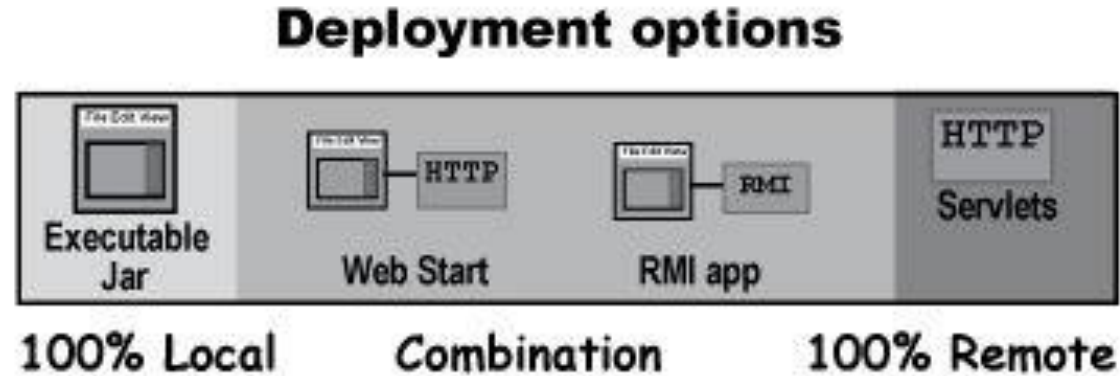


Package, Jars and Deployment - Release Your Code

Samkeun Kim <skim@hknu.ac.kr>

<http://cyber.hankyong.ac.kr>

Deploying Your Application



1. Local

전체 애플리케이션이 엔드 유저 컴퓨터에서 실행가능한 JAR 파일로 배치되어 stand-alone 프로그램으로 실행된다.

2. Combination of local and remote

애플리케이션이 사용자의 로컬 시스템에서 실행되는 클라이언트 부분과 함께 배포되며, 애플리케이션의 다른 부분이 실행되는 서버에 연결되어 있다.

3. Remote

전체 Java 애플리케이션이 서버 시스템에서 실행되며, 클라이언트는 Java가 아닌 다른 수단(예: 웹 브라우저)을 통해 시스템에 액세스한다.

자바 브라우저 플러그인 퇴출

오라클 발표

- JDK 9: 자바 SE 버전의 참조 구현체(Reference Implementation)
- JDK 9에서 자바 브라우저 플러그인 제외
- 대부분의 웹 브라우저 => 자바 브라우저 플러그인을 지원하지 않음

오라클이 제시한 주요 대안

- 자바 애플릿을 **자바 Web Start 애플리케이션**으로 전환
(애플리케이션을 브라우저 플러그인 없이 웹에서 바로 실행할 수 있게 해준다)

소스 코드와 클래스 파일 분리

Compiling with the -d(directory) flag

```
% mkdir MyProject\source
```

```
% mkdir MyProject\classes
```

Java 소스 코드(.java)를 source 디렉토리에 복사한다.

```
% cd MyProject\source
```

```
% javac -d ..\classes MyApp.java
```

-d flag를 사용함으로써 컴파일된 코드가 어느 디렉토리에 들어갈지 결정

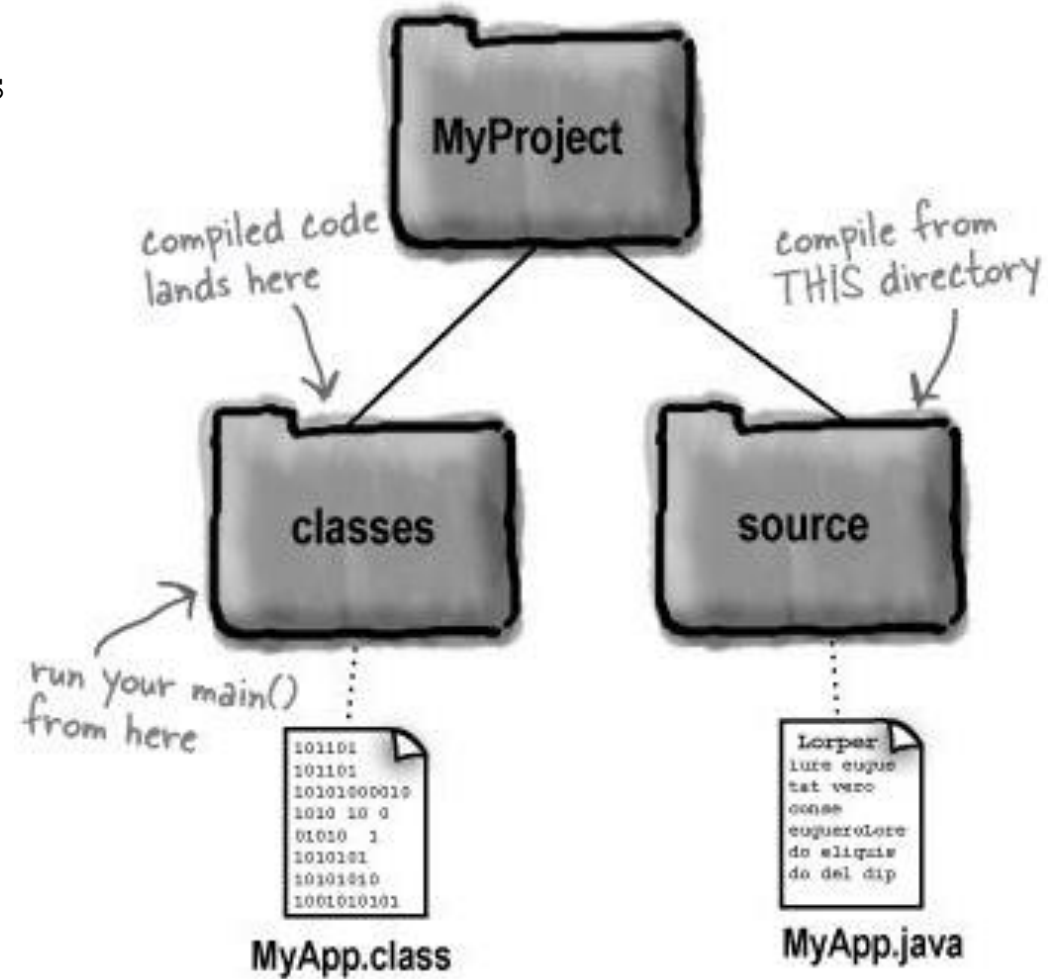
모든 Java 파일을 컴파일하려면,

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

Running your code

```
% cd MyProject\classes
```

```
% java MyApp
```



Put your Java in a JAR



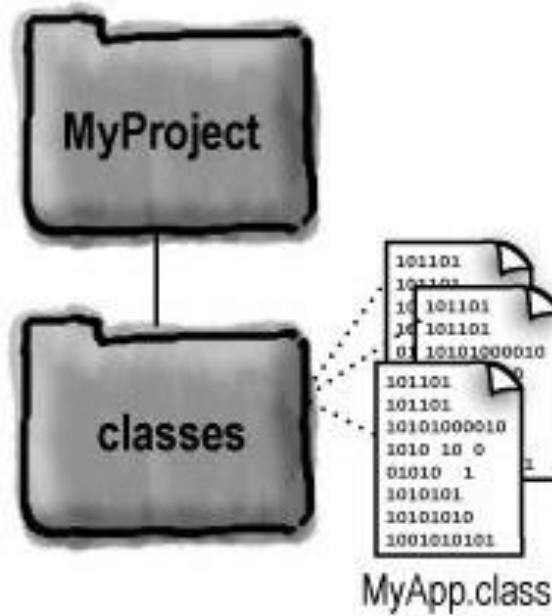
JAR: Java ARchive

사용자가 JAR를 가지고 무엇을 할 수 있나? 어떻게 실행할 수 있나?

⇒ JAR를 실행 가능(executable)하게 만들자!

Making an executable JAR

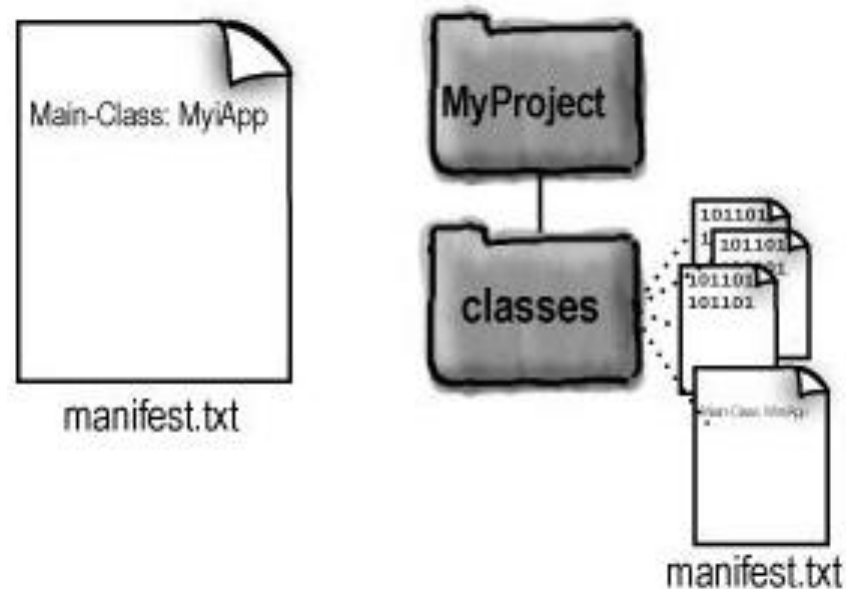
1. 모든 클래스 파일이 classes 디렉토리에 있는지 확인하자.



2. 어떤 클래스에 main() 메소드가 있는지 명시하는 manifest.txt 파일을 만든다.

Main-Class: MyApp ← don't put the .class
on the end

Main-Class 행을 입력한 후 **return 키**를 누른다. Return 키를 누르지 않으면 동작하지 않을 수 있다.
Manifest 파일을 "classes" 디렉토리에 넣는다.

