CORE JAVA – CLASS & OBJECT

Lecture Notes

Softech Solutions Inc.

www.softechsolutionsgroup.com

saney.alam@ softechsolutionsgroup.com

JAVA – CLASS & OBJECT

Table of Contents		***Page***
Class & Object		
	What is class in Java?	***02***
	What is Object in Java?	***02***
	Creating a Class	***02***
	Declaring a variable of class	***02***
	Class Member Variables	***03***
	Initializing a Class object	***04***
	Methods of a Class	***04***
Class Constructors		
	What is Class Constructor?	***06***
	Passing values to the constructor	***07***
	Assigning default values to Constructor	***09***
	Why do we need Constructor?	***10***
String Class		
	What is String?	***12***
	String Manipulations	***13***

Class & Object

What is Class in Java?

Classes are the templates for the creation of objects. When we say template, it means it always have the same structure and implementation. They set out rules for how the objects that they contain may behave. So all the objects created from a single class share the same behavior and state. The difference between a class and an object is that class is created when the program is created but the objects are created at the run time.

For an example if we create a class 'Car' and create an object 'Toyota' from that car class, then we can say that the Toyota car object is an instance of the class of objects known as Car. Toyota car have some states and behavior. However each brand of car will have its own state and behaviors.

What is an Object in Java?

In Object Oriented Programming (OOP), an object is an instance of a class. An object can be considered as a thing, which performs a set of activities. An object can be anything and in real world an object can be a dog, table, window, car etc.
But objects always share two characteristics, they all have state and they all have behaviors. For example a car has state (make, brand, color, and model) and have behavior (braking, accelerating, slowing down, changing gears).
The same way an object in any programming language too has state and behavior. A software object maintains its state in variables and implements its behavior with methods. So from the above example of car object, we can say that the object is a collection of methods (behavior – braking, accelerating, slowing down, and changing gears) and variables (state – make, brand, color, and model).

Creating a Class

A class can have just one variable or a group of variable. To create a class you start with the class keyword followed by a name and its body delimited by curly brackets.

```
class Car{

}
```

Declaring a Variable of class

In previous chapter we have learned how to create a variable. It works same with creating a class but in place of any data type just uses the class name. Like above we create a class ***Car***, now we can easily create a variable of above class.

```
public class ClassTestExercise {

        public static void main(String[] args) {
                //Use the Car keyword to declare Car Class variable
                Car Toyota;

                //Initialize the Toyota variable with a new Car object
                Toyota = new Car();
                }
        }
```

You can also do the creating and initializing of Class variable on a same line.

```
Car Toyota = new Car();
```

Class Member Variables

The curly braces '***{ and }***', of a class is referred to as its **body**. So that variables declared in the body of the class are called **member variables**. In java these ***member variables*** are called **fields**. The ***fields*** can be any ***type.***

```
public class Car {
        //Class Member Variables & Fields
        String sModel;
   int iGear;
   int iHighestSpeed;
   String sColor;
   int iMake;
   boolean bLeftHandDrive;
   String sTransmission;
   int iTyres;
   int iDoors;

}
```

Initializing a Class Object

After ***declaring*** an ***instance*** of a class, you can access each of its ***members*** and assign it the desired value.

```
public class Car {
        //Class Member Variables & Fields
        String sModel;
   int iGear;
   int iHighestSpeed;
   String sColor;
   int iMake;
   boolean bLeftHandDrive;
   String sTransmission;
   int iTyres;
   int iDoors;
}
```

Methods of a Class

So far we have seen that while creating a class, fields are meant to describe it. In our case the class Car is described with its gear, color, make, brand etc. Apart from these characteristics, a car object can also perform some actions. These actions are referred as methods of a class.

A method can produce some value or return some value. If the method is returning some value then the type of value that a method can provide (or return) is written on the left side of the method name. But when it does not, then method is to be declared as void method.

```
public class Car {
        //Class Member Variables & Fields
        String sModel;
   int iGear;
   int iHighestSpeed;
   String sColor;
   int iMake;
   boolean bLeftHandDrive;
   String sTransmission;
   int iTyres;
   int iDoors;

   public void DisplayCharacterstics(){
        System.out.println("Model of the Car: " + sModel);
        System.out.println("Number of gears in the Car: " + iGear);
        System.out.println("Max speed of the Car: " + iHighestSpeed);
        System.out.println("Color of the Car: " + sColor);
        System.out.println("Make of the Car: " + iMake);
        System.out.println("Transmission of the Car: " + sTransmission);

   }
}
```

After creating a method in its body delimited by its curly brackets, you can define the desired behaviors and once you are done with that, you can use that method by typing the class name followed by dot and the method name.

ClassName.MethodName();

```
public class ClassTestExecise {

        public static void main(String[] args) {
                //Use the Car keyword to declare Car Class variable
                Car Toyota = new Car();;

                Toyota.bLeftHandDrive = true;
                Toyota.iDoors = 4;
                Toyota.iGear = 5;
                Toyota.iHighestSpeed = 200;
                Toyota.iMake = 2014;
                Toyota.iTyres = 4;
                Toyota.sColor = "Black";
                Toyota.sTransmission = "Manual";
                Toyota.sModel = "Camry";

                //Using Car class method
                Toyota.DisplayCharacterstics();
                }
        }
```

Output:

Model of the Car: Camry
Number of gears in the Car: 5
Max speed of the Car: 200
Color of the Car: Black
Make of the Car: 2014
Transmission of the Car: Manual

Class Constructor

What is Class Constructor?

