CE0825a - Object Oriented Programming II
6: Inheritance/interfaces recap; Graphics; packaging

James A Sutherland

Abertay University

Monday, 15th February 2016

Object Oriented Programming

You can actually do "object oriented programming" in almost any language, even assembler!
The essence of OOP is that the code is associated with the data. Think of it in terms of the data having functionality of its own: where in non-OOP you might 'measure the length of a string' (for example, call `strlen` in C), in OOP you 'ask the string object how long it is'.
As a consequence of that, you might have different types of object with the same functionality. For example, you could have a special very long string type for storing entire documents, where rather than counting characters the length is pre-calculated and stored somewhere. Other bits of code don't need to know about that: they just ask for the length, and get it.

Early OOP: C++ or even C

Early 'C++ compilers' actually just turned C++ back into plain old C!
C has a data structure system (`struct`), and you can use pointers to functions as data. Put a list of function addresses somewhere, put a pointer to that at the start of every 'object' structure – and you have OOP! (Chain those tables together for inheritance.)
Microsoft also did 'objects in C' for COM; as further reading see `http://www.codeproject.com/Articles/13601/COM-in-plain-C`.

Java Inheritance: Another View

Back before Java and objects, we had data structures like this:

```
struct point {
        int x;
        int y;
        enum { GRASS, TREE, RIVER } pointtype;
};
```

No code, just data, so we'd create things like search_point.

Java Inheritance: Another View 2

We could extend, by wrapping and adding bits:

```
struct 3dpoint {
        struct point 2dpoint;
        int z;
};
```

Except now search_point doesn't understand how to search 3dpoints: it won't 'know' that a 3dpoint is a superset of a point. Why can't it operate on both types?

Java Inheritance: Another View 3

Object-oriented: associate functions (which we now call *methods* instead) with these structures (which are now *classes*).
We introduce a rule: when looking for a function, try the parent (then grandparent, etc) before giving up.
So, we have `Vector`, which extends `AbstractList`, which ... `AbstractCollection`, `Object`.

Java Interfaces

Interfaces are just a list of zero or more methods.
You mark a class as *implementing* an interface with the *implements* keyword.

OOP summary

In summary:

- OOP is about associating code more closely with data
- 'Active' structures which have functions not just data
- Inheritance lets you extend a class with extra data or features
- An interface defines a feature a class has got

Graphics formats – Introduction

Graphics files come in two forms: *vector* (mathematical description of shapes) and *bitmap* (a regular array of dots). Bitmaps in turn can be stored in *lossy* or *lossless* formats. JPEG is lossy: deliberately throws 'less important' data away to make a small photo that looks OK. PNG and GIF are lossless: mathematically identical at every stage, but bigger files.

A PDF can contain any of these types, along with text, scripts and other content.

So ... which of these would you use for a map?

Real world maps

Real maps use a mix depending on browser, content ... Google Maps draws route overlays as a vector object using SVG, for example.
For the coursework, we'll be using Abertay campus maps. These are available as PDFs – unfortunately, each PDF just contains a single JPEG photo of an actual map. So, can't search, edit or zoom sensibly.

Graphics summary

Graphics can be:

- Vector – shapes, curves, infinitely scalable
- Bitmap – pixels, zoom and you get big pixels
  - Lossy, for photographs
  - Lossless, for diagrams, screenshots

Packaging Java apps

Java libraries and applications can be packed into JAR files

- JAR = ZIP with different extension
- Can create/edit/view with regular Zip software
- But (usually) with special data: manifest file
- Usually created automatically as part of build

