Jar App / Java Web Start

Java Portability / Jar

Java Program Portability

- Java Runtime Environment JRE -- gives a machine the ability to run any Java program, but the user must have the JRE installed to run a Java program.
- Compile, .java produces set of files like MyProg.class bytecode files
- A ",jar" file is a way of packaging together all the .class files and other files that make up a program
- Someone with the JRE installed, can just double click the .jar to run the program, so the .jar acts like a .exe, except it is portable to any operating system.
- The same .jar file works on different operating systems (Windows, Mac, Linux, ...) without modification -- like a PDF file. The Swing GUI can change its "look and feel" to take on the look of the native operating system.
- Can set up "Java Web Start" web page so that a single click in the browser downloads and runs the .jar program.

Making a .jar File With Eclipse

- If you have your Swing program working in Eclipse, you can package it up in a .jar file with just a few steps.
- 1. Click file->export
- 2. Choose "JAR file", and click Next
- 3. You should see list of open projects. First, uncheck them all. Then use the triangle to open your project. Click the single checkbox next to "default package".
- 4. Still on the same page, click the "Browse" button to select a filename for the .jar, such as "MyGreatProgram.jar" and a location for the .jar file. You may store the .jar file in your project folder, or make a new folder just for it (remember to put copies of your data files, images etc. in the folder with the .jar).
- 5. Click Next, **twice**, to get to the last page.
- 6. On the last page, click Browse to select the main class – the class that contains the main() where the program should begin when double clicked, such as MyFrame.
- 7. Click "finish" to write out the .jar file. There may be some warnings from the compile, but it still produces the .jar. Double clicking the .jar file should launch the program if the JRE (Java Runtime Environment) is installed correctly.
- 8. Make up a distribution folder that contains your .jar file, any data files it needs, and maybe a README file that explains what it does. You are now a famous software author with a growing following! You do not need to include all your source .java files, .class files. etc. in the folder. Note that System.out.println() works when we run a program in Eclipse (i.e. debugging), but not when a normal user runs the program as a .jar. Therefore, the program should communicate with the user by drawing on screen or using JOptionPane, not println().
- With a .jar application, just reading files from the "current directory" is a little fragile (text files, images). There are ways to package data files etc. in the jar file, and read them from there -- so the whole application is a self-contained, portable unit. See the "getResource()" method.

Old -- Applets

- Java "applet" is a Java program that runs inside a web browser page. The applet appears within the browser window.

- Uses Java's portability feature
- Subclass off the Applet class
- Implement the paint() method to draw on screen
- Ultimately, a somewhat fragile system -- combines Java's complexity and the web browser (Internet Explorer, Mozilla, ...) complexity
- Microsoft dislikes Java, so historically Internet Explorer was weak on Applet support

New -- Java Web Start

- Java Web Start (JWS) – a standard part of the Java Runtime Environment (JRE)
- Write a regular Java application, package it in a .jar file
- Write a ".jnlp" file that describes where the .jar is
- Place the .jnlp and the .jar in your web site and create a link to the .jnlp

How Java Web Start Looks to the User

- User has JRE installed (many new computers now have the JRE installed from the factory, such as with Acrobat).
- User goes to page with URL pointing to a .jnlp file
- They click the URL which automatically downloads the .jar file if necessary and then runs the program. Some browsers put up a "confirm what I'm going to do with this .jnlp" dialog.
- JWS has a facility to automatically download newer versions of each app -- a nice way to maintain some utility program used within an organization
- By default, a JWS app does not have access to files on the machine where it runs (i.e. prevent virus/worm security type problems)
- Users can click on a JWS link, confident that it will not mess up their machine. In general, Java has robust anti-virus features built in.
- JWS allows you to deliver a whole application (as opposed to a web app) to the user, but keeping some of the convenience of the web, since to "run" it you just go to a web page.
- The JWS app runs in a "sandbox" that controls what it can touch by, at run time, controlling what system methods it can call.

Sandbox Strategy

- The JWS app runs in a "sandbox" that controls what it can touch by, at run time, controlling what system methods it can call. An exception is thrown if the code calls something it is not supposed to.
- The sandbox strategy works, since the code itself does not have access to buffer overflows etc. to use memory arbitrarily.

e.g. LookNFeel Jar / Java Web Start

- Simple application in http://www.stanford.edu/class/cs108/looknfeel/ (the .java and the .jar are there)
- Can download the ".jar" file and run it
- More usefully, click the Java Web Start link which points to the .jnlp file, and the browser downloads and runs the program from your browser

looknfeel .jnlp file (XML)

```
<?xml version="1.0" encoding="utf-8"?>
<!--  an example .jnlp file for looknfeel.jar -->
<jnlp
        spec="1.0+"  <!-- can be omitted -->

        <!-- URL where things below are found -->
        codebase="http://www.stanford.edu/class/cs108/looknfeel/"

        <!-- the .jnlp file to use -->
        href="looknfeel.jnlp"
```

```
>


<information>
    <title>Look And Feel</title>
    <vendor>Nick Parlante</vendor>
    <homepage href="http://www.stanford.edu/class/cs108/looknfeel/"
    <description kind="one-line">Simple Java Web Start application</description>
    <description kind="short">Demonstrates the flexibility of Swing Look And Feel.</description>

    <!-- could have a little icon
      <icon href="icon.jpeg"/>
    -->

    <!-- this allows the app to be run without a net connection -->
    <offline-allowed/>
</information>

<resources>
        <!-- the "+" is important here, so it works with future Java versions -->
        <j2se version="1.2+"/>


        <!-- the .jar file that contains the code -->
       <jar href="looknfeel.jar" main="true" download="eager" />
</resources>

<!-- what's the main class in the jar -->
<application-desc main-class="LookNFeel"/>

</jnlp>
```