Polymorphism and Composition Quiz Homework

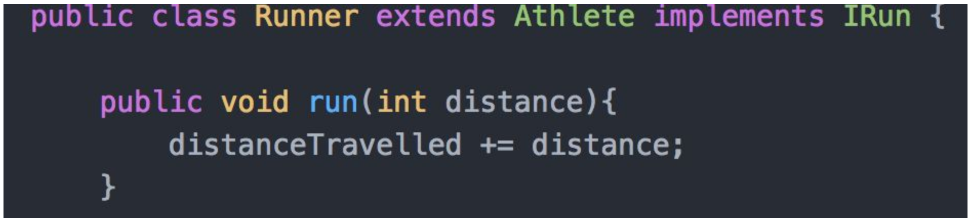
Polymorphism

1. What does the *word* ‘polymorphism mean?

Many forms/types.

2. What does it mean when we apply polymorphism to OO design? Give a simple Java example.

It means that an object can be considered as more than one thing. E.g a Runner can be considered both an Athlete and a Runner (if the Runner class is inheriting from the Athlete class).



```java
public class Runner extends Athlete implements IRun {

    public void run(int distance){
        distanceTravelled += distance;
    }
```

3. What can we use to implement polymorphism in Java?

We can use:

- Inheritance: Like the car in the example above. The Vehicle class is the super class, and the Car class is the child which inherits from the Vehicle class. The car can be considered both as a car and a vehicle.
- Interfaces: The Car class could implement an interface such as IDrive, ie any object that can drive such as a car or a motorbike etc can be considered a Car, an IDrive object, and a Vehicle.

4. How many ‘forms’ can an object take when using polymorphism?

As many as you want, there is no real limit.

5. Give an example of when you could use polymorphism.

If you have multiple classes that have the same properties, you can use a superclass that they can inherit these properties from.

Or, if you have multiple classes that have the same behaviours, you can use an interface to provide these behaviours.

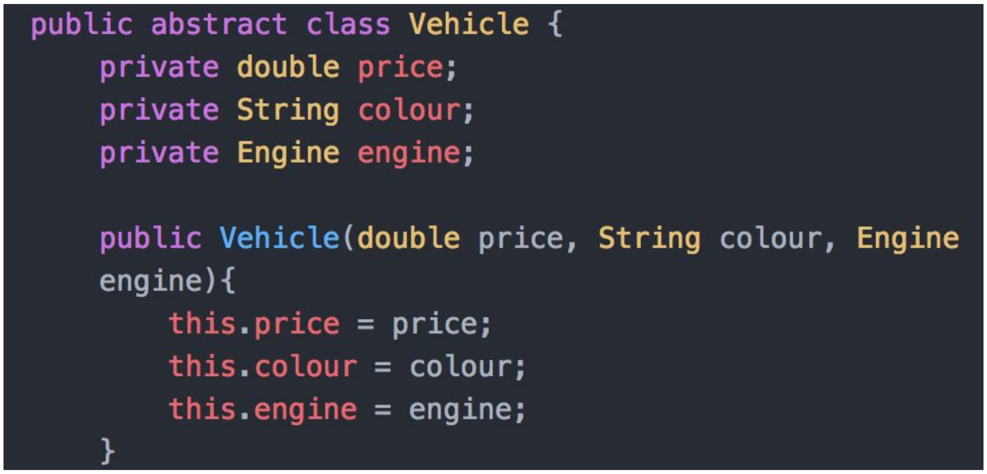
Composition

1. What do we mean by ‘composition’ in reference to object-oriented programming?

If an object is composed of other objects this is known as composition.

2. When would you use composition? Provide a simple example in Java.

E.g a vehicle HAS AN engine, the engine is an object within itself, but is the property of the vehicle object.



```
public abstract class Vehicle {
    private double price;
    private String colour;
    private Engine engine;

    public Vehicle(double price, String colour, Engine
    engine){
        this.price = price;
        this.colour = colour;
        this.engine = engine;
    }
```

3. What is/are the advantage(s) of using composition?

It allows us to get behaviours from other classes without having to use inheritance. Inheritance would force us to inherit everything from a class, whereas composition allows us to choose what behaviours we want to incorporate into one class from another.

Inheritance also limits us to inheriting from one class only, with composition we can choose multiple other classes to get behaviours from, and choose which behaviour we want in a given situation.

4. What happens to the behaviours when the object composed of them is destroyed?

The object composed of other behaviours owns those behaviours. When the object is destroyed the behaviours are also all destroyed.