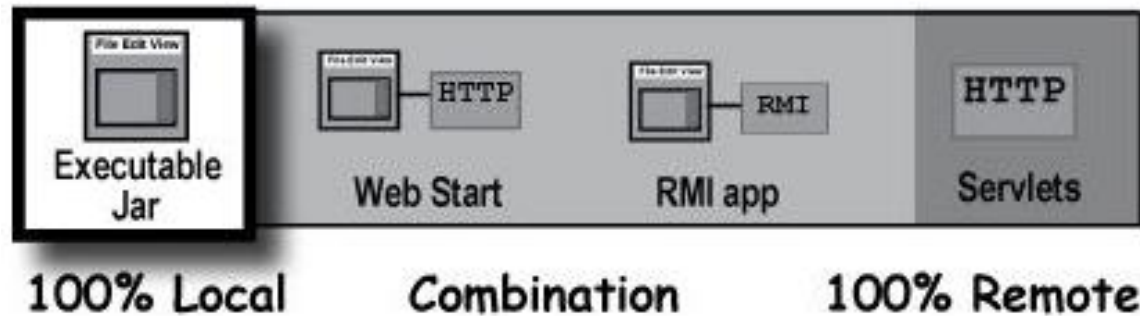
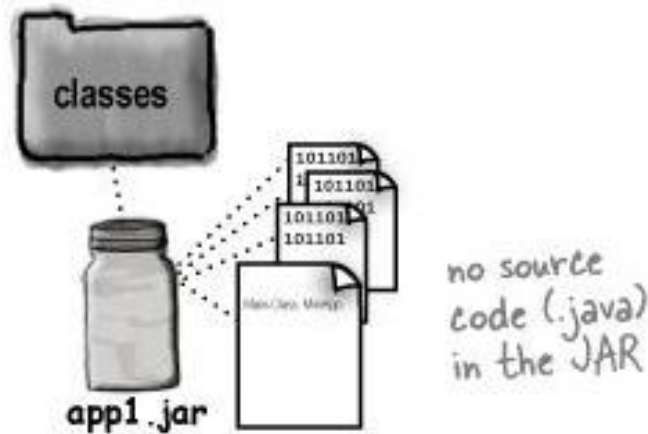


3. Jar 도구를 실행하여 classes 디렉토리의 모든 항목과 manifest가 포함된 JAR 파일을 만든다.

```
% cd MyProject\classes
```

```
% jar -cvmf manifest.txt app1.jar *.class or
```

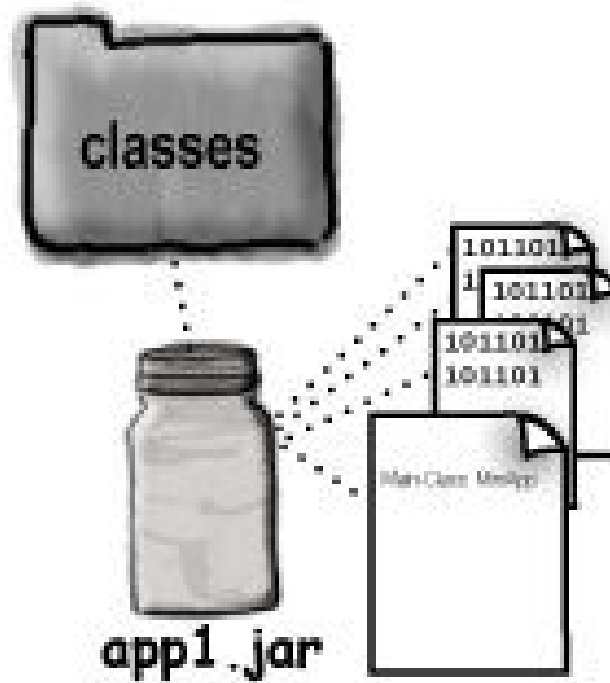
```
% jar -cvmf manifest.txt app1.jar MyApp.class
```



Running(executing) the JAR

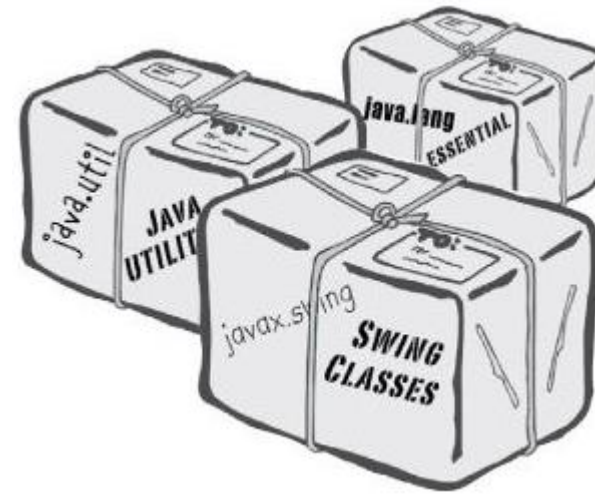
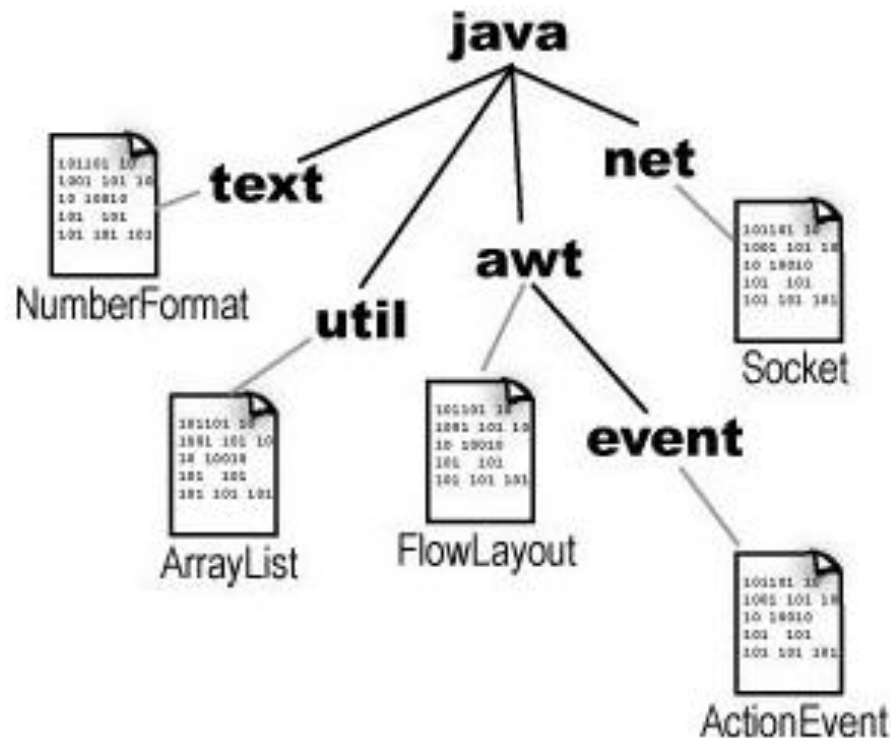
```
% cd MyProject\classes
```

```
% java -jar app1.jar
```



Put your classes in packages!

Packages prevent class name conflicts



`com.headfirstbooks.Book`

package name

class name

...so I finally settled on
foo.bar.Heisenberg for my
quantum baking class

Why, that's the same name
I was thinking of for my
sub-atomic ironing class!
Guess I'll just have to come
up with something else.



Preventing package name conflicts

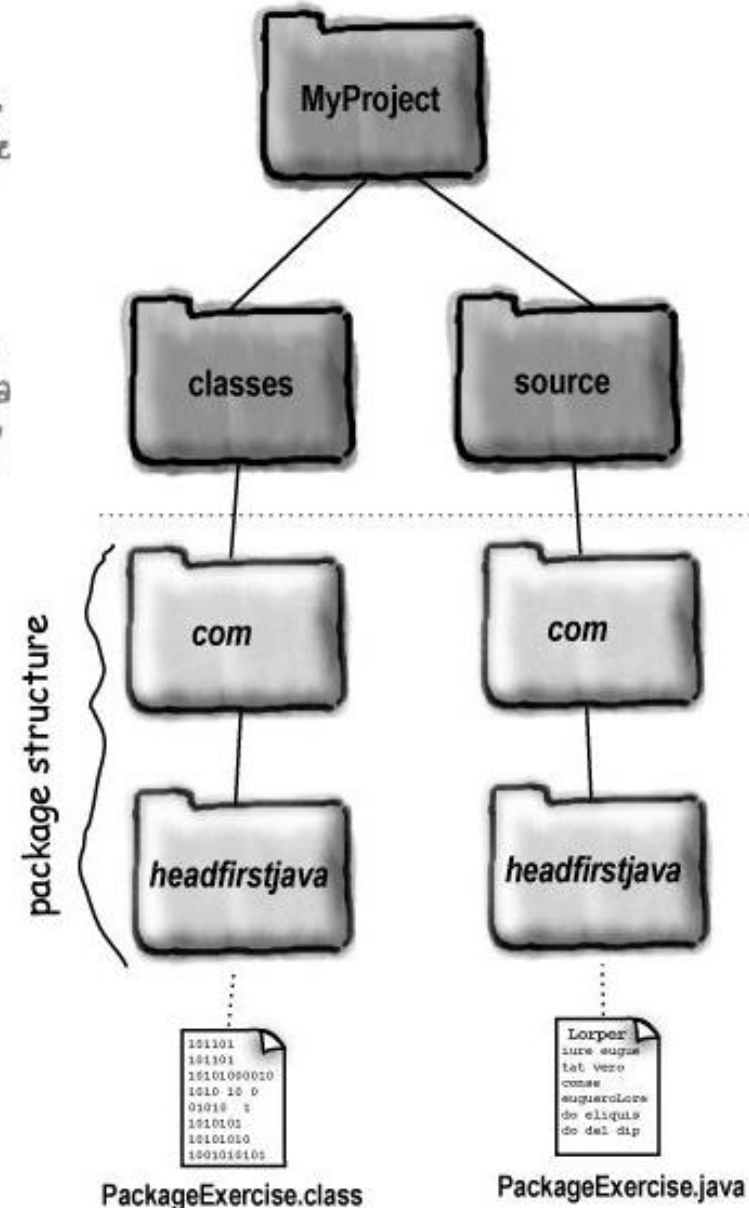
Reverse domain package names

`com.headfirstjava.projects.Chart`

start the package with your reverse domain, separated by a dot (.), then add your own organizational structure after that

projects.Chart might be a common name, but adding com.headfirstjava means we have to worry about only our own in-house developers.

