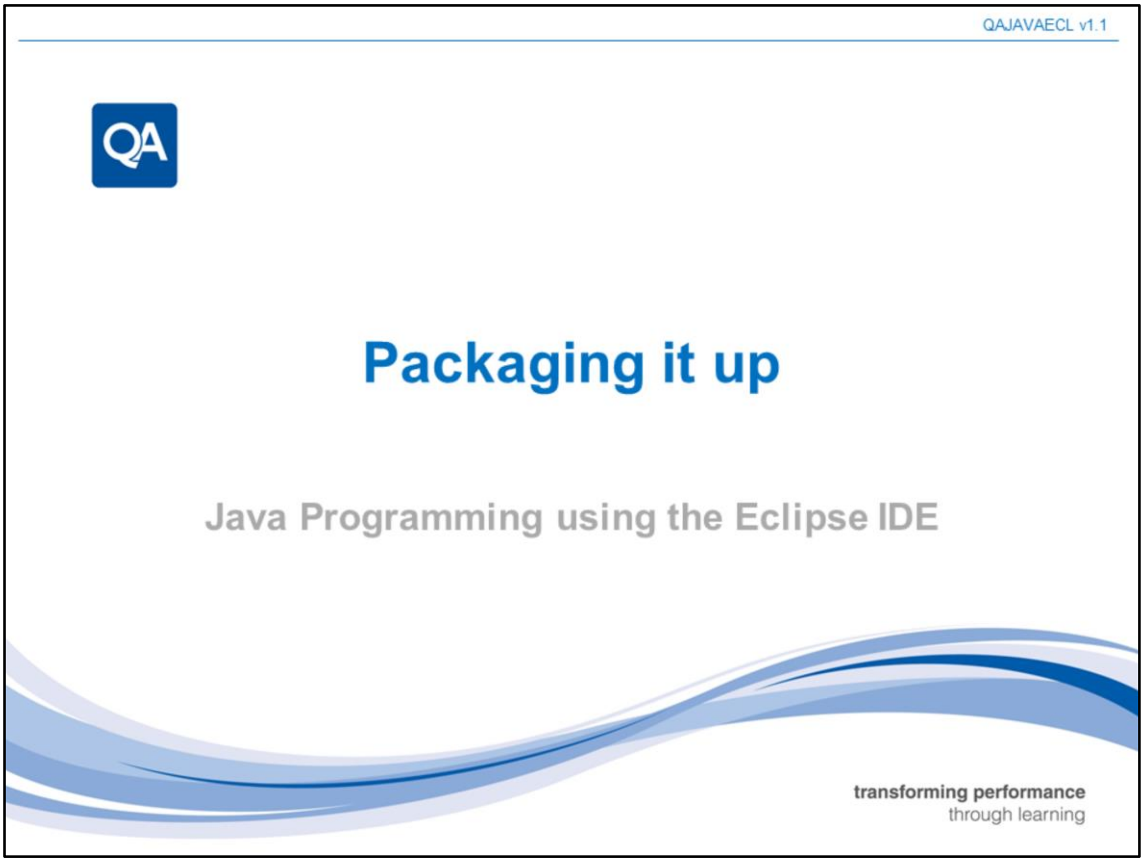
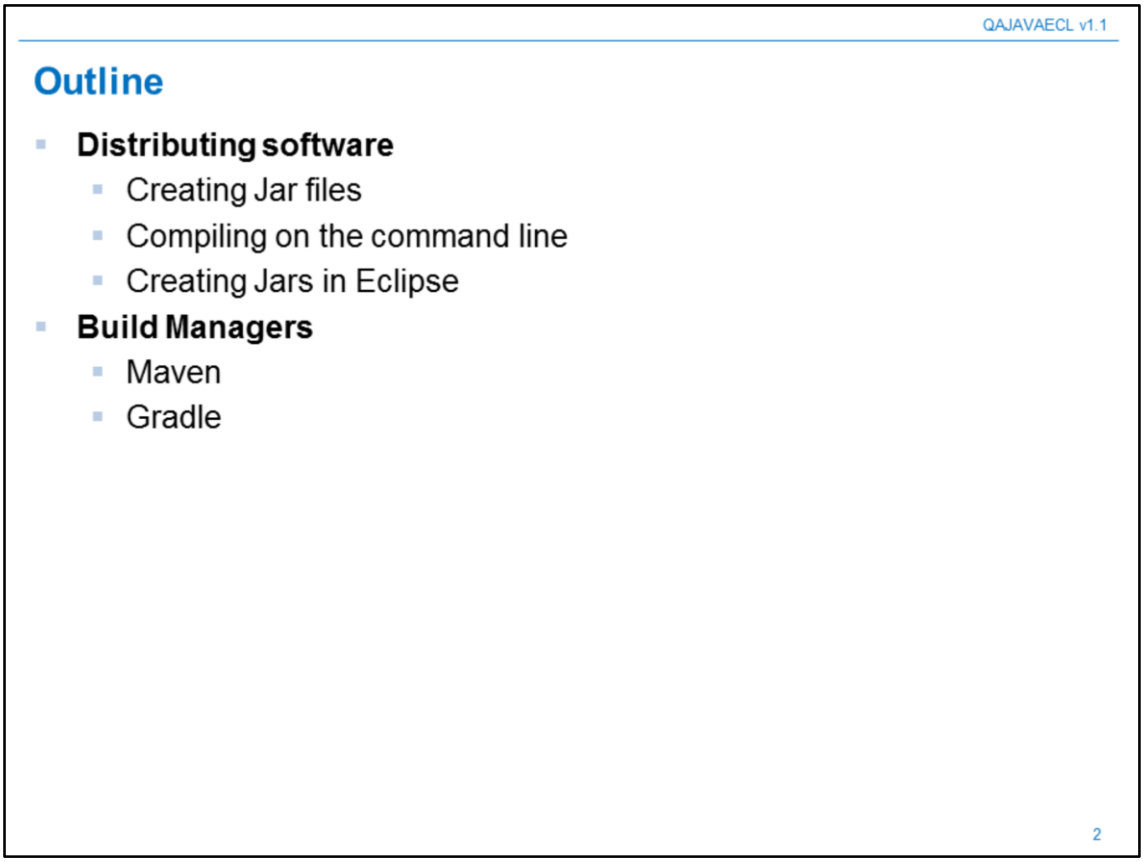