Commercial Java

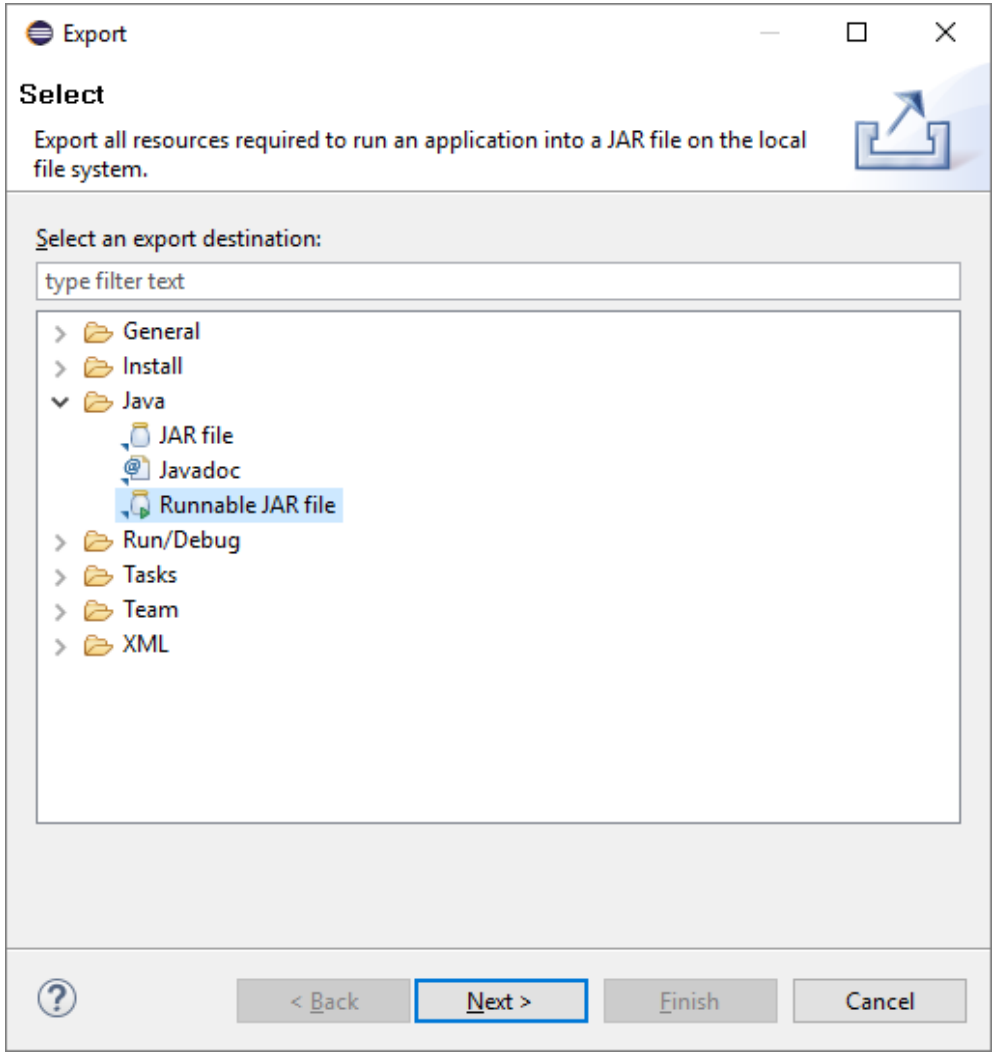
For commercial distribution, a Jar file is rare. More commonly:

- Java Web Start - Java downloads and runs from the web
- Wrap in an EXE file (Windows) or .app (Mac)
- Various commercial tools for doing this too