the class name is always capitalize



Compiling and running with packages

Compiling with the -d(directory) flag

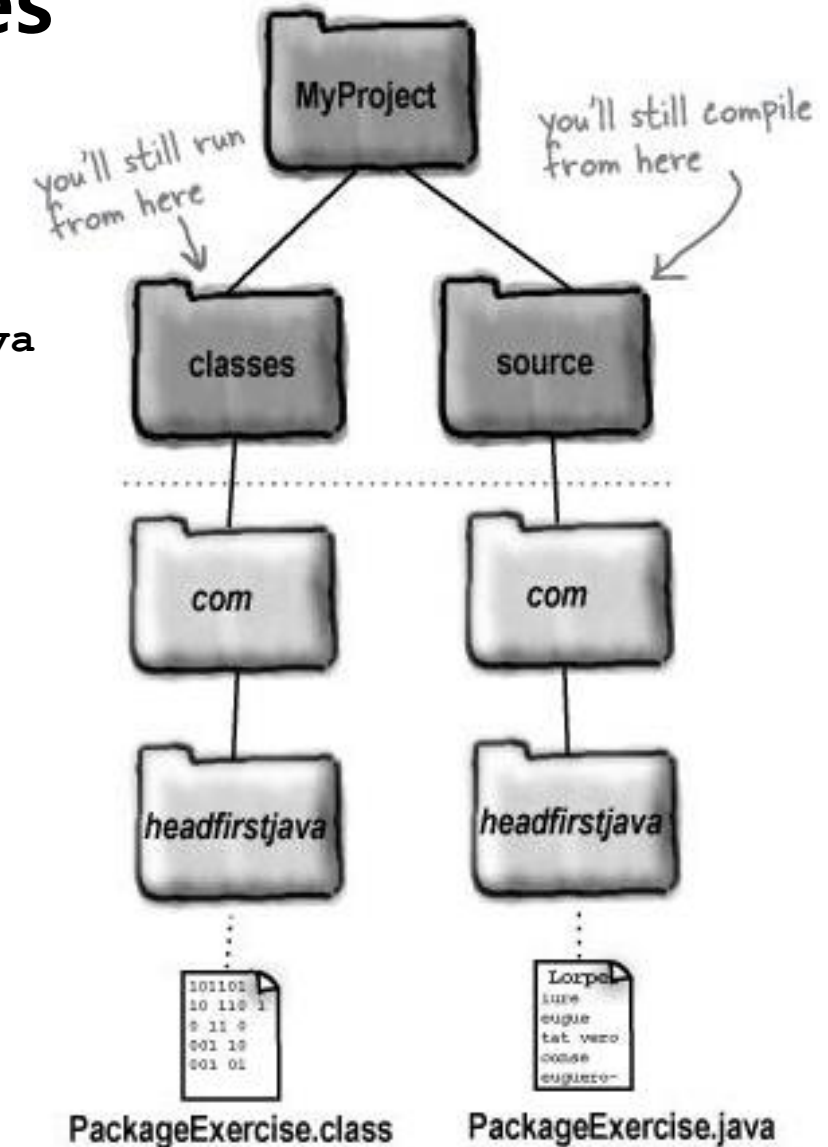
```
% cd MyProject\source
% javac -d ..\classes com\headfirstjava\PackageExercise.java
```

com.headfirstjava 패키지의 모든 .java 파일을 컴파일하려면,
아래와 컴파일한다:

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

Running your code

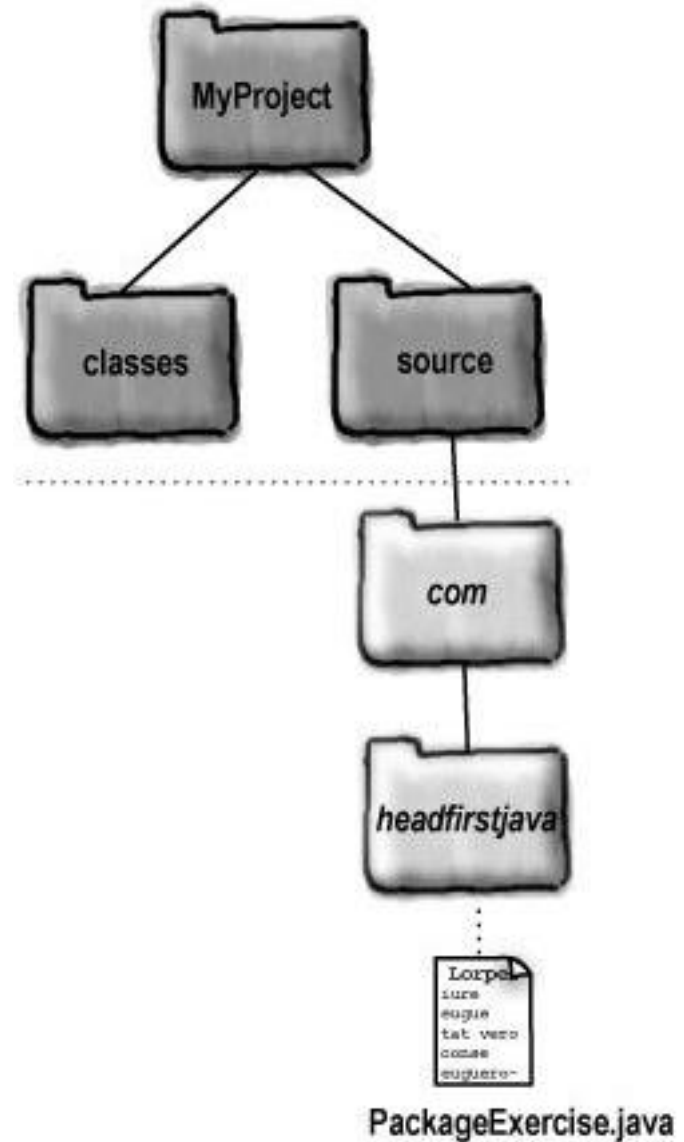
```
% cd MyProject\classes
% java com.headfirstjava.PackageExercise
```



The -d flag is even cooler than we said

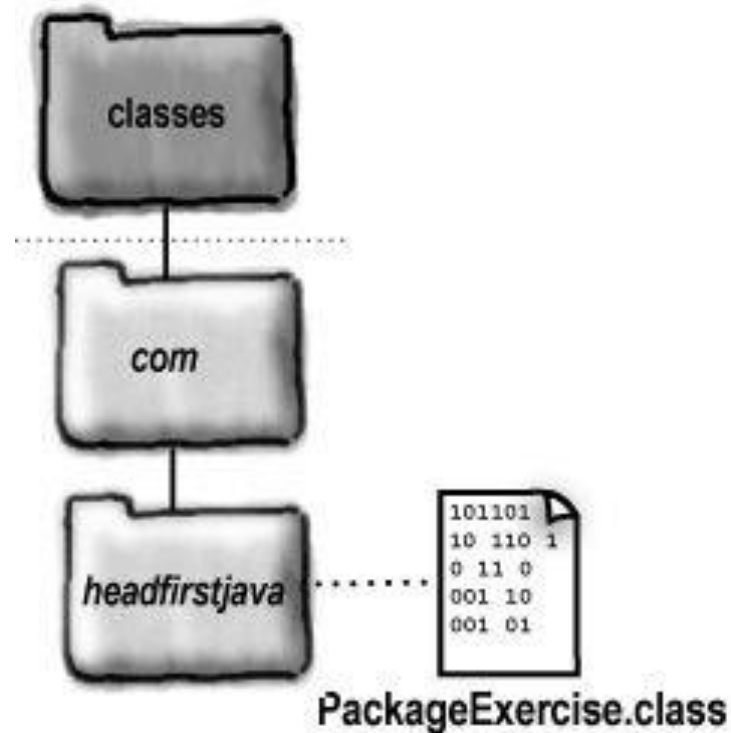
-d 플래그:

컴파일된 클래스 파일을 소스 파일이 있는 곳과 다른 디렉토리로 보낼 수 있을 뿐만 아니라, 클래스가 들어 있는 패키지의 정확한 디렉토리 구조에 클래스를 넣을 수 있다.



Making an executable JAR with packages

1. 모든 클래스 파일이 classes 디렉토리 아래의 올바른 패키지 구조 내에 있는지 확인하자.



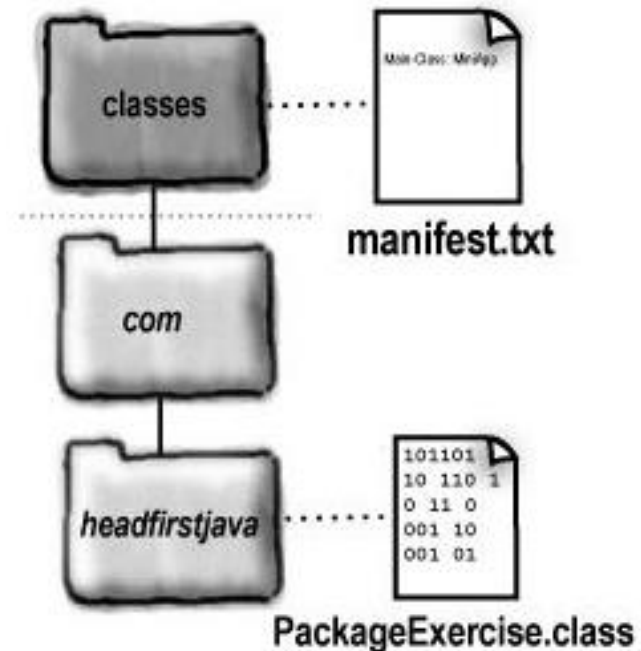
2. 어떤 클래스에 `main()` 메소드가 있는지 명시해 주는 `manifest.txt` 파일을 만든다.
이때 전체 클래스 이름을 사용해야 한다.