QAJAVAECL v1.1

Packaging it up

Java Programming using the Eclipse IDE

transforming performance
through learning

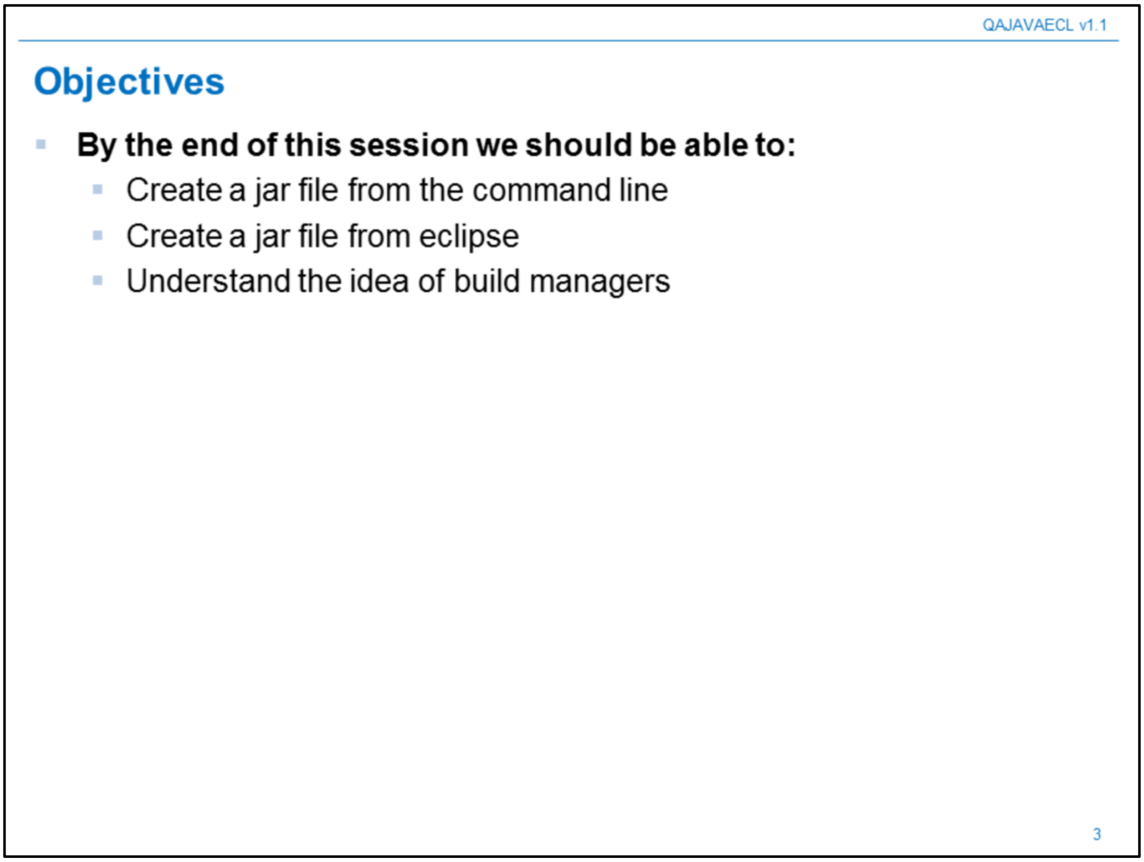


QAJAVAECL v1.1

Outline

- **Distributing software**
  - Creating Jar files
  - Compiling on the command line
  - Creating Jars in Eclipse
- **Build Managers**
  - Maven
  - Gradle

2

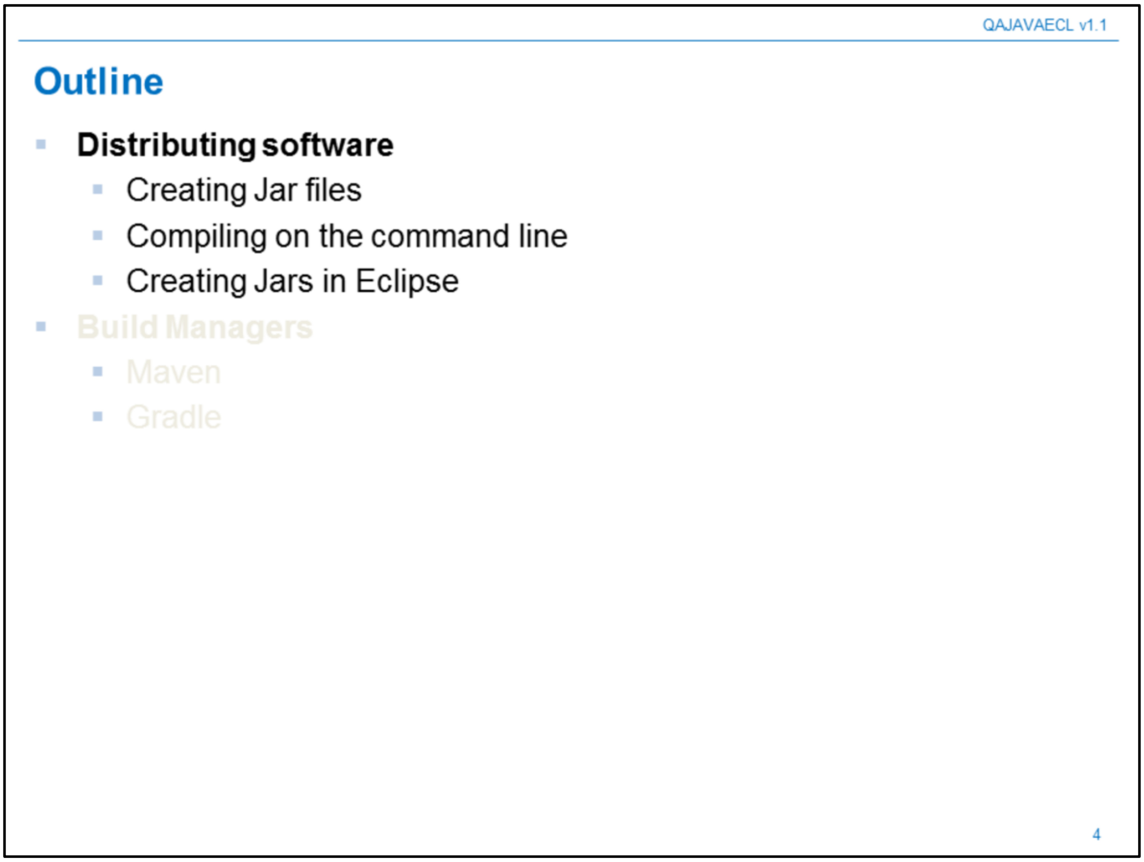


QAJAVAECL v1.1

Objectives

- **By the end of this session we should be able to:**
  - Create a jar file from the command line
  - Create a jar file from eclipse
  - Understand the idea of build managers

