PAL Consolidation 13 *University of Hertfordshire – Week 15 – March 2016 –*

Aim & Objectives

- Discuss previous session
- Look forward to next session

Introduction

We will try to cover the most basic key Java concepts that relate to how Python and begin show how different Java is. If you run through the following exercises hopefully this will become clear.

Exercise - Classes

You may notice that in order to start the program, you must run:

```
public static void main(String[] args){
   /* Look at me, I'm a comment */
   /** I'm also a comment **/
   // As am I!
}
```

Many of you that don't know Java will begin to start writing commands here as you would with Python. The first line, typically '`public class ClassName`' will declare the name of the class you're currently working in. All classes are declared in a similar fashion, with some declaration near the start of the file as to their type.

It is then expected that the user of Python will want to start working in a more Object Orientated (OO) setting. This can be done in the following way:

```
public class ClassName{
   public static void main(String[] args){
      ClassName cn = new ClassName();
   }
}
```

What we have done here is we have created an object. This is something we may now manipulate to any affect we like. We will discuss how this can be done in the next exercise, what's important now is that we may make an '`instance`' of a '`class`', something we call an '`object`'. Instances are created with the '`new`' keyword.

There are many reasons we may want to create a class, but for now we can think of it as a way of organising ideas. Similar methods or similar ideas can be grouped and used together. If you have multiple cat objects you need to create for example, you may choose to create a class containing everything they have in common, such as name, fur type, size, etc.

Create you own class called *Dog* with a '`main()`' method that allows the program to be run. We'll build on this later.

Exercise - Objects & Variables

An '`object`' is an '`instance`' of a '`class`'. The class can be anything, but typically if we want our own version of that class, either decided at runtime or program initialisation, we will typically need an instance. These can be created in the following way:

```
ClassName obj = new ClassName();
```

If the class name followed by the brackets looks all too familiar, it's because they are. It's a method - the method being classed is to initialise the object. This can be taken control of in the following way:

```
public class ClassName{
   public static void main(String[] args){
      ClassName cn = new ClassName();
   }

   public ClassName(){
      System.out.println("Hello");
      /* TODO: Write code here.  */
   }
}
```

As you can see when you run the code, the initialiser is something that is run to setup the instance of the class. We can pass values when we instantiate the object as parameters. This is normal practice that can be seen with our example class:

```
public class ClassName{
   private int num = 0;

   public static void main(String[] args){
      ClassName cn = new ClassName();
   }

   public ClassName(int n){
      num = n;
   }
}
```

Change your dog class to take in the following information:

- *Name* – A '`String`' containing the dog's name.
- *Age* – An '`int`' containing the dog's age.
- *Breed* – A '`String`' containing the dog's breed.
- *Colour* – A '`String`' containing the dog's colour.
- *Temperament* – A '`String`' containing description of how the dog behaves.

Exercise - Methods

Methods are a lot like those in Python. You want to take parameters and do something specific to the object. This can be done in the same way Python also does it's methods, with a '`foo.bar()`' format. The difference here is that types are important for a logical flow of a program.

Some nice things we can do if we specify types are the following:

- Guarantee that if a method is run it has the correct type passed to it.
- "*Overload*" methods, meaning that they are specified multiple times with different parameters (but never the same parameters).
- Specify whether the methods are accessible from the outside ('`public`') or only accessible internally ('`private`').

Experiment with your Dog class and write methods to get the information back that you specified. Methods should include:

- '`getName()`' – Gets the name of the dog.
- '`getAge()`' – Gets the age of the dog.
- '`getBreed()`' – Gets the breed of the dog.
- '`getColour()`' – Gets the colour of the dog.
- '`getTemperament()`' – Gets the temperament of the dog.

Implement any other methods you think would be interesting to also include.

Exercise - Functions

We're stretching the word function here, but these would effectively be methods that do not use object data specifically but could be considered nonetheless. A useful tool for an Integer class may be to convert a String to an int. Java of course already have this and it can be called by using '`Integer.parseInt("-9324374")`'. This can be found at '*http://docs.oracle.com/javase/7/docs/api/java/lang/Integer.html*'.

These are done by implementing '`static`' methods, methods that do not need to be invoked on an object but can be done straight from the class. An example for our dog class might be we want to check whether a colour is a valid colour for a dog. Implement the following method:

```
public static boolean validColour(String
col){
   /* TODO: Write this code here.  */
}
```

Resources & Further Reading

'*http://homepages.herts.ac.uk/~db12aba/*' – All content from these sessions updated weekly.

'*http://code.org/*' – A good resource testing your programming skills.

'*http://stackoverflow.com/*' – Highly recommended online help for programmers (NOTE: Employers are interested to know whether you're an active member of this site!).

'*http://draw.io*' – A very good, free online drawing tool that exports to many formats, including '`XML`' and '`JPG`'.