**Java Classes and Objects**

**Java Classes/Objects**

Java is an object-oriented programming language.

Everything in Java is associated with classes and objects, along with its attributes and methods. For example: in real life, a car is an object. The car has **attributes**, such as weight and color, and **methods**, such as drive and brake.

A Class is like an object constructor, or a "blueprint" for creating objects.

**Create a Class**

To create a class, use the keyword class:

**MyClass.java**

Create a class called "MyClass" with a variable x:

public class MyClass {  
  int x = 5;  
}

Remember from the [Java Syntax chapter](https://www.w3schools.com/java/java_syntax.asp) that a class should always start with an uppercase first letter, and that the name of the java file should match the class name.

**Create an Object**

In Java, an object is created from a class. We have already created the class named MyClass, so now we can use this to create objects.

To create an object of MyClass, specify the class name, followed by the object name, and use the keyword new:

**Example**

Create an object called "myObj" and print the value of x:

public class MyClass {  
  int x = 5;  
  
  public static void main(String[] args) {  
    MyClass **myObj** = new MyClass();  
    System.out.println(myObj.x);  
  }  
}

**Multiple Objects**

You can create multiple objects of one class:

**Example**

Create two objects of MyClass:

public class MyClass {  
  int x = 5;  
  
  public static void main(String[] args) {  
    MyClass **myObj1** = new MyClass();  // Object 1  
    MyClass **myObj2** = new MyClass();  // Object 2  
    System.out.println(myObj1.x);  
    System.out.println(myObj2.x);  
  }  
}

**Using Multiple Classes**

You can also create an object of a class and access it in another class. This is often used for better organization of classes (one class has all the attributes and methods, while the other class holds the main() method (code to be executed)).

Remember that the name of the java file should match the class name. In this example, we have created two files in the same directory/folder:

* MyClass.java
* OtherClass.java

**MyClass.java**

public class MyClass {  
  int x = 5;  
}

**OtherClass.java**

class OtherClass {  
  public static void main(String[] args) {  
    MyClass **myObj** = new MyClass();  
    System.out.println(myObj.x);  
  }  
}

When both files have been compiled:

C:\Users\*Your Name*>javac MyClass.java  
C:\Users\*Your Name*>javac OtherClass.java

Run the OtherClass.java file:

C:\Users\*Your Name*>java OtherClass

And the output will be:

5