3



QAJAVAECL v1.1

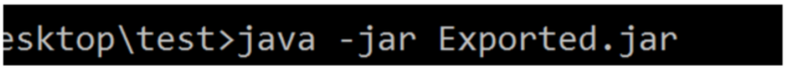
Outline

- **Distributing software**
  - Creating Jar files
  - Compiling on the command line
  - Creating Jars in Eclipse
- **Build Managers**
  - Maven
  - Gradle

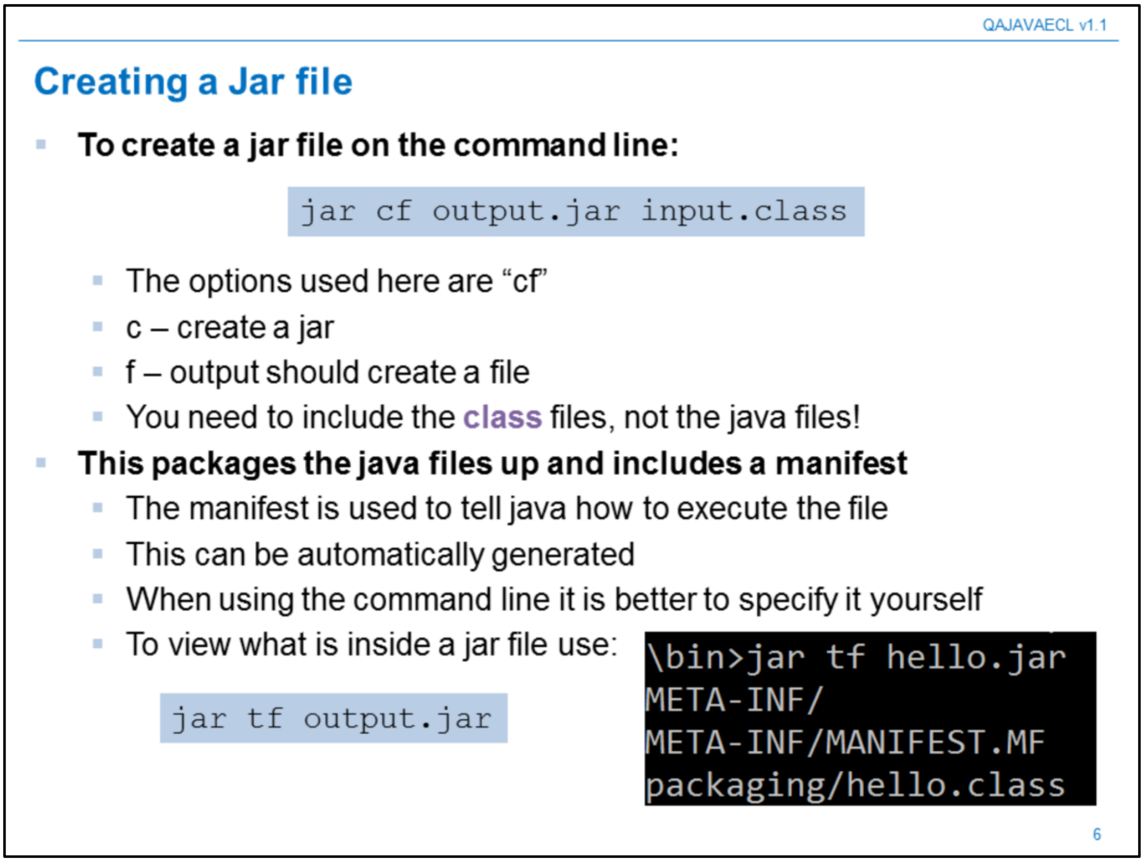
4

Distributing Software

- **So far we've used eclipse to run our programs**
  - Do you want to give the source code to an end user?
  - Do you want to explain how to open eclipse and run the program ever time?
- **Java can generate Jar files**
  - Package the code and libraries into one file
  - Can be executable by clicking on it (like .exe files)
  - Use the command line to launch command line programs



```
esktop\test>java -jar Exported.jar
```



QAJAVAECL v1.1

Creating a Jar file

- **To create a jar file on the command line:**

```
jar cf output.jar input.class
```

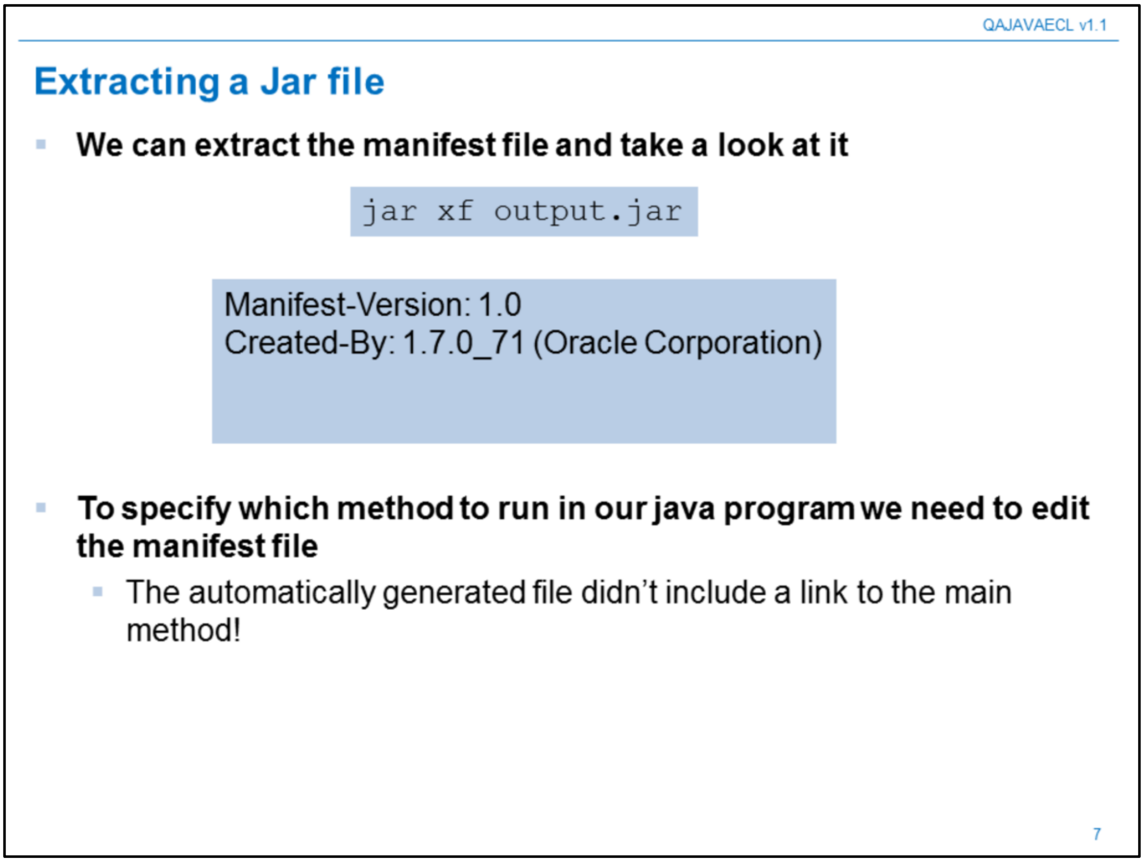
  - The options used here are “cf”
  - c – create a jar
  - f – output should create a file
  - You need to include the **class** files, not the java files!
- **This packages the java files up and includes a manifest**
  - The manifest is used to tell java how to execute the file
  - This can be automatically generated
  - When using the command line it is better to specify it yourself
  - To view what is inside a jar file use:

```
jar tf output.jar
```

```
\bin>jar tf hello.jar
META-INF/
META-INF/MANIFEST.MF
packaging/hello.class
```

