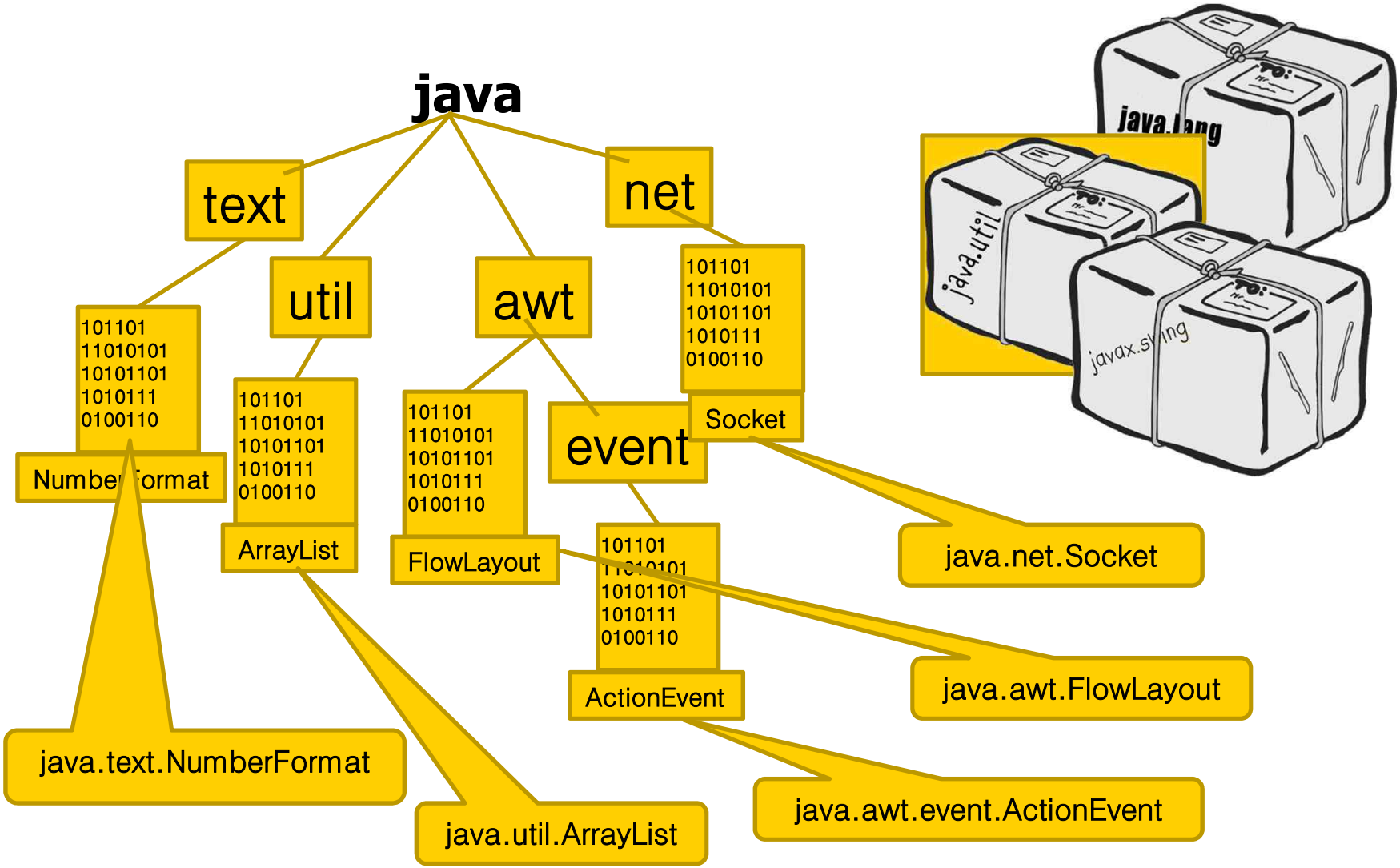



```
%cd MyProject/classes
%java -jar app1.jar
```

패키지

- 똑같은 이름을 가진 클래스가 두 개 이상 있으면 아주 골치 아픈 상황을 맞닥뜨리게 됩니다.
- 클래스명 충돌을 막으려면 패키지를 적절히 활용하는 것이 좋습니다.
- 다른 사람들도 쓸 수 있는, 재사용성이 좋은 클래스를 만들고 싶다면 패키지를 활용하여 다른 클래스와 조화롭게 사용할 수 있는 형태로 구성해줘야 합니다.