한 줄로 된 `manifest.txt` 파일을 만든다:

Main-Class: com.headfirstjava.PackageExercise

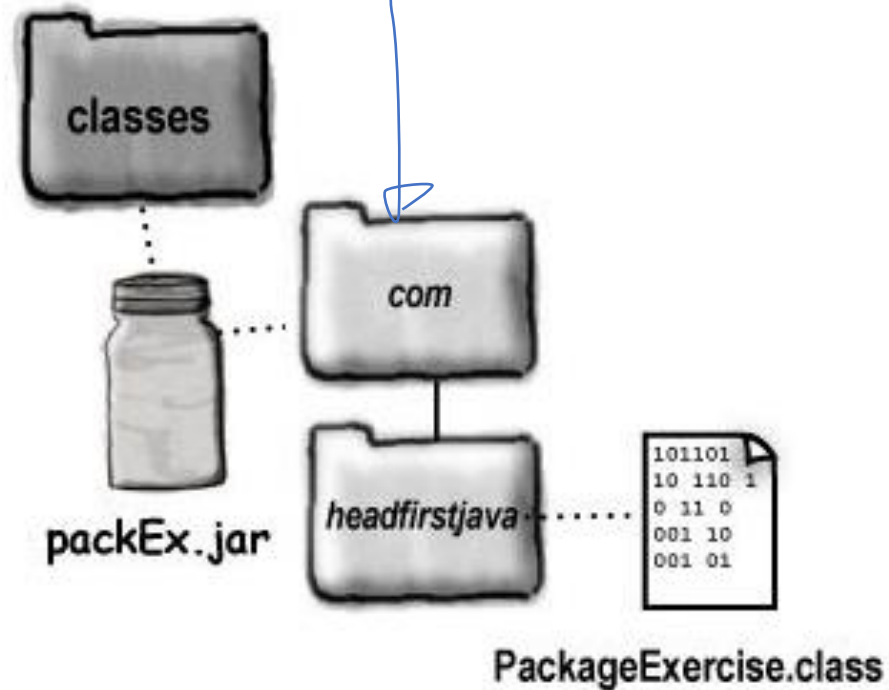
✓ 라인 마지막에서 반드시 **return** 키를 누른다.

Manifest 파일을 classes 디렉토리에 넣는다.



3. Jar 도구를 실행하여 package 디렉토리와 manifest가 포함된 JAR 파일을 만든다.
The only thing you need to include is the 'com' directory, and the entire package (and all classes) will go into the JAR.

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



So where did the manifest file go?

JAR를 들여다 보면 어떤 모양일까?

- ✓ 커맨드라인에서, jar 도구는 JAR를 만들고 실행하는 것 이상의 기능을 수행할 수 있다.
- ✓ JAR의 내용을 추출할 수 있다('unzipping' 또는 'untarring' 처럼).

packEx.jar를 Skyler라는 디렉토리에 넣었다고 상상해보자.

jar commands for listing and extracting

1. List the contents of a JAR

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

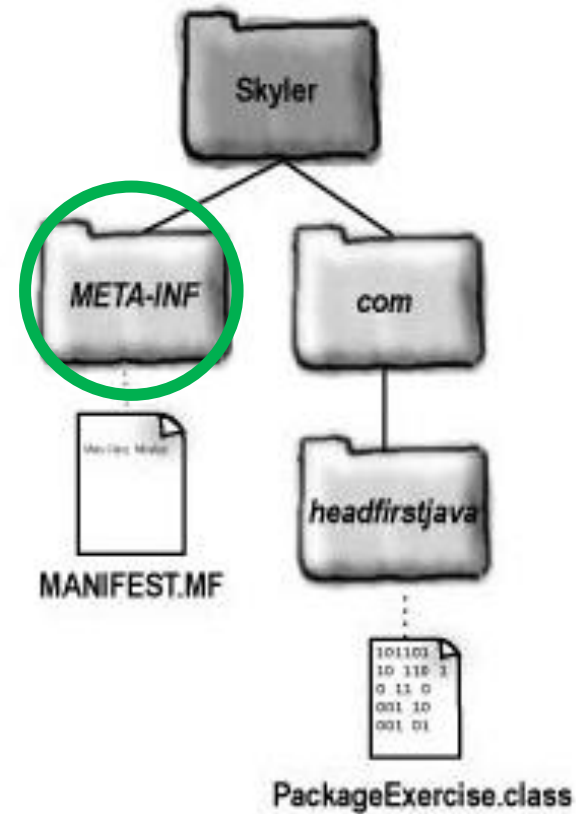
↑ -tf stands for 'Table File' as in
"show me a table of the JAR file"

```
File Edit Window Help Pickle
% cd Skyler
% jar -tf packEx.jar
META-INF/
META-INF/MANIFEST.MF
com/
com/headfirstjava/
com/headfirstjava/
PackageExercise.class
```

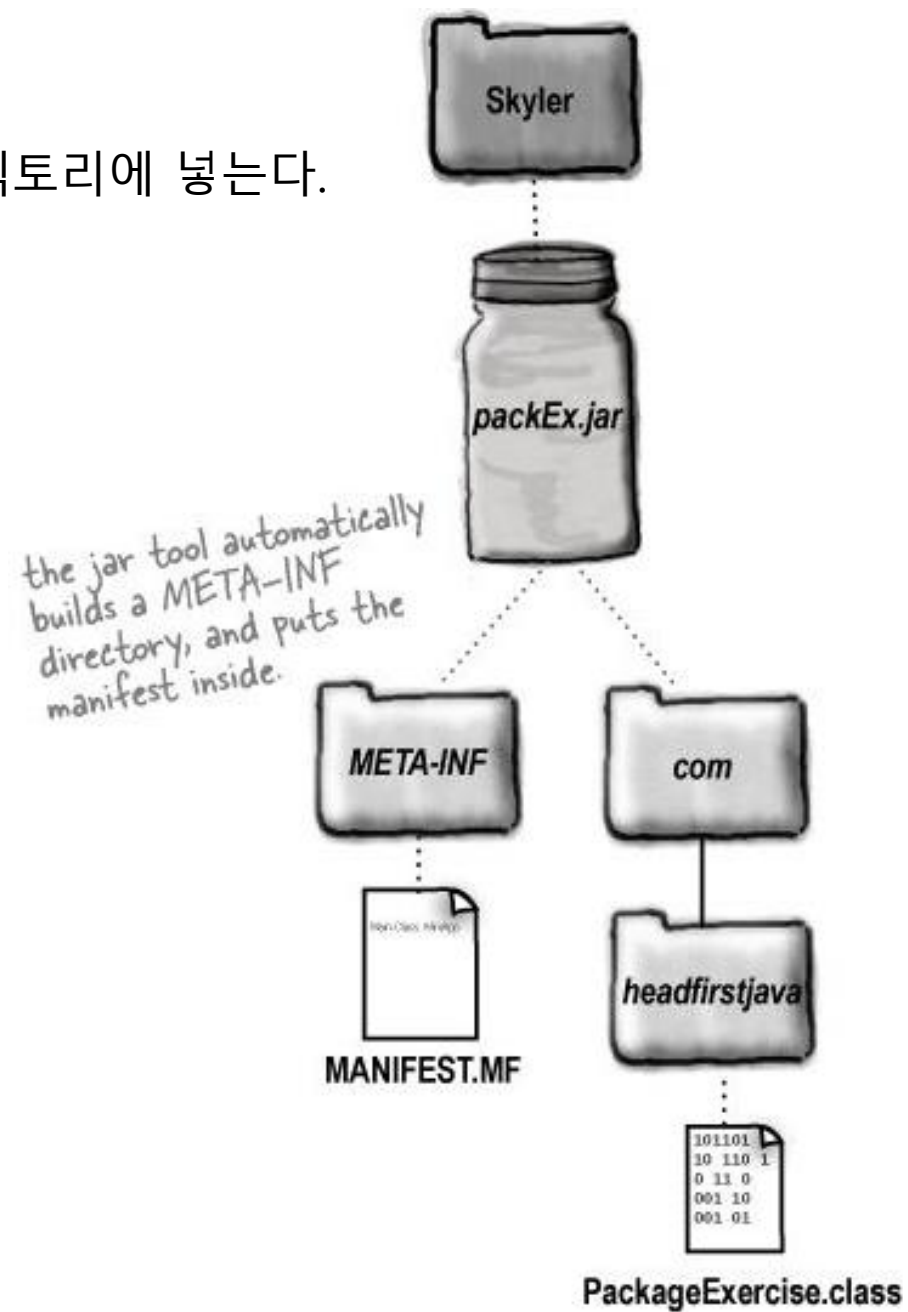
2. Extract the contents of a JAR(i.e. unjar)

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

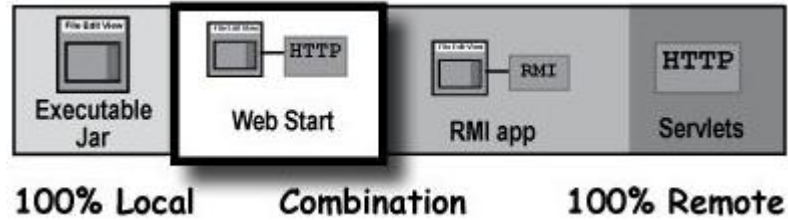
↑
-xf stands for 'Extract File' and it works just like unzipping or untarring. If you extract the packEx.jar, you'll see the META-INF directory and the com directory in your current directory



JAR 파일을 skyer라는 디렉토리에 넣는다.