6

The official tutorial on the oracle site for this is:
http://docs.oracle.com/javase/tutorial/deployment/jar/index.html



QAJAVAECL v1.1

Extracting a Jar file

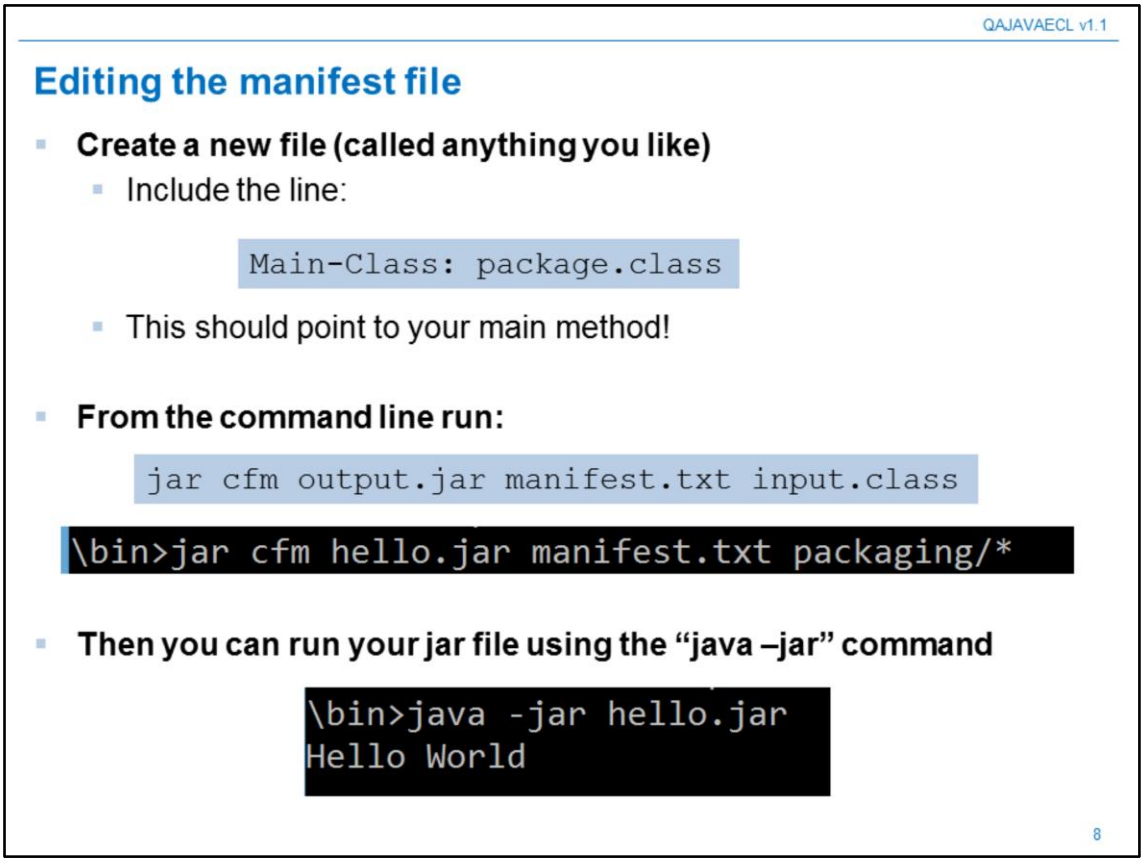
- **We can extract the manifest file and take a look at it**

```
jar xf output.jar
```

```
Manifest-Version: 1.0
Created-By: 1.7.0_71 (Oracle Corporation)
```

- **To specify which method to run in our java program we need to edit the manifest file**
  - The automatically generated file didn’t include a link to the main method!

7



Editing the manifest file

- **Create a new file (called anything you like)**
  - Include the line:

```
Main-Class: package.class
```

  - This should point to your main method!

- **From the command line run:**

```
jar cfm output.jar manifest.txt input.class
```

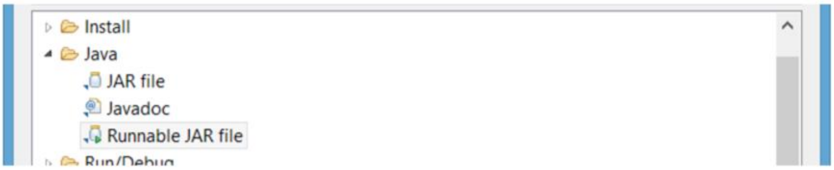
```
\bin>jar cfm hello.jar manifest.txt packaging/*
```

- **Then you can run your jar file using the “java –jar” command**

```
\bin>java -jar hello.jar
Hello World
```

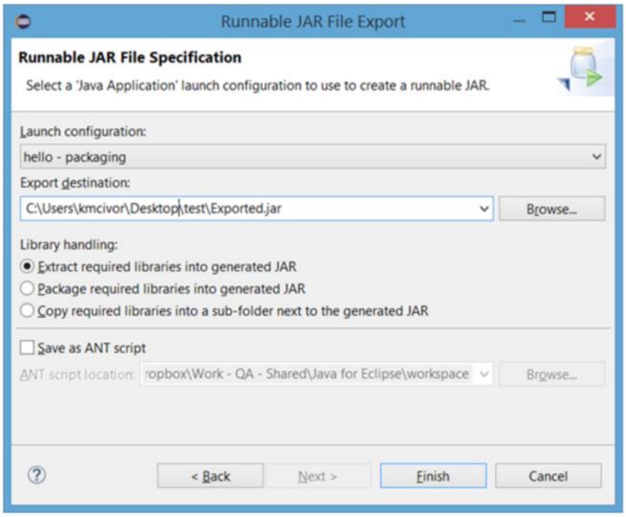
Creating a Jar from eclipse

- **When using eclipse we can export a project to a jar file**
  - File menu → Export
  - We can export to a Jar file or a runnable Jar File
    - A standard Jar file is the same as the one we created on the command line, requires starting with the “java –jar” command
    - An executable Jar file can run as a standalone
      - Good for GUI programs



Creating a Jar from eclipse

- **There are similar options in the dialog screens provided by eclipse as on the command line**
  - Launch Configuration: The main class you want to run
  - Export destination: Where you want the Jar file to be created



Runnable JAR File Export

Runnable JAR File Specification

Select a 'Java Application' launch configuration to use to create a runnable JAR.

