CSC 110 2.0
Object Oriented Programming

Tute 02
Classes, Objects, Constructors, Methods and Access Modifiers

Dulan S. Dias © 2019
Class

A class is an entity that determines how an object will behave and what the object will contain.

In other words, it is a blueprint or a set of instruction to build a specific type of object.
Class

Syntax

class <class_name>{
    field;
    method;
}

Dulan S. Dias © 2019 - www.dulandias.com
Object

An object a basic unit of Object Oriented Programming and represents the real life entities, and we call this as an instance of a class.

Syntax

```
ClassName ReferenceVariable = new ClassName();
```
Exercise

Let's take an example of developing a pet management system, specially meant for dogs.

You will need various information about the dogs like different breeds of the dogs, the age, size, etc.

You need to model real-life beings, i.e., dogs into software entities.
Exercise - Approach to the Solution

You can see a picture of three different breeds of dogs below.

List down the differences between them.
Exercise - Approach to the Solution

Some of the differences you might have listed out maybe breed, age, size, color, etc.

If you think for a minute, these differences are also some common characteristics shared by these dogs.

These characteristics (breed, age, size, color) can form a data members for your object.
Exercise - Approach to the Solution

List out the common behaviors of these dogs like sleep, sit, eat, etc. So these will be the actions of our software objects.
Exercise - Approach to the Solution

To sum up what we have understood so far,

- **Class** - Dogs
- **Data members** - size, age, color, breed, etc.
- **Methods** - eat, sleep, sit and run.
Exercise - Approach to the Solution

Now, for different values of data members (breed size, age, and color) in Java class, you will get different dog objects.
Constructors

A constructor in Java is a special method that is used to initialize objects. The constructor is called when an object of a class is created.

It can be used to set initial values for object attributes.
Constructors - Example

// Create a MyClass class

public class MyClass {

    int x;    // Create a class attribute

    // Create a class constructor for the MyClass class

    public MyClass() {

        x = 5;    // Set the initial value for the class attribute x

    }

}
Constructors - Example

```java
public class MyClassTest {

    public static void main(String[] args) {

        MyClass myObj = new MyClass(); // Create an object of class MyClass (This will call the constructor)

        System.out.println(myObj.x); // Print the value of x

    }

}
```
Exercise

Create a class Dog as previously discussed in the exercise, with a constructor to initialize the data members/fields to empty strings.

In the same class, create another constructor to initialize all the data members/fields by passing parameters to the constructor.

Create a main method in a Driver Class, to create two objects using the two constructors we created above.
Exercise

This is an example for the concept of Overloading in Object Oriented Programming.
Methods

A method is a block of code which only runs when it is called.

You can pass data, known as parameters, into a method.

Methods are used to perform certain actions, and they are also known as functions.

Why use methods? To reuse code: define the code once, and use it many times.
Methods

```java
class Main {
    public static void main(String[] args) {
        ... ...
        myFunction();
        ... ...
    }
}

private static void myFunction() {
    // function body
    ... ...
    ... ...
}
```
Methods with Arguments and Return Value

class SquareMain {
    public static void main(String[] args) {
        ... ... ...
        n = 3;
        result = square(n);
        ... ... ...
    }

    private static int square(int i) {
        // return statement
        return i*i;
    }
}

Access Modifiers

**public:** When a member of a class is modified by `public`, then that member can be accessed by any other code.

**private:** When a member of a class is specified as `private`, then that member can only be accessed by other members of its class.
Access Modifiers

Now you can understand why main( ) has always been preceded by the public modifier.

It is called by code that is outside the program—that is, by the Java run-time system.

**default:** When no access modifier is used, then by default the member of a class is public within its own package, but cannot be accessed outside of its package.

**protected** applies only when **inheritance** is involved.
## Access Modifiers - Summary

<table>
<thead>
<tr>
<th>Access Modifier</th>
<th>within class</th>
<th>within package</th>
<th>outside package by subclass only</th>
<th>outside package</th>
</tr>
</thead>
<tbody>
<tr>
<td>Private</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Default</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Protected</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>Public</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
</tbody>
</table>
Exercise

What should be the access modifier of a constructor of a class which we intend to create objects from a Driver Class?

Create Getters and Setters for the data members/fields of the Dog class we created in the previous exercise with appropriate access modifiers.