Programming: Week 24 Lab (Polymorphism and Refactoring)

This week's lab is an exercise in **Refactoring.** You are given some code which works, but which is poorly designed. Your task is to improve the design using inheritance and polymorphism.

Learning objectives:

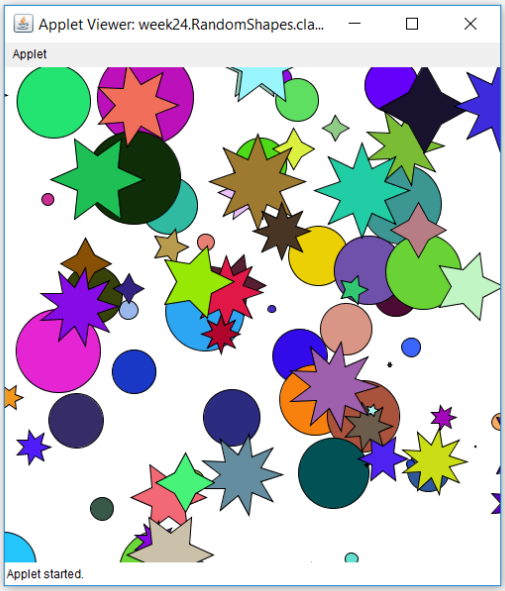
- Using **Inheritance** and **Polymorphism** to simplify a design.
- **Refactoring** – improving the design of existing code with the benefit of hindsight.
- A recap on using Processing in Eclipse.

Resources

- Week 23 Lecture Slides.
- Week 10 Lab (Moodle week 28 November - 4 December).

Preparation

- Create a new Java project in Eclipse.
- Download the Java file **RandomShapes.java** from Moodle, and import it into the **src** folder of the project (use **Import > Filesystem** or create a new class called **RandomShapes** and cut-and-paste the code from Moodle into the new class file.
- Import the Processing **core.jar** file into the project, and add it to the build path. Refer to Exercise 2 in Week 10's lab (see resources above) if you have forgotten how to do this.
- Run as a Java Applet – you should see some random circles and stars moving gently around the screen in a pleasing manner (see below)



Scenario

The code produces a pleasing effect, but if you look at the code itself, it is a lot less pleasing. It looks like the programmer started out making the simulation with circles only. A **Circle** class was defined, with update and draw methods, and an **ArrayList** of these added to the main class. The **ArrayList** is iterated over once per update, calling the update and draw methods. So far, so good. Then the programmer figured out how to make stars. However, at this stage, instead of thinking carefully about the design, they simply cut and paste the circle class, renamed it to **Star**, modified the draw method, and added another list in the main class to be iterated over. This is not so good. You task is to **refactor** the design, using inheritance and polymorphism.

Stage 1

Look over the code to see how it works. The main class is called **RandomShapes**. It extends **PApplet**, and overrides the **setup** and **draw** methods. There is a helper function (**randomColour**) for choosing the random colours of the shapes. There are two ArrayLists – one of **Circles**, and one of **Stars** (see below). The **setup** method fills the lists with random shapes. The **draw** method iterates over the lists, calling the shapes' update and draw methods.

The **Circle** class has fields for x, y (position), size, colour, parent (a reference to the parent **PApplet**) and xt, yt (the position of its current target point in its wanderings). There are **update** and **draw** methods which do the obvious things. The update method blends the current position towards the target, and when it is sufficiently close, it chooses a new random target.

The **Star** class has one extra field – **numPoints** – the number of points on the star. The **draw** method is different but the update method is identical to the one in **Circle**.

Stage 2

Make a class that both **Circle** and **Star** can inherit from. Copy the relevant code into your new class, then modify **Circle** and **Star** so that they inherit from your new class. You will need to call the superclass constructor in your constructors for **Star** and **Circle** – look at the **ActorComponent, PhysicsComponent** and **MeshRenderComponent** classes in Lecture 23 for inspiration – note in particular how the derived class constructors call **super**, then fill in any extra data.

Think about which of the two methods (update and draw) you will need to override in the derived classes and which you can leave in the superclass.

Don't change the **RandomShapes** class at this point – this should still work as it did. Confirm that the code still behaves as expected.

Stage 3

Now clean up the **RandomShapes** class using polymorphism. You should be able to replace the two ArrayLists with one. Look at the revised **GameActor** class in Lecture 23 for inspiration.

Portfolio

This week's deliverable for the portfolio is the refactored version of **RandomShapes.java**. It should use inheritance and polymorphism. Both the **Circle** and **Star** classes should inherit from a common superclass, and there should be only **one** list of objects in the main class.