패키지


java
text
util
awt
net
101101
11010101
10101101
1010111
0100110
Socket
event
ArrayList
FlowLayout
ActionEvent
java.net.Socket
java.awt.FlowLayout
java.text.NumberFormat
java.util.ArrayList
java.awt.event.ActionEvent
java.lang
java.util
javax.swing

패키지 명명법

- **패키지명 충돌 문제는 어떻게 해결해야 할까요?**

- **역도메인 패키지명**
  - 패키지명 앞에 도메인명을 역순으로 쓰는 방법
  - headfirstjava.com 도메인을 사용하는 사람이 만든 Book이라는 클래스의 전체 이름
    - com.headfirstjava.Book
  - headfirstjava.com 도메인을 사용하는 사람이 만든 projects라는 패키지에 들어있는 Chart라는 클래스의 전체이름
    - com.headfirstjava.projects.Chart



클래스를 패키지에 집어넣는 방법

1. **패키지명 결정**

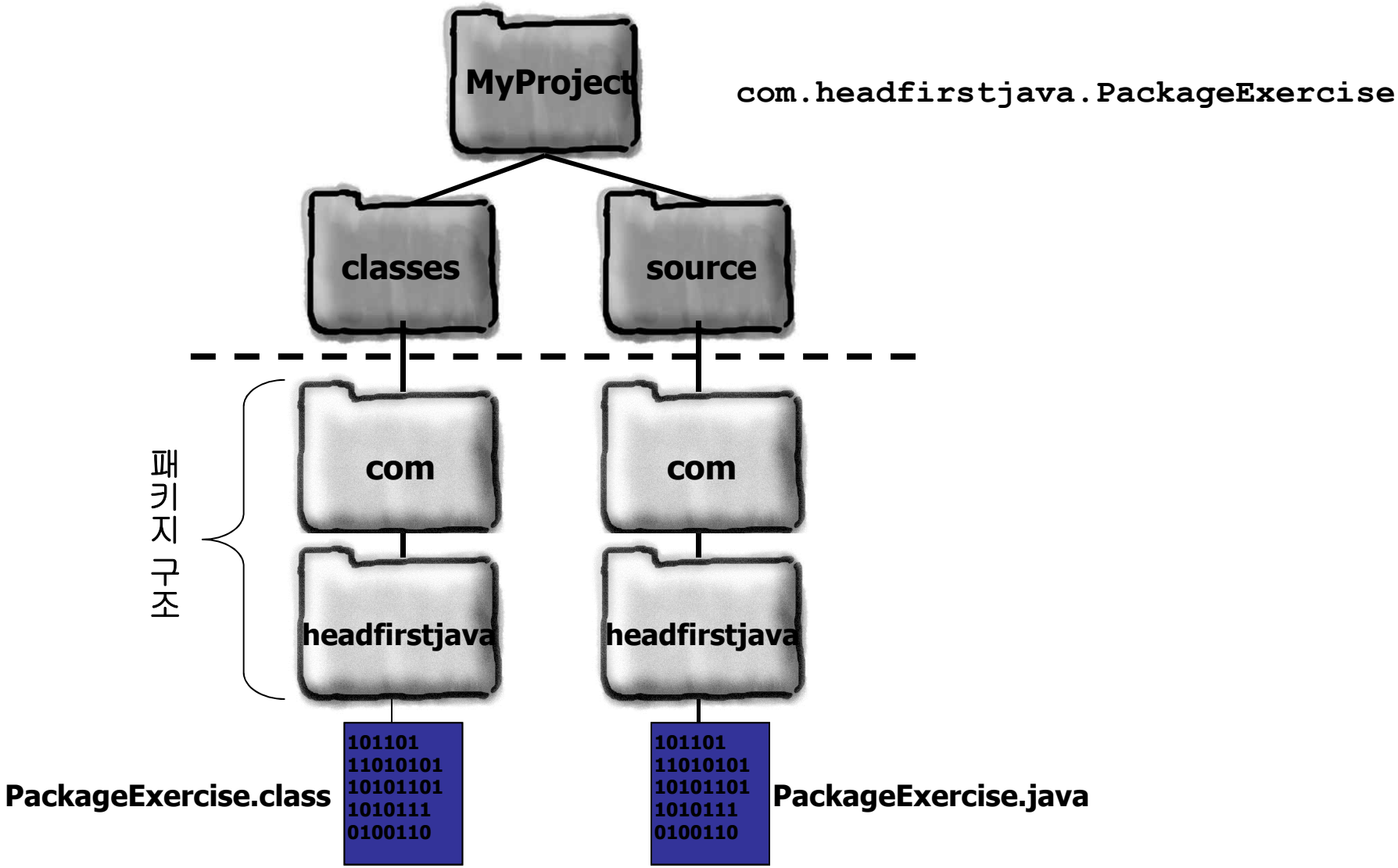
```
com.headfirstjava.PackageExercise
```