Java Web Start

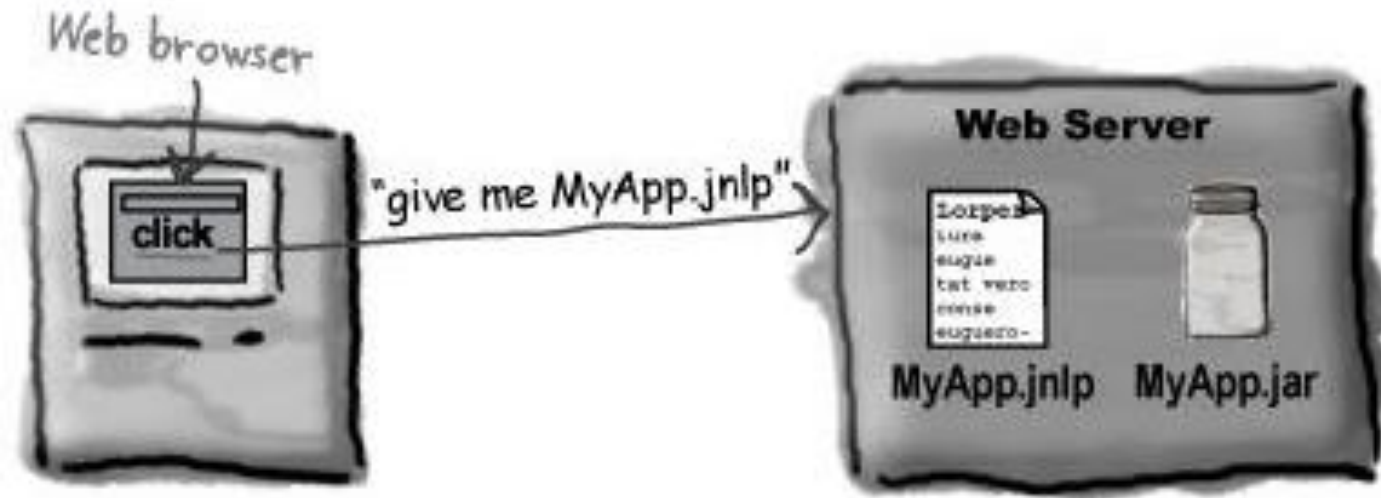


Java Web Start 'helper app'

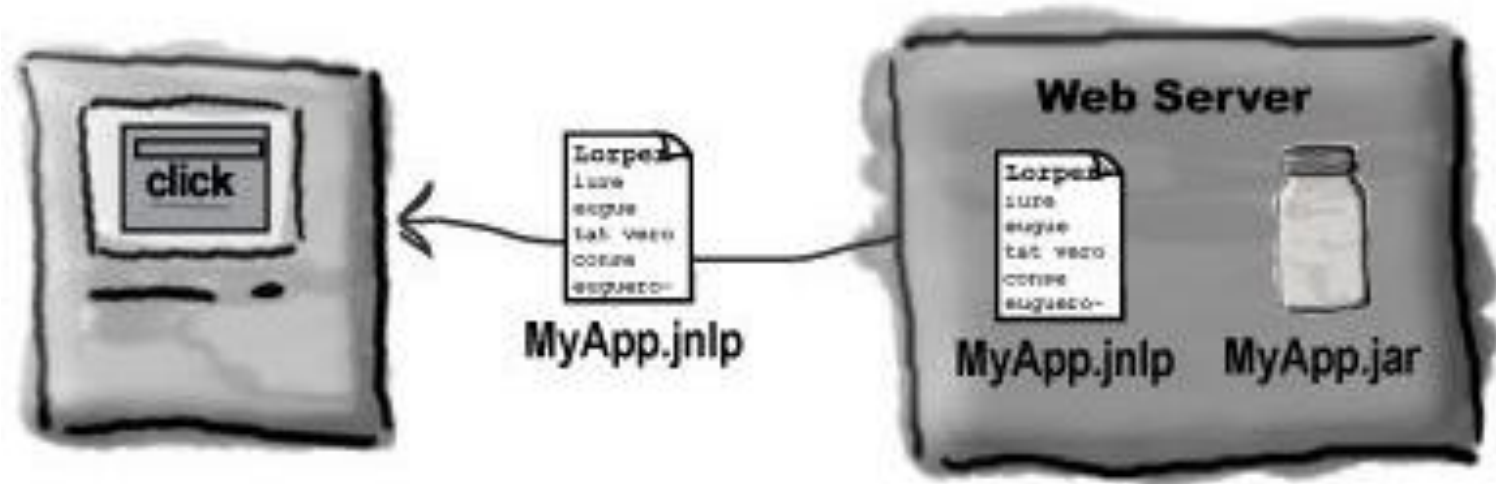
How Java Web Start works

1. 클라이언트가 JWS 응용 프로그램 (.jnlp 파일)에 대한 웹 페이지 링크를 클릭한다:

```
<a href="MyApp.jnlp">Click</a>
```



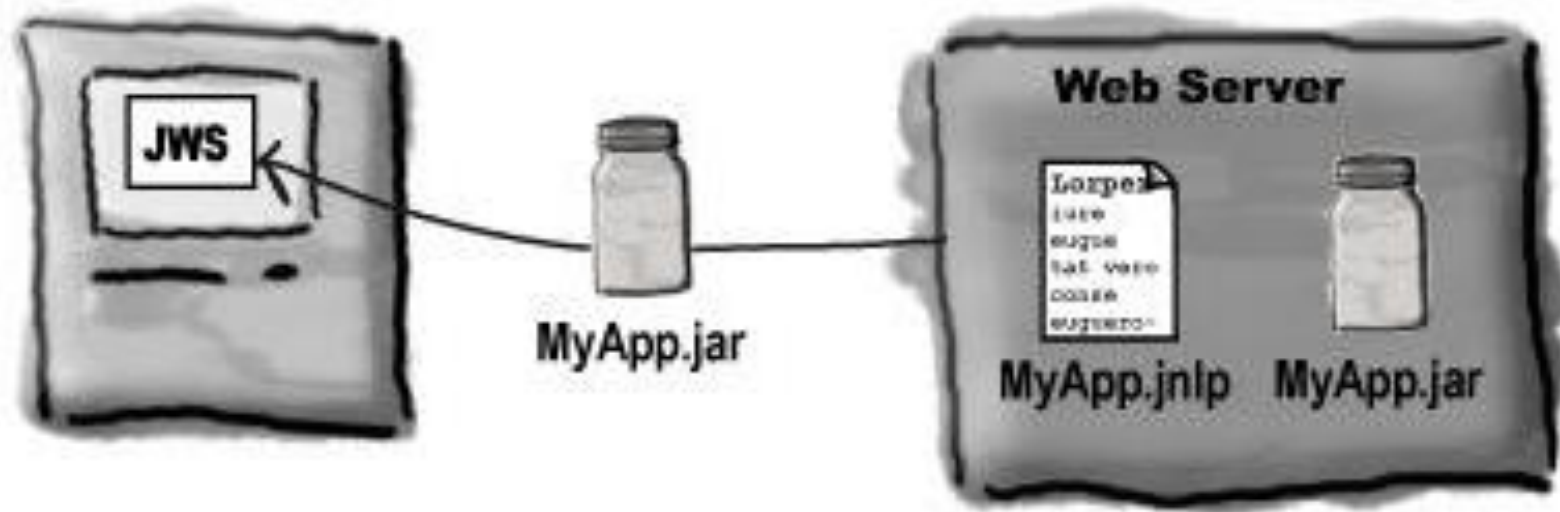
2. 웹 서버(HTTP)가 요청을 받아서 .jnlp 파일을 클라이언트에게 보낸다.
.jnlp 파일은 애플리케이션의 실행 가능한 JAR 파일 이름을 나타내는 XML 문서이다.



3. Java Web Start (small 'helper app' on the client)가 브라우저에 의해 시작된다.
JWS helper app은 .jnlp 파일을 읽어서 서버에 MyApp.jar 파일을 요청한다.



4. 웹 서버는 요청된 .jar 파일을 '제공'한다.



5. Java Web Start는 JAR 파일을 가져와서 지정된 main() 메소드를 호출하여 애플리케이션을 시작한다(실행 가능한 JAR와 동일).

다음 번에 사용자가 이 애플리케이션을 실행하고 싶을 때, Java Web Start 애플리케이션을 열 수 있고, 거기에서 온라인 상태가 아니어도 애플리케이션을 시작할 수 있다.

HelloWebStart (the app in the JAR)



The .jnlp file

Java Web Start 애플리케이션을 만들려면 애플리케이션을 설명하는 .jnlp(Java Network Launch Protocol) 파일을 만들어야 한다.

이 파일은 JWS 애플리케이션이 읽어서 JAR를 찾아 애플리케이션을 시작하는데 사용하는 파일이다.

.jnlp 파일은 아래와 같은 내용을 포함하는 간단한 XML 문서이다:

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<jnlp spec="0.2 1.0"
```

```
  codebase="http://127.0.0.1/~kathy"
```

```
  href="MyApp.jnlp">
```

The 'codebase' tag is where you specify the 'root' of where your web start stuff is on the server. We're testing this on our localhost, so we're using the local loopback address "127.0.0.1". For web start apps on our internet web server, this would say, "http://www.wickedlysmart.com"

This is the location of the .jnlp file relative to the codebase. This example shows that MyApp.jnlp is available in the root directory of the web server, not nested in some other directory.

`<information>`

`<title>kathy App</title>`

`<vendor>Wickedly Smart</vendor>`

`<homepage href="index.html"/>`

`<description>Head First WebStart demo</description>`

`<icon href="kathys.gif"/>`

`<offline-allowed/>`

`</information>`

Be sure to include all of these tags, or your app might not work correctly! The 'information' tags are used by the JWS helper app, mostly for displaying when the user wants to relaunch a previously-downloaded application.

← This means the user can run your program without being connected to the internet. If the user is offline, it means the automatic-updating feature won't work.

`<resources>`

`<j2se version="1.3+"/>`

← This says that your app needs version 1.3 of Java, or greater.

`<jar href="MyApp.jar"/>`

← The name of your executable JAR! You might have other JAR files as well, that hold other classes or even sounds and images used by your app.

`</resources>`

`<application-desc main-class="HelloWebStart"/>`

`</jnlp>`

← This is like the manifest Main-Class entry... it says which class in the JAR has the main() method.

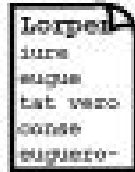
Steps for making and deploying a Java Web Start app

1. Make an executable JAR for your application.



MyApp.jar

2. Write a .jnlp file.



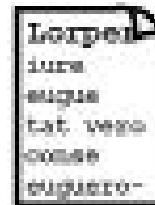
MyApp.jnlp

3. Place your JAR and .jnlp files on your Web server.



4. Web server.application/x-java-jnlp-file에 새로운 mime 타입을 추가한다.
5. .jnlp 파일에 대한 링크가 있는 웹 페이지를 만든다.

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



MyJWSApp.html

Java Web Start demo

자바 웹 스타트 애플리케이션 만들기

작업 디렉토리: miniplayer

