**Objects**

In the inherited Online E Learning course an object is described as a dog. It has **properties** like name, colour, breed and age. It has **behaviours** like barking, eating and running.

Using that anology, dogs are members of a general type or class - the class of dogs.

A **class**is a type, while an **object**is an instance, the actual object that can be manipulated. The difference between classes and objects is critical. You cannot pet the class (the concept dog) but you can pet the object (the instance) of a dog.

Direct quote from E Learning:

“The act of creating an object is called **instantiation**. Programmers talk about instantiating an object. This means that they create an object of a given type as defined by the class definition.

In OOP, a class defines a blueprint for a type of object. The characteristics and behaviours that belong to a class are jointly referred to as **members** of that class. The characteristics (in the dog example, the name, age, breed, and colour) are called **properties** of the class and are represented as variables; the behaviours (barking, eating, and running) are called **methods** of the class, and are represented as functions.”

**Inheritance** is a benefit of OOP. You can **extend** a class to create a subclass letting you reuse code.

When you extend a class, the subclass **inherits** all the properties and methods of the original class. Typically the subclass defines additional methods and properties **overriding** methods or properties.

In the ELearning example, you could build a superclass called Animal, which contains common characteristics and behaviours of all animals. Next, you could build several subclasses that inherit from the Animal superclass and add characteristics and behaviours specific to that type of animal.

**Polymorphism** is an object taking on many forms. With polymorphism, classes can override methods of their superclasses and define specialized methods.

In the ELearning example, “you might start with a class called Mammal that has play() and sleep() methods. You then create Monkey and Dog subclasses to extend the Mammal class. The subclasses override the play() method from the Mammal class to reflect the habits of those particular kinds of animals. Monkey implements the play() method to swing from trees and Dog implements the play() method to fetch a ball. Because the sleep() functionality is similar among the animals, you would use the superclass implementation.” (Direct quote from Inherited content”)

**Encapsulation** is combining elements to create a new entity. Encapsulation implements the idea of “create something good and reuse it.”

In today’s review, I’ve included links to online lessons and explanations beginning with easy basic and leading to full discussion that might take place at the university level. (Something for everyone, I hope.)

One of my fav books is “Head’s First” Series, if you can get your hands on this, here is the github bulletpoint summary of the inheritance chapter:

<https://gist.github.com/raskasa/4411043>

Links I’ve put up today:

(From beginner to Advanced)

<https://beginnersbook.com/2013/03/inheritance-in-java/>

2nd

<http://www.informit.com/articles/article.aspx?p=1021579>

3rd

<https://stackify.com/oops-concepts-in-java/>

4th

<https://www.ntu.edu.sg/home/ehchua/programming/java/J3b_OOPInheritancePolymorphism.html>

5th

<https://www.protechtraining.com/content/java_fundamentals_tutorial-object_oriented>