2. **클래스에 package 선언문 추가**

```
package com.headfirstjava;
import javax.swing.*;
public class PackageExercise {
    …
}
```

3. **디렉토리 구조를 패키지 구조에 맞게 설정**

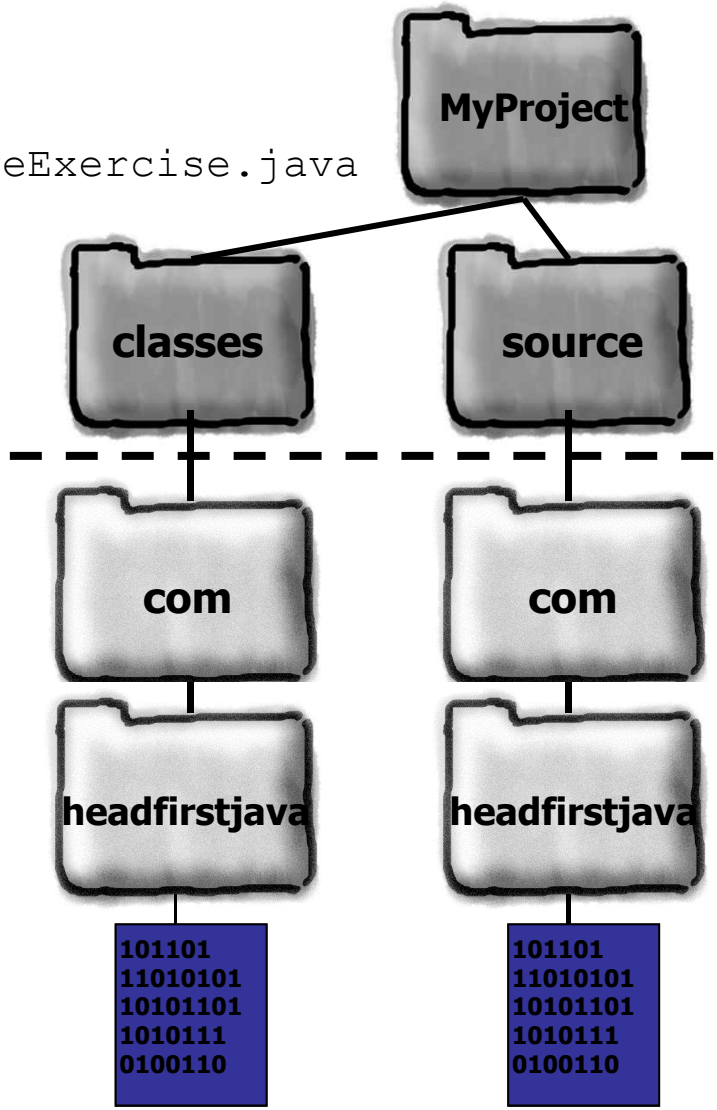
패키지와 디렉토리 구조


MyProject
com.headfirstjava.PackageExercise
classes
source
com
com
패키지 구조
headfirstjava
headfirstjava
101101
11010101
10101101
1010111
0100110
101101
11010101
10101101
1010111
0100110
PackageExercise.class
PackageExercise.java

-d 컴파일 옵션

```
%cd MyProject/source
%javac -d ../classes com/headfirstjava/PakcageExercise.java
```

```
%javac -d ../classes com/headfirstjava/*.java
```



