CE0825a - Object Oriented Programming II
3: Object Recap, GUI, Regular Expressions

James A Sutherland

Abertay University

Monday, 25th January 2016

Objects Recap

Everything is based on `java.lang.Object`.
When we talk about "an object", we mean an instance of a Class.
Real world equivalent: my phone (an object) is an iPhone 6+ (a class).
Java best practice: Title Case for Classes, lower case for objects/instances.
Subclassing: "it's like an X, *but*..."
RW: "The iPhone 6+ is like an iPhone 6, but a bit bigger"
i.e. it has the same buttons, software, sockets ...
in Java: `class BigPhone extends OrdinaryPhone`

Interfaces, Abstracts

An Abstract Class is just a class you can't *directly* instantiate, only subclass, usually because it has something missing: for example, some essential method is not yet implemented.
For example, a button defines some things - but not what the text on it says, or what happens when it's pressed.
An interface just lists some functionality that must be present.
For example, "Deletable" might specify that any class implementing it as a method "delete":
`interface Deletable { void delete(); }`
Note: Title Case (like Classes), no method bodies allowed at all. Ever.

Generics

When we don't care what things are, but they all need to be the same. For example, the implementation of a List we saw last week, in Vector<?>.
Or you can care a little bit: `Vector<? extends Number>`.
You specify when you create the instance of Vector exactly which kind of Number you want.

Anonymous Classes

What use is a class without a name? When you're using it as a one-off instance (see also the "Singleton Pattern").

```
Double fortytwo = new Double(42) { String
toString() { return "What are six sevens?"; } }
```

It's a normal Double, you can do maths with it:

```
System.out.println(fortytwo+1.0);
```

but it's got an extra feature:

```
System.out.println(fortytwo);
```

Concurrency Control

Java tends to use lots of threads. If you're writing a web or chat server in Java, you'll almost certainly create a thread per client/connection.

You can create a new thread by either subclassing `Thread`, or implementing `Runnable`. Either way, you'll be writing a new method, `run()` which does something.

```
Thread t=new Thread() { void run() { ...  } };
t.start();
```

Potential pitfall here: do NOT call `run()` directly! That will skip creating a new thread, and just run the code directly, on the thread you're in now.

Threading example

```
class Demo extends Thread {
  public void run() {
    for (int i=1;i<5;i++) {
      System.out.println(i);
      Thread.sleep(100);
    }
  }
  public static void main(String args[]) {
    for (int i=1;i<3;i++) {
      Thread t=new Demo();
      t.start();
    }
  }
}
```

Threading example details

- Why extend Thread?
- Where does `start()` come from?
- How would I do it by implementing `Runnable`?
- Why would I want or need to do that?

Threading example details

- Why extend Thread?
- Where does `start()` come from?
- How would I do it by implementing `Runnable`?
- Why would I want or need to do that?

Threading example details

- Why extend Thread?
- Where does `start()` come from?
- How would I do it by implementing `Runnable`?
- Why would I want or need to do that?

Threading example details

- Why extend Thread?
- Where does `start()` come from?
- How would I do it by implementing `Runnable`?
- Why would I want or need to do that?

Graphical User Interfaces — AWT

AWT: Sun's first attempt at giving Java a GUI, back in 1995.
Unfortunately ... it wasn't very good.
(To be fair, a totally cross platform GUI library was new territory at the time!)
In 1995, it was new, and seemed cool. It tried to use the native interface components.

GUI take 2 — Swing

In the next version of Java, 1.2, Sun tried again.
Swing.
A whole new look and feel of its own: whatever platform you're on, Swing looks the same.
Which means your Java applications stick out like a sore thumb! People don't like that.
Sun/Oracle moved on to JavaFX — which we will ignore.

GUI take 3 — SWT

Use the native UI components, but with enough of a Java abstraction layer on top to make things work uniformly everywhere.
This is what Eclipse is built on, so you've already used an SWT application quite a bit.

Start with SWT

Download the SWT framework from eclipse.org — be careful, it defaults to 64 bit for Windows!
Follow instructions:
`https://www.eclipse.org/swt/eclipse.php`

SWT first programme

```
public static void main(String[] args) {
    Display display = new Display ();
    Shell shell = new Shell (display);
    shell.open ();
    while (!shell.isDisposed ()) {
      if (!display.readAndDispatch ())
       display.sleep ();
    }
    display.dispose ();
    }
}
```

Regular Expressions

A simple text description language for matching strings. In Java, `String.match()`.
For example, check if a string starts with a capitalised word: `"^[A-Z][a-z]*"`.

Week 3 Lab Tasks

- Get the SWT example working
- Add a MenuBar, MenuItems, window title and StyledText widget
- Make that fill the window, complete your text editor application
- Fill up the MenuBar with the usual File, Edit etc.
- If that all seems easy, implement some regular expressions. Report matches to the user.