```
miniplayer> mkdir source\chap12
```

```
miniplayer> mkdir classes
```

source\chap12 디렉토리에 MiniMusicPlayer3.java 복사

MiniMusicPlayer JAR 만들기

1. MiniMusicPlayer3.java 컴파일

```
miniplayer> cd source
```

```
~source> javac -d ../classes chap12\MiniMusicPlayer3.java
```

(컴파일된 클래스가 classes 폴더에 들어간다)

2. classes 폴더에 manifest.txt 파일 생성(main() 메소드를 포함하는 클래스 지정)

✓ 아래처럼 한 줄로 된 텍스트 파일을 만든다:

Main-Class: chap12.MinMusicPlayer3 [엔터 키] <- 반드시!!

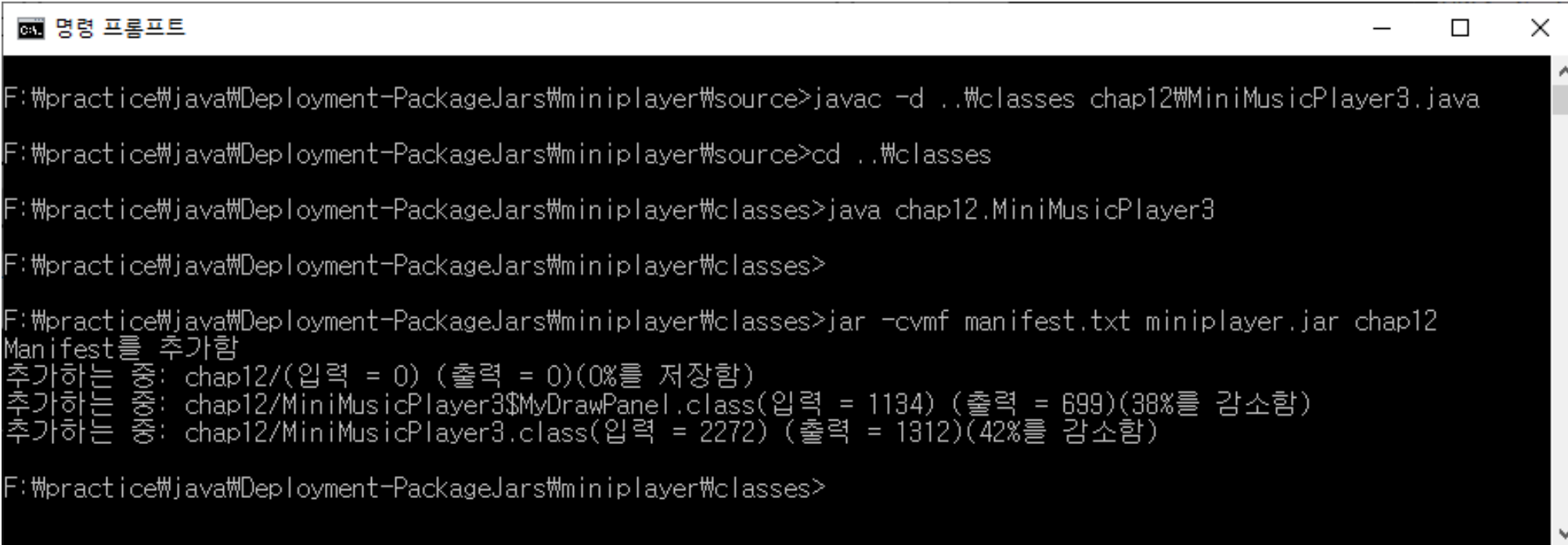
반드시 마지막에 엔터 키를 쳐야 한다

3. Jar 툴을 이용하여 클래스 파일들 + manifest 파일을 포함하는 JAR 파일을 만든다

```
~source> cd ../classes
```

```
miniplayer\classes> java chap12.MinimusicPlayer3
```

```
miniplayer\classes> jar -cvmf manifest.txt miniplayer.jar chap12
```



```
명령 프롬프트
F:\practice\java\Deployment-PackageJars\miniplayer\source> javac -d ../classes chap12\MiniMusicPlayer3.java
F:\practice\java\Deployment-PackageJars\miniplayer\source> cd ../classes
F:\practice\java\Deployment-PackageJars\miniplayer\classes> java chap12.MinimusicPlayer3
F:\practice\java\Deployment-PackageJars\miniplayer\classes>
F:\practice\java\Deployment-PackageJars\miniplayer\classes> jar -cvmf manifest.txt miniplayer.jar chap12
Manifest를 추가함
추가하는 중: chap12/(입력 = 0) (출력 = 0)(0%를 저장함)
추가하는 중: chap12/MiniMusicPlayer3$MyDrawPanel.class(입력 = 1134) (출력 = 699)(38%를 감소함)
추가하는 중: chap12/MiniMusicPlayer3.class(입력 = 2272) (출력 = 1312)(42%를 감소함)
F:\practice\java\Deployment-PackageJars\miniplayer\classes>
```

(참고용) 리스팅과 추출을 위한 jar 커맨드:

```
miniplayer\classes> jar -tf miniplayer.jar
```

명령 프롬프트

```
F:\practice\java\Deployment-PackageJars\miniplayer\classes>jar -tf miniplayer.jar
META-INF/
META-INF/MANIFEST.MF
chap12/
chap12/MiniMusicPlayer3$MyDrawPanel.class
chap12/MiniMusicPlayer3.class
F:\practice\java\Deployment-PackageJars\miniplayer\classes>
```

```
miniplayer\classes> jar -xf miniplayer.jar
```

명령 프롬프트

```
F:\practice\java\Deployment-PackageJars\miniplayer\classes>jar -xf miniplayer.jar
F:\practice\java\Deployment-PackageJars\miniplayer\classes>
```

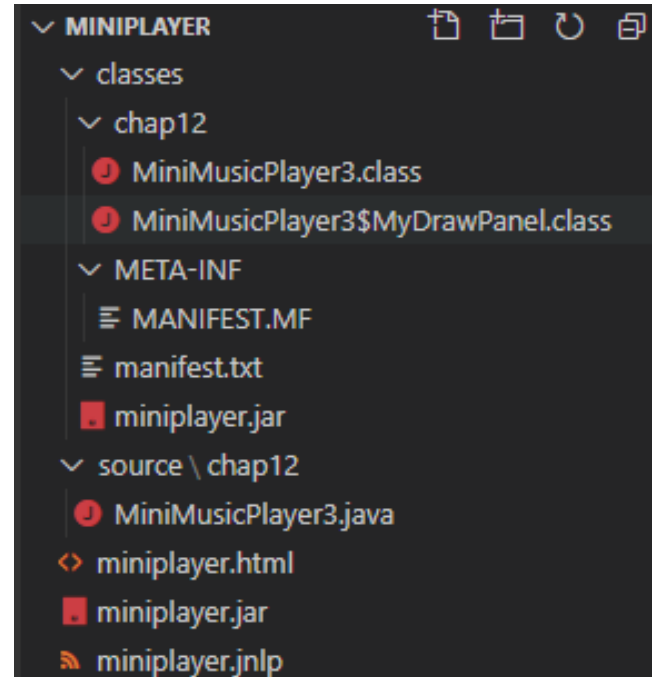
·01 (F:) > practice > java > Deployment-PackageJars > miniplayer > classes

이름	수정한 날짜	유형
chap12	2019-07-09 오후 2:47	파일 폴더
META-INF	2020-06-26 오전 9:57	파일 폴더
manifest.txt	2018-03-31 오전 6:10	텍스트 문서
miniplayer.jar	2020-06-26 오전 9:57	ALZip JAR File

.jnlp 파일 만들기 (miniplayer.jnlp)

miniplayer.jnlp

```
1  <?xml version="1.0" encoding="utf-8"?>
2
3  <jnlp spec="0.2 1.0"
4      codebase="http://ksamkeun.dothome.co.kr/"
5      href="miniplayer/miniplayer.jnlp">
6
7      <information>
8          <title>My MiniMusicPlayer</title>
9          <description>My MiniMusicPlayer Web Start Demo</description>
10         <offline-allowed/>
11     </information>
12
13     <resources>
14         <j2se version="1.3+"/>
15         <jar href="miniplayer/miniplayer.jar"/>
16     </resources>
17
18     <application-desc main-class="chap12.MinimusicPlayer3"/>
19 </jnlp>
```



Web page 만들기 (miniplayer.html)