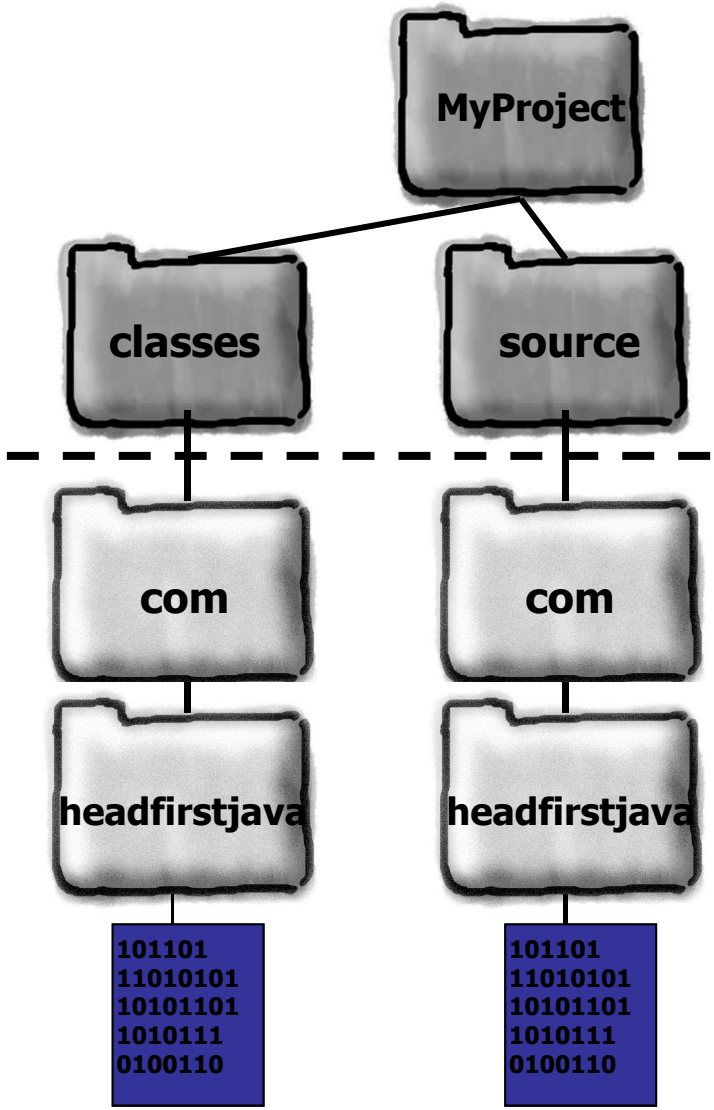
- 실행 방법

```
%cd MyProject/classes
%java com.headfirstjava.PakcageExercise
```

-d 컴파일 옵션

-d 옵션을 사용하면 사용자가 classes 디렉토리 쪽에 하위디렉토리를 따로 만들지 않아도 컴파일러가 자동으로 디렉토리를 만들어서 컴파일한 결과를 저장해줍니다.

```
%javac -d ../classes com/headfirstjava/*.java
```



실행 가능한 JAR(패키지 구조)

1. 클래스 파일이 classes 디렉토리 밑에 제대로 된 패키지 구조로 저장되어있는지 확인

2. main() 메소드가 들어있는 클래스를 지정하기 위한 manifest.txt 파일 만들기 (이 파일은 classes 디렉토리에 집어넣음)

```
Main-Class: com.headfirstjava.PackageExercise
```