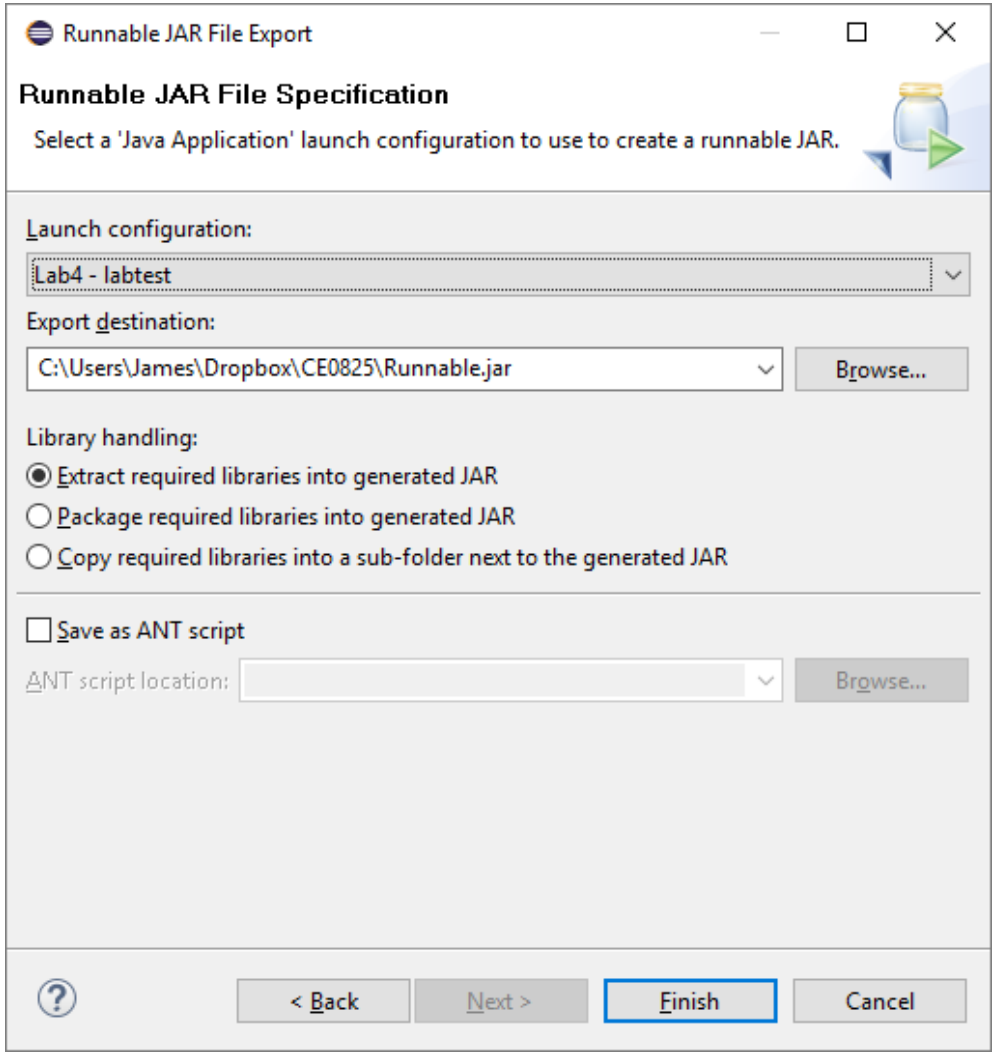
Creating JARs from Eclipse

For now, we'll create one manually by right-clicking on your project and selecting 'Export...'
Note the mention of an Ant task. That's the automation I mentioned earlier: on big commercial projects, with lots of source code to manage, you use a tool like Ant to get everything compiled and current with a single command, or even automatically on a schedule (continuous integration, CI).