```
<> miniplayer.html > ...
1  <!DOCTYPE html>
2  <html>
3      <body>
4          <a href="http://ksamkeun.dothome.co.kr/miniplayer/miniplayer.jnlp">
5              Launch My MiniMusicPlayer</a>
6      </body>
7  </html>
```

- MINIPLAYER
 - classes
 - chap12
 - META-INF
 - manifest.txt
 - miniplayer.jar
 - source
 - MiniMusicPlayer3.java
 - <> miniplayer.html
 - miniplayer.jar
 - miniplayer.jnlp

Dothome에 파일 업로드

로컬 사이트: F:\practice\java\Deployment-PackageJars\miniplayer\

- Deployment-PackageJars
 - beatbox
 - beatbox2
 - beatbox3
 - dh-beatbox2
 - dh-beatbox3
 - dh-musicserver2
 - mini-music-player
 - miniplayer
 - musicserver
 - musicserver2

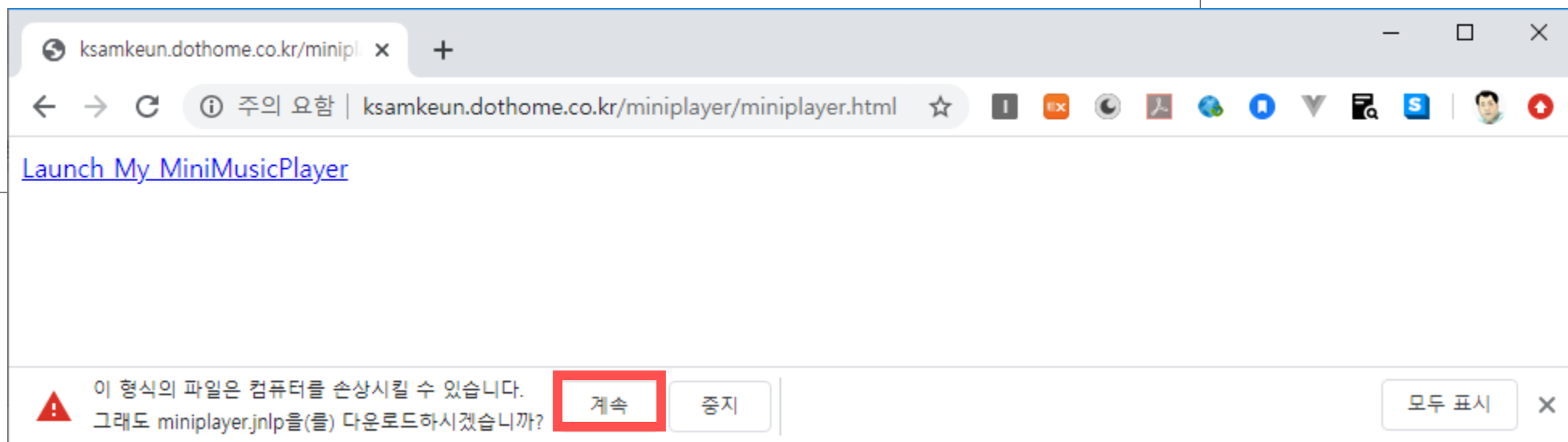
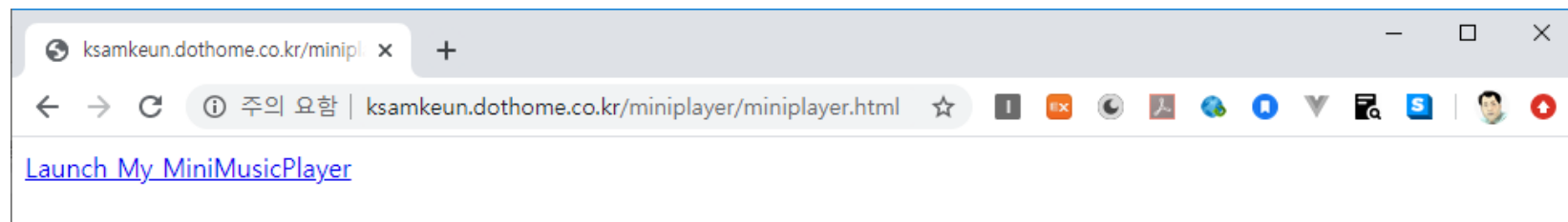
리모트 사이트: /html/miniplayer

- /
 - .gnome2
 - html
 - datasets
 - miniplayer

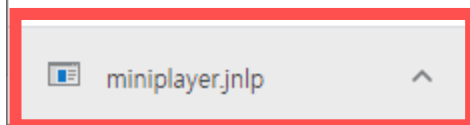
파일명	크기	파일 유형	최종 수정
..			
classes		파일 폴더	2019-07-09 오후 ...
source		파일 폴더	2019-07-09 오후 ...
miniplayer.html	179	Chrome HTML Do...	2018-03-31 오전 ...
miniplayer.jar	2,802	ALZip JAR File	2019-01-04 오후 ...
miniplayer.jnlp	524	JNLP File	2018-03-31 오전 ...

파일명	크기	파일 유형	최종 수정
..			
miniplayer.html	172	Chrome H...	2020-06-26 ...
miniplayer.jar	2,802	ALZip JAR ...	2020-06-26 ...
miniplayer.jnlp	526	JNLP File	2020-06-26 ...

<http://ksamkeun.dothome.co.kr/miniplayer/miniplayer.html>




[Launch My MiniMusicPlayer](#)



Java 보안으로 차단된 애플리케이션

Java 업데이트 필요

 사용자의 Java 버전이 오래되었습니다.

→ 업데이트(권장)
java.com에서 최근 보안 업데이트를 가져옵니다.


→ 차단
이 브라우저 세션에서 Java 콘텐츠가 실행되는 것을 차단함

→ 나중에
계속하다가 나중에 다시 업데이트하도록 알려줍니다.

☐ 다음 업데이트가 사용 가능할 때까지 다시 묻지 않습니다.

Java 애플리케이션 차단

Java 보안으로 차단된 애플리케이션



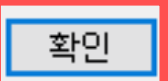