Launch configuration: hello - packaging

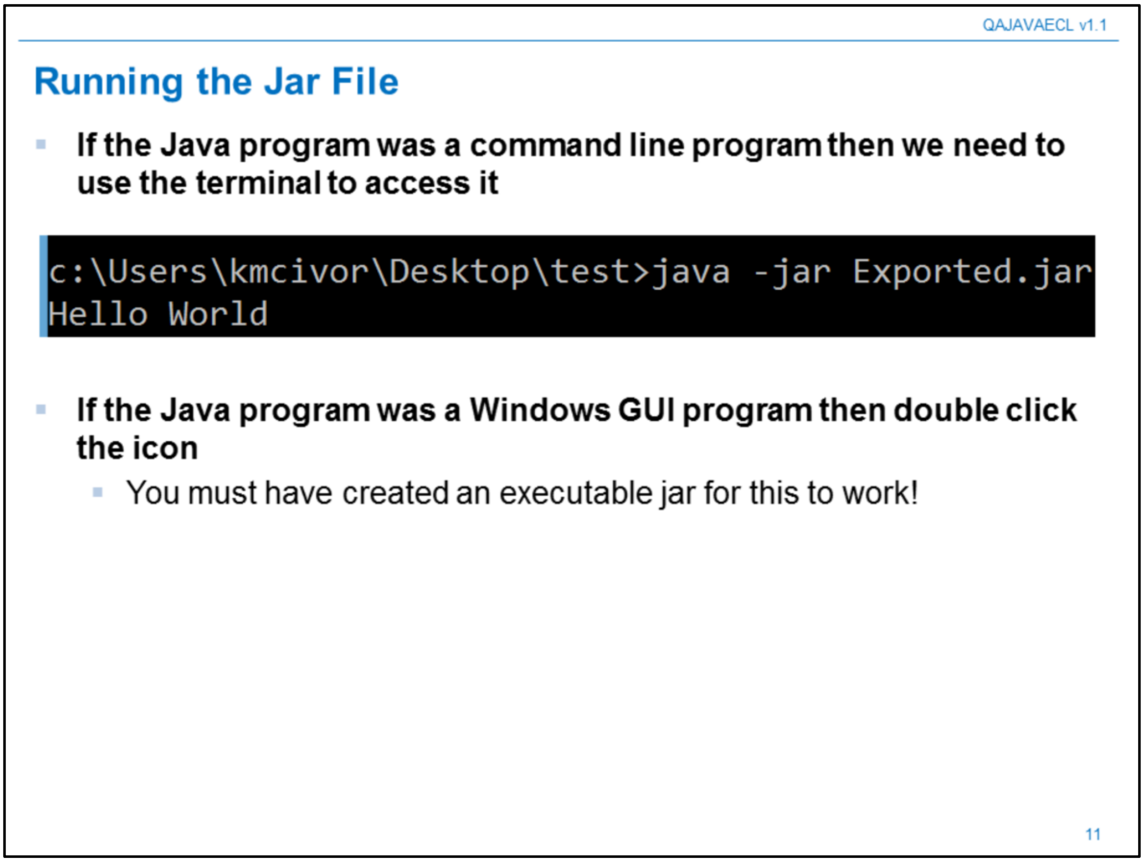
Export destination: C:\Users\kmcivor\Desktop\test\Exported.jar

Library handling:

● Extract required libraries into generated JAR
○ Package required libraries into generated JAR
○ Copy required libraries into a sub-folder next to the generated JAR

□ Save as ANT script

ANT script location: ropbox\Work - QA - Shared\Java for Eclipse\workspace



Running the Jar File

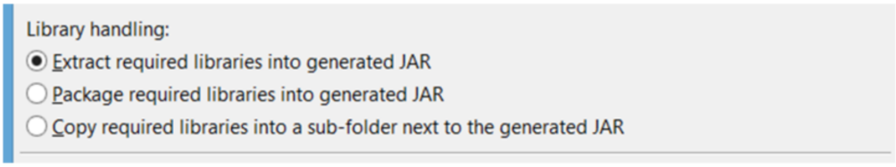
- **If the Java program was a command line program then we need to use the terminal to access it**

```
c:\Users\kmcivor\Desktop\test>java -jar Exported.jar
Hello World
```

- **If the Java program was a Windows GUI program then double click the icon**
  - You must have created an executable jar for this to work!

Bundling other libraries

- **In Eclipse we can bundle all the dependencies when generating the jar file**
  - This will include any libraries that you have added to your project



Library handling:

