

# 5COSC001W - OBJECT ORIENTED PROGRAMMING

## Lecture 1: Object Oriented Programming and Some Java Fundamentals

Dr Dimitris C. Dracopoulos

# Programming Paradigms

- ▶ Procedural programming
- ▶ Functional programming
- ▶ Logic Programming
- ▶ Object-oriented programming

# Why Object Oriented Programming?

- ▶ Easy
- ▶ Powerful
- ▶ Many languages with similar syntax (Java, C++, C#)
- ▶ Popular (Job Market)

# Java vs C++

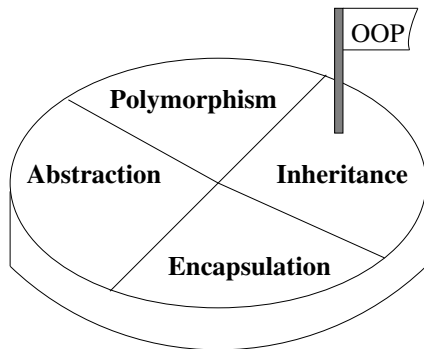
Many commonalities. However they are different programming languages. Some of the most important differences are:

- ▶ C++ supports operator overloading. Operators like `+`, `-`, `*`, etc. can be redefined to work with user defined types.
- ▶ Java has automatic memory management (garbage collector). There is no need to deallocate types allocated in the heap
- ▶ All Java objects are allocated dynamically (in the heap).
- ▶ Java does not support multiple inheritance.

# Major Characteristics of Object Oriented Programming

Four of the major characteristics of the Object Oriented Paradigm are:

1. Abstraction
2. Polymorphism
3. Inheritance
4. Encapsulation



## Abstraction

In the process of solving a problem:

- ▶ A particular representation of the solution must be chosen.
- ▶ The representation (*object*) of such a component contains the important characteristics (data) of the component, and the allowed operations on them which are necessary for the particular situation.

*Example:*

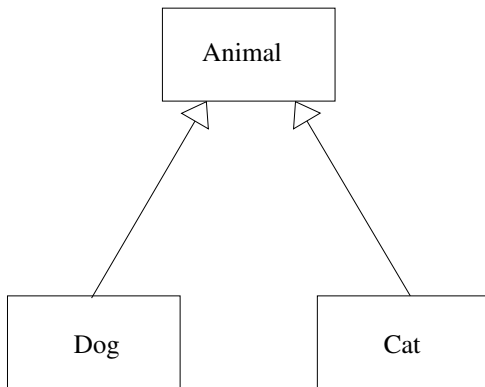
To model a car in a particular problem, only the size of the car and its colour are important. Therefore, to represent a car in this specific problem, a class holding information about the size of the car and its colour is needed only.

Such a class is created and it is an abstraction of a real car, because all the other characteristics of a real-world car are not needed and therefore not modelled.

## Inheritance

*Inheritance* organises classes in hierarchies.

- ▶ It is based on how similar the functionality of classes is.

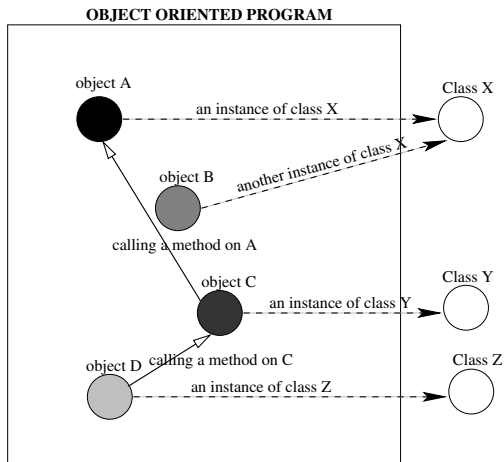


## Characteristics of Inheritance

- ▶ A class A which inherits from a class B is called the child of B. B is called the parent class of A. In other words, A is derived from B.
- ▶ A child class inherits all the fields and methods of its parent class.
- ▶ A child class should always be consistent with the “is a” relationship with the parent class. For example, a Dog “is an” Animal.

# Classes and Objects (Structure of an Object Oriented Program)

- ▶ An object oriented program has multiple instances (objects) of various classes.
- ▶ The objects interact with each other by sending signals to each other (i.e. calling methods on other objects).



## Primitive types and Objects

A Java program has primitive types and user defined types (classes). Instances of classes are called objects.

```
String greeting = "Hello World"; // object assigned to a variable  
int i = 15; // a primitive type (int) assigned to a variable
```

- Objects are created using the new operator. However, String objects are a special case, as they can also be created by simply enclosing a number of characters in double quotes.

```
String message = new String("First week of lectures");
```

## Calling Methods on Objects

A class defines methods which can be called on specific objects of the class.

For example, method `length` is defined in the library class `String` and it can be called for any object of the class:

```
String greeting = "Hello World";  
int n = greeting.length();  
System.out.println("n=" + n);
```

```
String message = new String("First week of lectures");  
n = message.length();  
System.out.println("n=" + n);
```

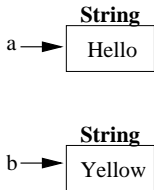
The above segment of code displays:

`n=11`

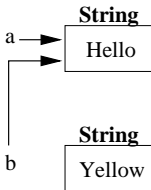
`n=22`

## Assigning an object reference variable to another variable

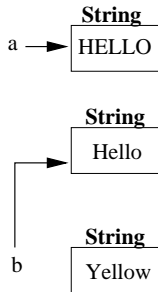
- ▶ Variables which are assigned objects actually store the address of the object.
- ▶ This means that assigning such a variable to another, will not duplicate the object but will create an additional reference to the initial object.
- ▶ Note that the assignment operator is the single equals (=) sign. This is different than the double equals sign (==) used to test for equality. *It is a common programming mistake, to use the single equals sign when testing for equality.*



1) After: `String a = "Hello";`  
`String b = "Yellow";`



2) After: `b = a;`



3) After:  
`a = a.toUpperCase();`