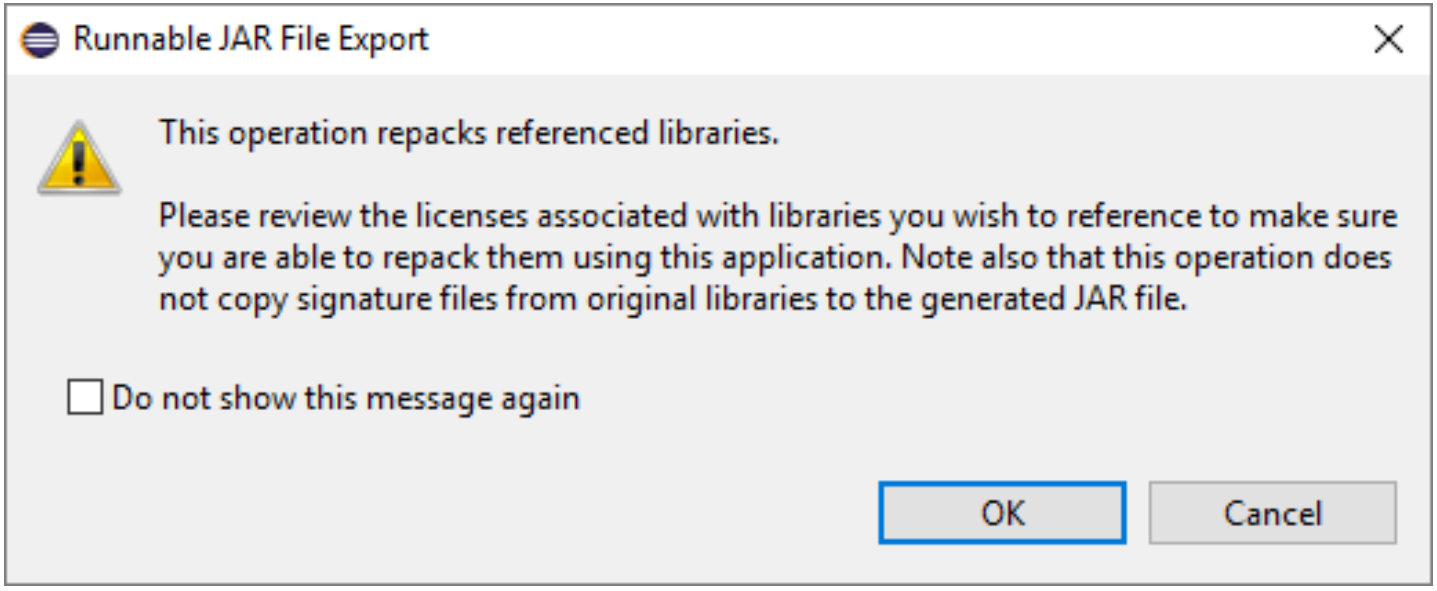
JAR 1


Export
Select
Export all resources required to run an application into a JAR file on the local file system.
Select an export destination:
type filter text
General
Install
Java
JAR file
Javadoc
Runnable JAR file
Run/Debug
Tasks
Team
XML
< Back
Next >
Finish
Cancel

JAR 2


Runnable JAR File Export
Runnable JAR File Specification
Select a 'Java Application' launch configuration to use to create a runnable JAR.
Launch configuration:
Lab4 - labtest
Export destination:
C:\Users\James\Dropbox\CE0825\Runnable.jar
Browse...
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:
Browse...
< Back
Next >
Finish
Cancel

JAR 3


Runnable JAR File Export
This operation repacks referenced libraries.
Please review the licenses associated with libraries you wish to reference to make sure you are able to repack them using this application. Note also that this operation does not copy signature files from original libraries to the generated JAR file.
Do not show this message again
OK
Cancel

Lab Task 6

There are five JPEG map images at `http://driesh.abertay.ac.uk/~j204453/` - use them!
Two tasks there - display an image in an SWT window, and provide some sort of selector for the 5 levels.
For practice, try packaging your SWT application as a JAR file and running it. (Not for submission, just for use later.)

Week 7 – Coursework

The coursework is to build a map of the Abertay campus as a Java (SWT) application. For week 7, think about how you will achieve this: what features would you like? Searching, directions ... For week 7, present a *specification* for your project: what it is to achieve and how. During week 7 you will get feedback on this plan, which you should incorporate in your report.

Coursework deliverables:

- Java application
- Project report: what you did, why, and how

NB: Week 7 is attendance monitoring week! No lectures, though, just labs as usual 3-5 in 4506.