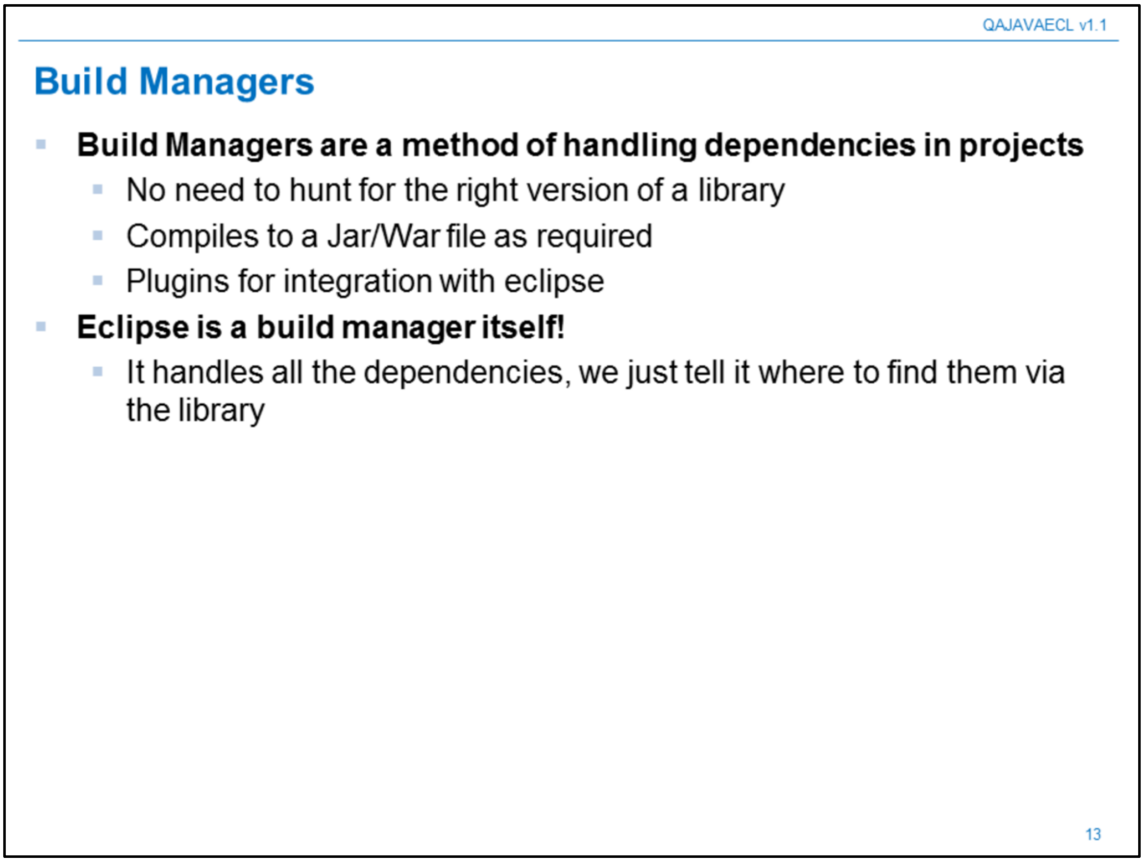
● Extract required libraries into generated JAR

○ Package required libraries into generated JAR

○ Copy required libraries into a sub-folder next to the generated JAR

  - The runnable jar includes everything as standard

- **On the command line we bundle dependencies by specifying them in the command that creates the jar file**

