CS 351
Design of Large Programs
Creating Executable Jar

Brooke Chenoweth

University of New Mexico

Spring 2024

Jar Files

- The Java Archive (JAR) file format bundles multiple files into a single archive file.
- Uses ZIP file format
- Contains class files and auxiliary resources.
- May hold a library or standalone application.

Jar Command

- The `jar` command allows you to create and manipulate jar files.
- Common options:

f	Specify jar file name
v	Be more verbose
c	Create a jar file
e	Specify entry point
t	View table of contents
x	Extract files

- `https://docs.oracle.com/javase/tutorial/deployment/jar/index.html`

Creating Executable Jar

1. Compile your classes.
2. Make sure you have all resources needed (sounds, images)
3. Create jar file. If all files needed are in current directory, easiest to use wildcard:

   ```
   jar cvfe MyProgram.jar MyMainClass *
   ```

4. Make sure program runs from the jar.
   Use the `-jar` option with java.

   ```
   java -jar MyProgram.jar
   ```

Extracting Files from Jar

- Use `jar xvf MyProgram.jar` to extract all files.
- Use `jar xvf MyProgram.jar filename` to extract specific file(s).

Loading Resources from Jar

- Regular file operations will be looking for files relative to the current program directory.
- Use ClassLoader to look for files relative to class location (even inside a jar)
- `getClass().getClassLoader()`
  `.getResourceAsStream(resourceFileName)`
  will give an InputStream which you can use in other IO operations.
- Run jar in new location to make sure you are properly loading resources.

Configuring Executable Jar with IntelliJ IDEA

- File → Project Structure...
- Project Settings → Artifacts
- Click Add Icon (green plus)
- Select JAR → From modules with dependencies...
    - Select Main Class
    - Select extract to target JAR
    - Click OK
- Add source code to jar as well as compiled files
    - Click Add Icon on Output Layout
    - Select Directory Content
    - Browse to `src` directory

Building Executable Jar with IntelliJ IDEA

- Build → Build Artifacts
  - Select Build.
  - Once it's done building, the jar file will be located inside of your out folder: `out/artifacts/project_jar/project.jar` (or whatever you named the project and jar)
- If this is a version of the jar that you are submitting, move the file to the top level of your project, add it to git, commit, and push.

Loading Resources from Jar

- Regular file operations will be looking for files relative to the current program directory.
- Use ClassLoader to look for files relative to class location (even inside a jar)
- `getClass().getClassLoader()` `.getResourceAsStream(resourceFileName)` will give an InputStream which you can use in other IO operations.
- Run jar in new location to make sure you are properly loading resources.

Setting resources folder

- Images, sounds, and other files to be included in the jar belong in the resources folder
- Right-click on folder and "Set as resources root"
- Now you'll be able to load files with the class loader using just the file name relative to this resource folder.
- Make sure you include resources when building the jar!