3. jar 도구를 실행시켜서 패키지 디렉토리와 manifest 파일이 들어있는 JAR 파일 만들기

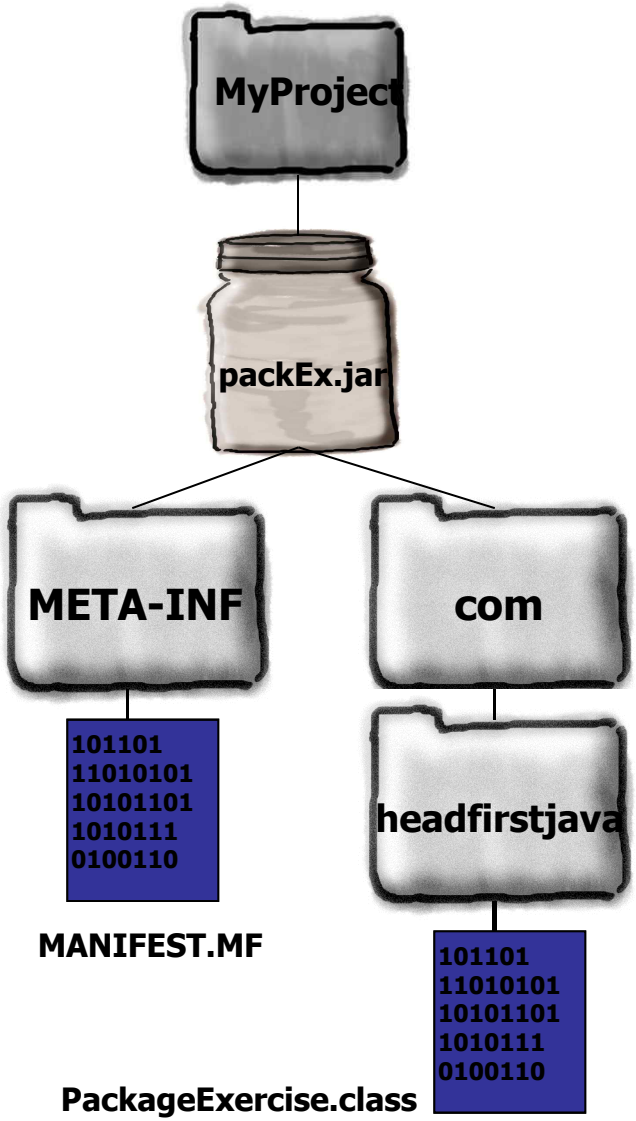
```
%cd MyProject/classes
%jar -cvmf manifest.txt packEx.jar com
```

manifest 파일은?

```
% jar -tf packEx.jar
```

```
% cd Skyler
% jar -tf packEx.jar
META-INF/
META-INF/MANIFEST.MF
com/
com/headfirstjava/
com/headfirstjava/PackageExercise.class
```

```
% cd Skyler
% jar -xf packEx.jar
```



자바 웹 스타트

- **자바 웹 스타트 (Java Web Start; JWS)**
  - 처음에는 웹 브라우저를 통해서 실행되지만 그 후로는 거의 독립형 애플리케이션처럼 실행됨
  - 서버에 있는 애플리케이션이 조금이라도 변경되면 자동으로 갱신된 코드를 내려받고 기존 코드에 통합시켜주는 기능이 있음
  - JWS나 자바 런타임 환경(JRE)을 설치하는 과정을 매우 쉽게 처리할 수 있음
  - 무엇보다도 최종 사용자는 웹 페이지에서 링크만 클릭하면 프로그램을 실행시킬 수 있기 때문에 사용하기가 매우 쉽다는 장점이 있음

JWS 작동 원리

1. 클라이언트에서 JWS 애플리케이션(.jnlp 파일)에 대한 웹 페이지 링크를 클릭
2. 웹 서버에서 .jnlp 파일을 보내줌
3. 브라우저에서 JWS를 시작함. JWS 보조 애플리케이션에서 .jnlp 파일을 읽은 다음 서버에 MyApp.jar 파일을 요구
4. 웹 서버에서 요청받은 .jar 파일을 제공
5. JWS에서 JAR를 받아서 지정된 main() 메소드를 호출하여 애플리케이션을 시작

.jnlp 파일

- **JNLP**
  - 자바 네트워크 구동 규약(Java Network Launch Protocol)
- **JWS 애플리케이션에서 이 파일을 읽어서 JAR 파일을 찾아내고 그 애플리케이션을 시작하기 위한 용도로 사용함**
- **XML 문서로, 여러 내용을 집어넣을 수 있습니다.**

JWS 애플리케이션 제작 및 배포 방법

1. 실행가능한 JAR 만들기
2. .jnlp 파일 만들기
3. JAR와 .jnlp 파일 웹 서버에 저장
4. 웹 서버에 새로운 mime 유형 추가

```
application/x-java-jnlp-file
```

5. .jnlp 파일에 대한 링크가 들어있는 웹 페이지 만들기

```
<HTML>
  <BODY>
    <a href="MyApp2.jnlp">Launch My Application</a>
  </BODY>
</HTML>
```