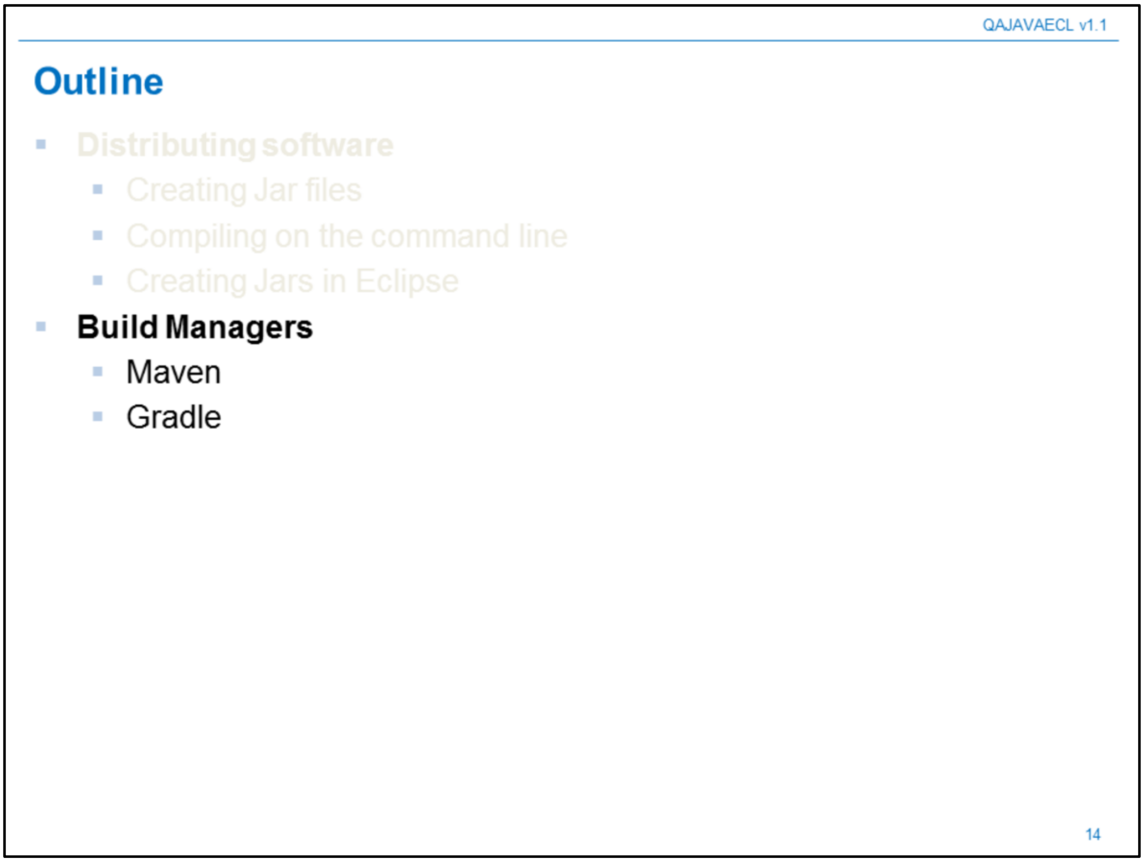


QAJAVAECL v1.1

Build Managers

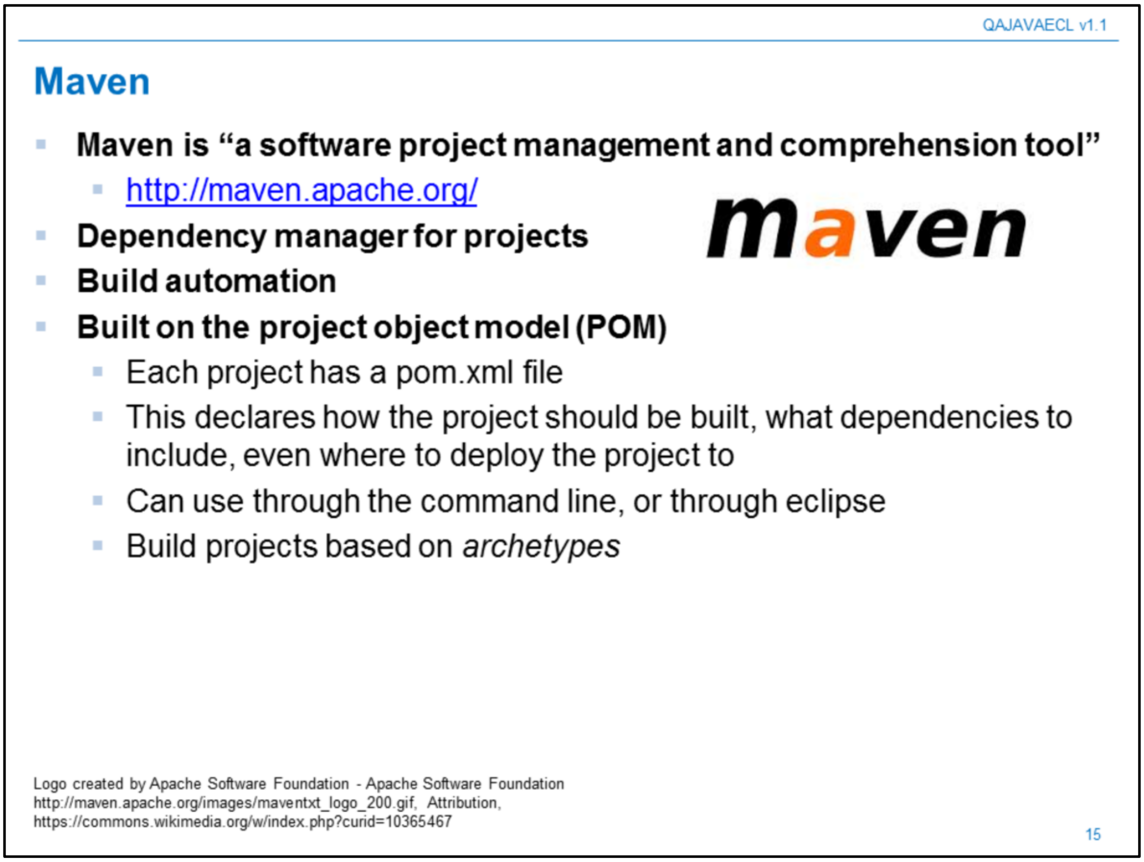
- **Build Managers are a method of handling dependencies in projects**
  - No need to hunt for the right version of a library
  - Compiles to a Jar/War file as required
  - Plugins for integration with eclipse
- **Eclipse is a build manager itself!**
  - It handles all the dependencies, we just tell it where to find them via the library

13



Outline

- Distributing software
  - Creating Jar files
  - Compiling on the command line
  - Creating Jars in Eclipse
- **Build Managers**
  - Maven
  - Gradle



QAJAVAECL v1.1

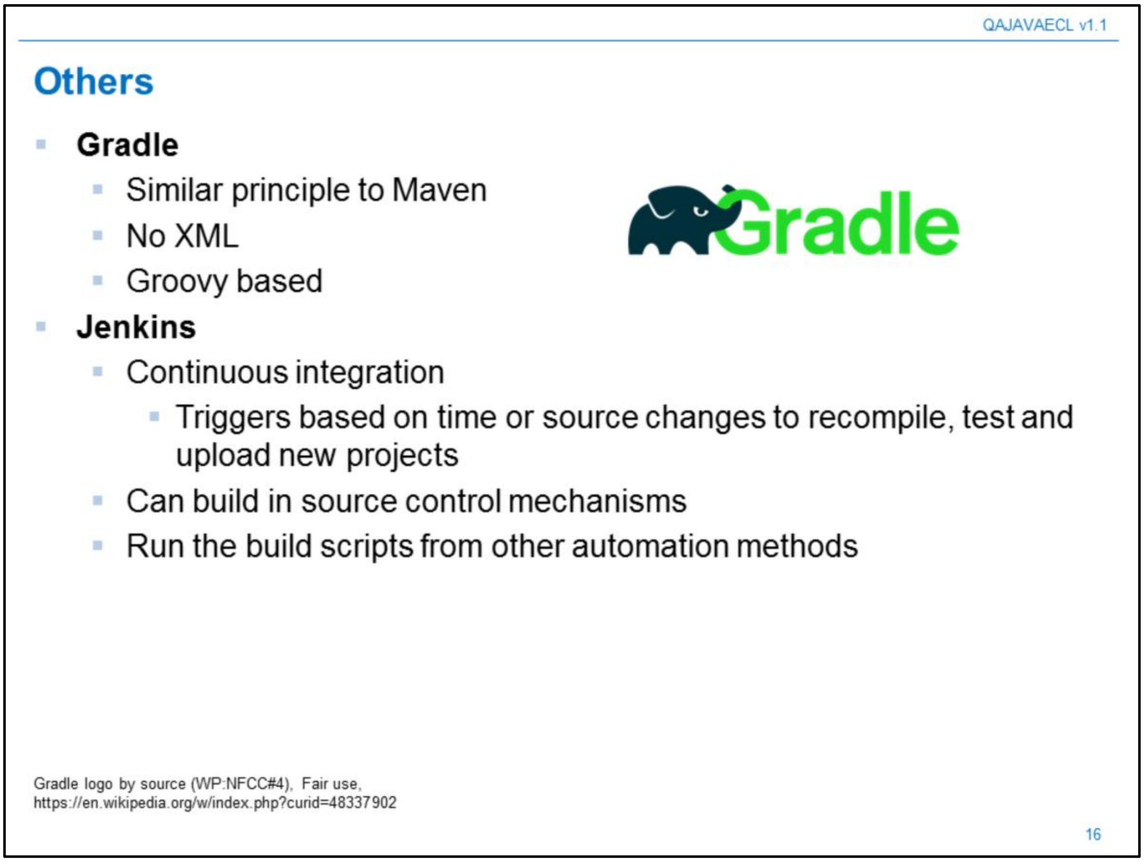
Maven

- **Maven is “a software project management and comprehension tool”**
  - http://maven.apache.org/
- **Dependency manager for projects**
- **Build automation**
- **Built on the project object model (POM)**
  - Each project has a pom.xml file
  - This declares how the project should be built, what dependencies to include, even where to deploy the project to
  - Can use through the command line, or through eclipse
  - Build projects based on *archetypes*

Logo created by Apache Software Foundation - Apache Software Foundation http://maven.apache.org/images/maventxt_logo_200.gif, Attribution, https://commons.wikimedia.org/w/index.php?curid=10365467

15

All logos are owned by their respective companies!