보안을 위해서는 애플리케이션이 높음 또는 매우 높음 보안 설정에 대한 요구 사항을 충족하거나 실행이 허용되는 예외사항 사이트 목록의 일부여야 합니다.

추가 정보(M)

이름: My MiniMusicPlayer

위치: <http://ksamkeun.dothome.co.kr>

이유: 사용자의 보안 설정에서 애플리케이션이 오래되거나 만료된 Java 버전으로 실행되는 것을 차단했습니다.

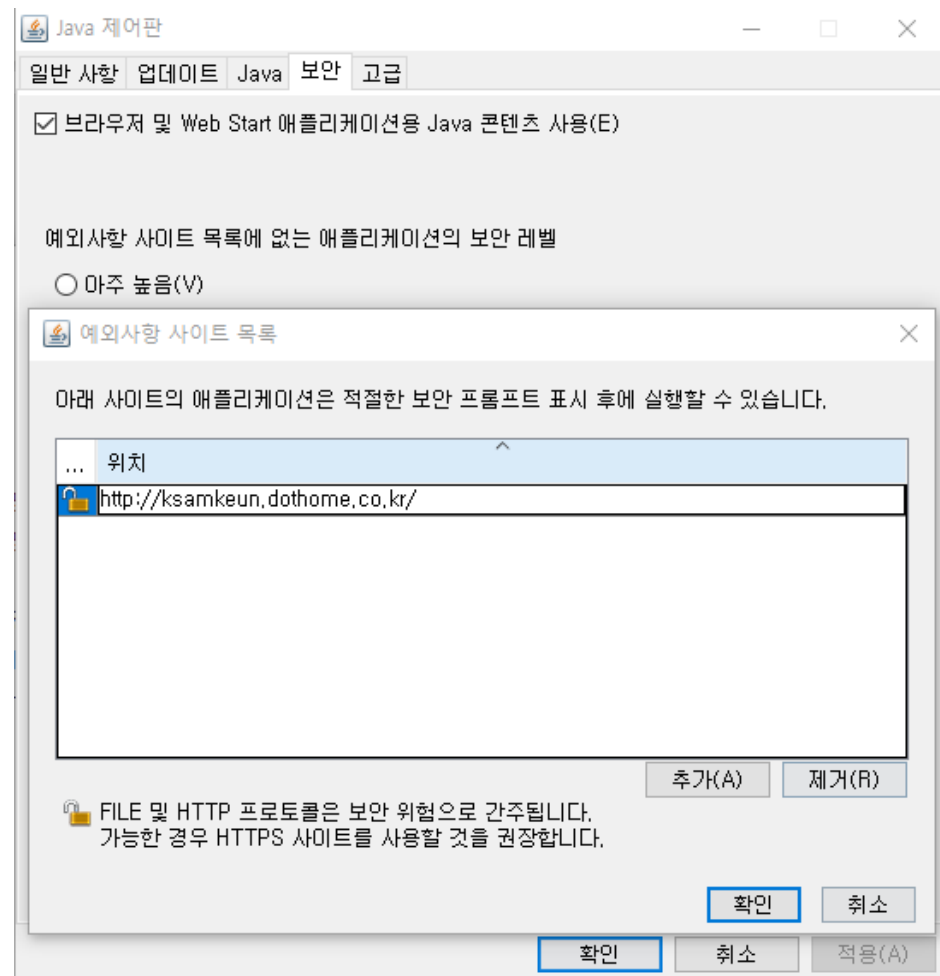


JAR를 웹 브라우저에서 실행하기 전에

제어판 > 프로그램 > Java > 보안 > 예외사항 사이트 목록: 아래 사이트 추가

사이트 예:

<http://YOUR-ID.dothome.co.kr/> (닷홈에서 실행하고자 하는 경우)



보안 경고 - HTTP 위치

예외사항 사이트 목록에 HTTP 위치가 포함되면 보안
위험으로 간주됩니다.



위치: http://ksamkeun.dothome.co.kr

HTTP를 사용하는 위치는 보안 위험이므로 컴퓨터의 개인 정보가 손상될 수 있습니다.
예외사항 사이트 목록에 HTTPS 사이트만 포함할 것을 권장합니다.

이 위치를 수락하려면 [계속]을 누르고, 이 변경을 중단하려면 [취소]를 누르십시오.

계속

취소

Java 제어판

일반 사항 업데이트 Java 보안 고급

☒ 브라우저 및 Web Start 애플리케이션용 Java 콘텐츠 사용(E)

예외사항 사이트 목록에 없는 애플리케이션의 보안 레벨

☐ 아주 높음(V)

신뢰할 수 있는 기관의 인증서로 식별된 Java 애플리케이션만 허용할 수 있으며, 인증서가 철회되지 않은 것으로 확인된 경우에만 가능합니다.

☒ 높음(H)

인증서의 철회 상태를 확인할 수 없더라도 신뢰할 수 있는 기관의 인증서로 식별된 Java 애플리케이션을 실행할 수 있습니다.

예외사항 사이트 목록

아래 사이트의 애플리케이션은 적절한 보안 프롬프트 표시 후에 실행할 수 있습니다.

http://ksamkeun.dothome.co.kr/

사이트 목록 편집(S)...

보안 프롬프트 복원(R)

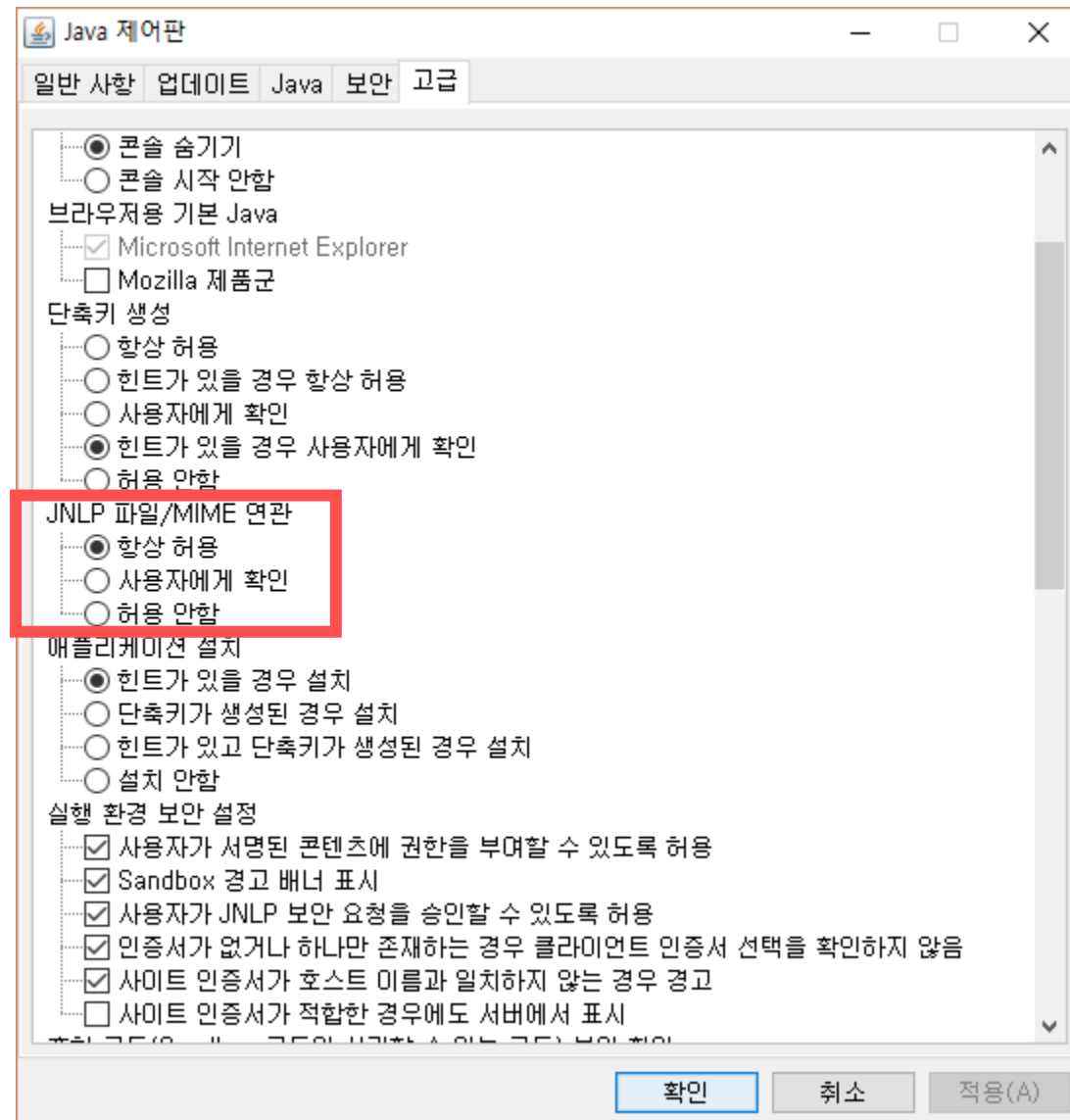
인증서 관리(M)...

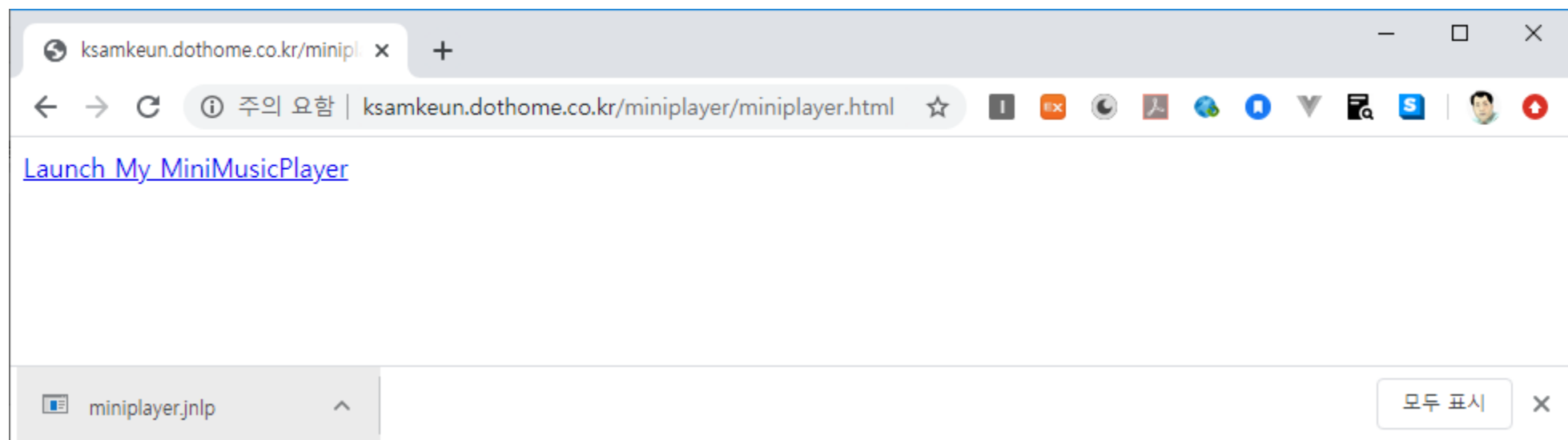
확인

취소

적용(A)

제어판 > 프로그램 > Java > 고급: JNLP 파일/MIME 연관 -> 항상 허용





보안 경고



이 애플리케이션을 실행하겠습니까?



사용자의 Java 버전이 오래되었으며 아래 위치에서 서명되지 않은 애플리케이션이 실행 권한 설정을 요청하는 중입니다.

위치: <http://ksamkeun.dothome.co.kr>

추가 정보(M)

아래 단추를 사용하여 Java를 업데이트할 것을 권장합니다. 이 앱을 중지하려면 [취소]를 누르고, 계속하려면 [실행]을 누르십시오.

실행(R)

업데이트(U)

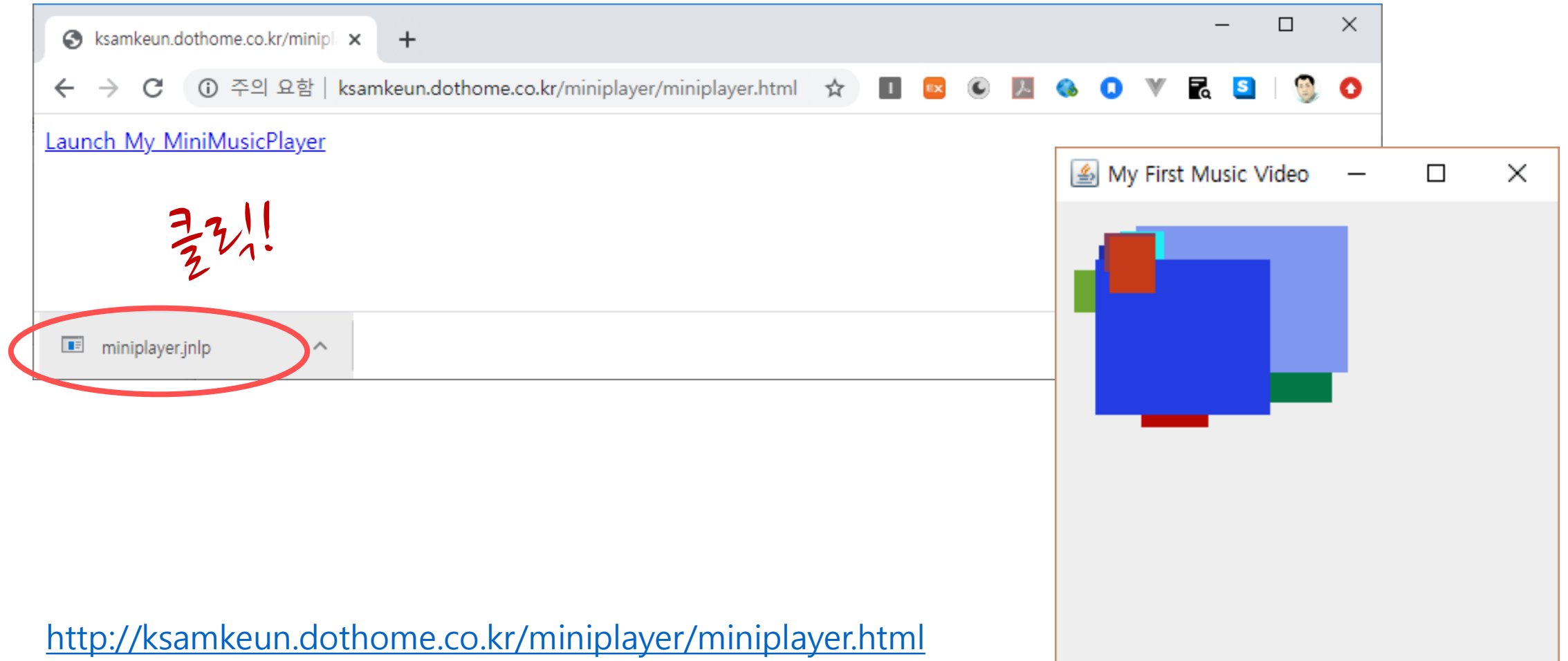
취소



My First Music Video



실습과제 17-1 Browser에서 실행



실습과제 17-2

<https://ksamkeun.000webhostapp.com/slotmachine/slotmachine.html>

실습과제 13-3의 Slot Machine 예제를 Java Web Start Application으로 실행하기