QAJAVAECL v1.1

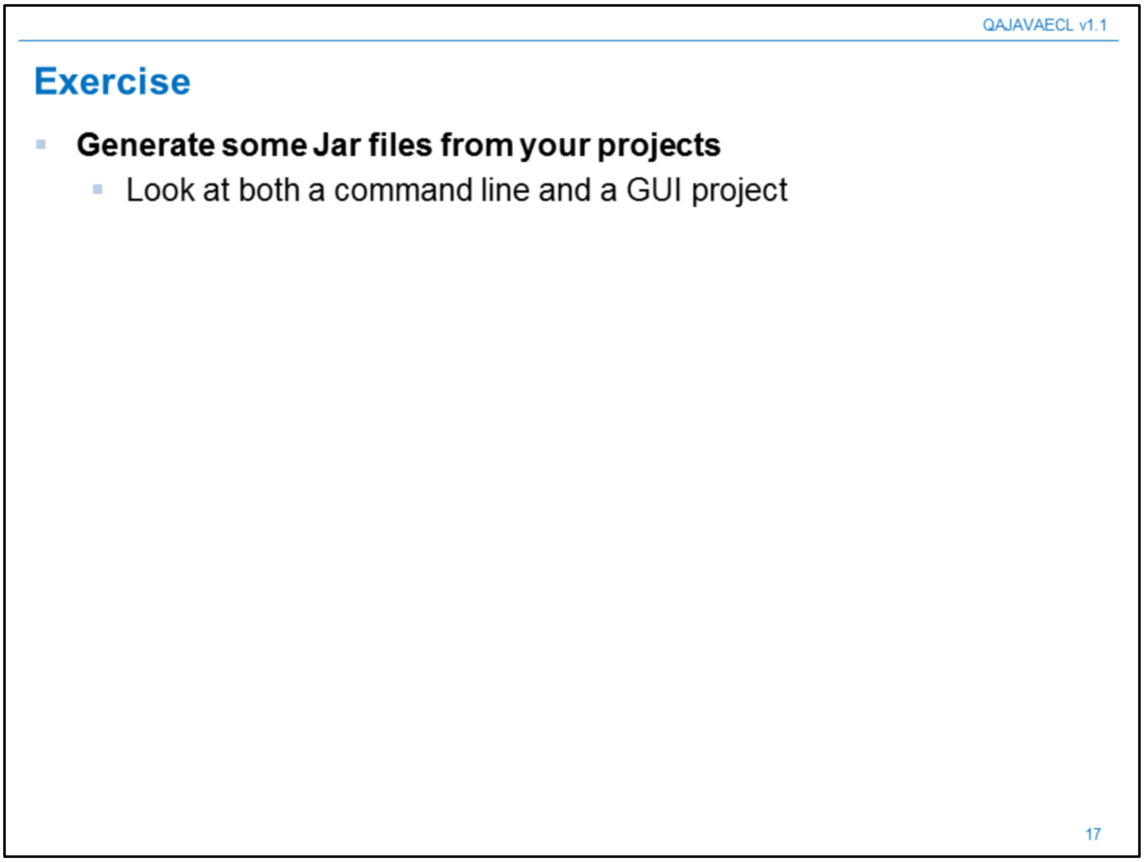
Others

- **Gradle**
  - Similar principle to Maven
  - No XML
  - Groovy based
- **Jenkins**
  - Continuous integration
    - Triggers based on time or source changes to recompile, test and upload new projects
  - Can build in source control mechanisms
  - Run the build scripts from other automation methods

Gradle logo by source (WP:NFCC#4), Fair use, https://en.wikipedia.org/w/index.php?curid=48337902

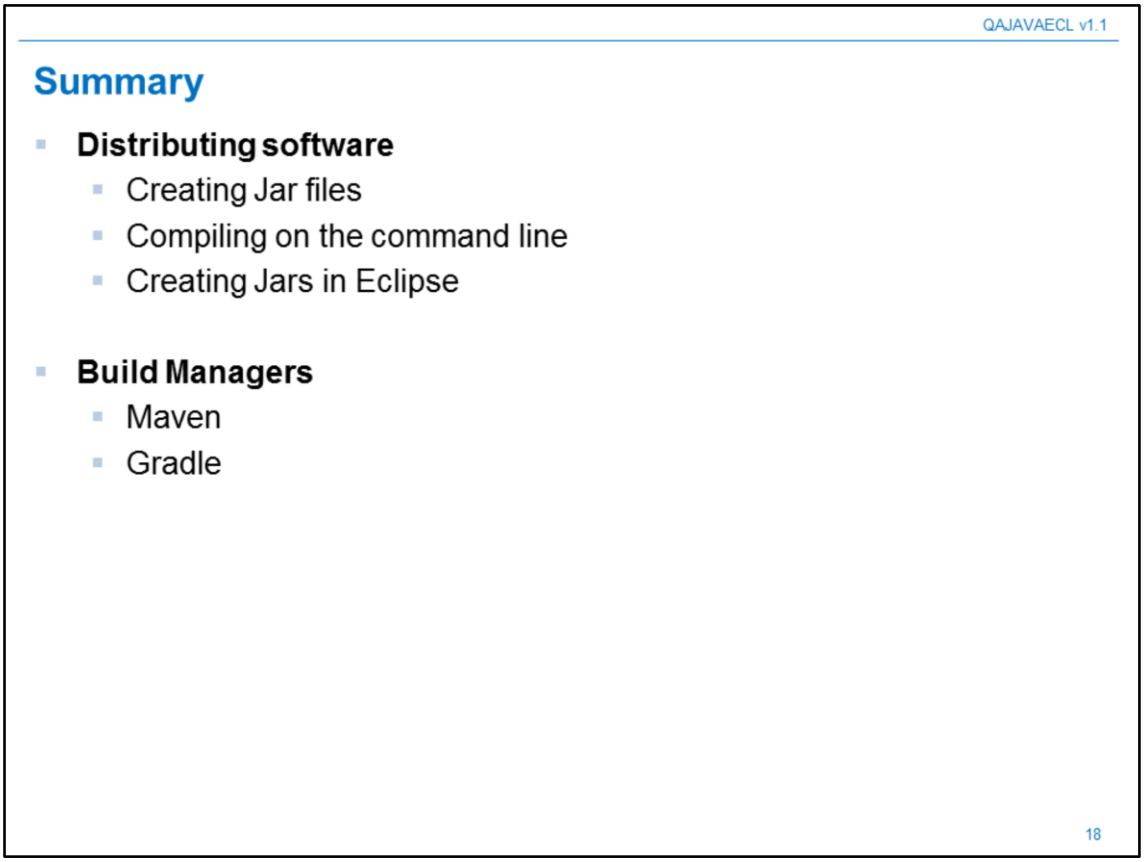
16

All logos are owned by their respective companies!



Exercise

- **Generate some Jar files from your projects**
    - Look at both a command line and a GUI project



QAJAVAECL v1.1

Summary

- **Distributing software**
  - Creating Jar files
  - Compiling on the command line
  - Creating Jars in Eclipse
- **Build Managers**
  - Maven
  - Gradle

18