Constructor is a method which is use to set initial values for field variables. In Java when the object is created, compiler calls the constructor first. It means any code written in the constructor will then get executed. On top of it, there is no need to make any special calls to a constructor method – it happens automatically at the run time when the compiler creates a new object.

Constructor methods take the same name as the class.

```
public class Car {
            //Class Member Variables & Fields
            String sModel;
   int iGear;
   int iHighestSpeed;
   String sColor;
   int iMake;
   boolean bLeftHandDrive;
   String sTransmission;
   int iTyres;
   int iDoors;

   //This is the syntax to create a Constructor
   public Car(){

   }
}
```

So, if you have noticed in the above example, the name of the constructor is 'Car'. Which is exactly the same name as the class itself? Unlike normal methods, class constructors don't need a return type like int or double, nor any return value.

Note: If we do not explicitly write a constructor for a class the Java compiler builds a default constructor for that class.

Passing values to the constructor

However, values can be passed to the constructor.

Car Class

```
public class Car {
        //Class Member Variables & Fields
        String sModel;
   int iGear;
   int iHighestSpeed;
   String sColor;
   int iMake;
   boolean bLeftHandDrive;
   String sTransmission;
   int iTyres;
   int iDoors;

   //Constructor with values passed
   public Car(String Model, int Make,boolean LeftHandDrive ){
        sModel = Model;
        iMake = Make;
        bLeftHandDrive = LeftHandDrive;
   }

   //Method
   public void DisplayCharacterstics(){
        System.out.println("Model of the Car: " + sModel);
        System.out.println("Number of gears in the Car: " + iGear);
        System.out.println("Max speed of the Car: " + iHighestSpeed);
        System.out.println("Color of the Car: " + sColor);
        System.out.println("Make of the Car: " + iMake);
        System.out.println("Transmission of the Car: " + sTransmission);

   }

}
```

Main Method

```
public class ConstructorTestExercise {

        public static void main(String[] args) {
                //Use the Car keyword to declare Car Class variable
        //Passing values to the constructor
                Car Toyota = new Car("Camry",2014,true);

                Toyota.iDoors = 4;
                Toyota.iGear = 5;
                Toyota.iHighestSpeed = 200;
                Toyota.iTyres = 4;
                Toyota.sColor = "Black";
                Toyota.sTransmission = "Manual";

                //Using Car class method
                Toyota.DisplayCharacterstics();

        }

}
```

Above code will produce:

Model of the Car: Camry
Number of gears in the Car: 5
Max speed of the Car: 200
Color of the Car: Black
Make of the Car: 2014
Transmission of the Car: Manual

Assigning default values to the Constructor

However, it's a good idea to just set some default values to the field variables. These values will then be assigned when the object is created. Add the following code in the constructor:

Car class:

```
public class Car {
        //Class Member Variables & Fields
        String sModel;
    int iGear;
    int iHighestSpeed;
    String sColor;
    int iMake;
    boolean bLeftHandDrive;
    String sTransmission;
    int iTyres;
    int iDoors;

    //Default values set in Constructor
    public Car(){
        sModel = "Camry";
        iMake = 2014;
        bLeftHandDrive = true;
    }

    public void DisplayCharacterstics(){
        System.out.println("Model of the Car: " + sModel);
        System.out.println("Number of gears in the Car: " + iGear);
        System.out.println("Max speed of the Car: " + iHighestSpeed);
        System.out.println("Color of the Car: " + sColor);
        System.out.println("Make of the Car: " + iMake);
        System.out.println("Transmission of the Car: " + sTransmission);

    }

}
```

Main Method

```
public class ConstructorTestExercise {

        public static void main(String[] args) {
                //Use the Car keyword to declare Car Class variable
                Car Toyota = new Car();;

                Toyota.iDoors = 4;
                Toyota.iGear = 5;
                Toyota.iHighestSpeed = 200;
                Toyota.iTyres = 4;
                Toyota.sColor = "Black";
                Toyota.sTransmission = "Manual";

                //Using Car class method
                Toyota.DisplayCharacterstics();

        }

}
```

Above code will produce again the same result:

Model of the Car: Camry
Number of gears in the Car: 5
Max speed of the Car: 200
Color of the Car: Black
Make of the Car: 2014
Transmission of the Car: Manual

Why do we need Constructor?

There can be a situation where exposing the class variable to the main program is not secure. At that point of time, class variables can be turned to private because the private variables are not accessible from the other classes. In this situation if the constructors are defined, then the main method would not need to access class variables directly.

Car class:

```
public class Car {
        //Class Member Variables & Fields
   //Private variables
   private String sModel;
   private int iMake;
   private int iGear;
   int iHighestSpeed;
   String sColor;
   boolean bLeftHandDrive;
   String sTransmission;
   int iTyres;
   int iDoors;

   //Default values set in Constructor
   public Car(){
        sModel = "Camry";
        iMake = 2014;
        iGear = 5;
   }

   public void DisplayCharacterstics(){
        System.out.println("Model of the Car: " + sModel);
        System.out.println("Number of gears in the Car: " + iGear);
        System.out.println("Max speed of the Car: " + iHighestSpeed);
        System.out.println("Color of the Car: " + sColor);
        System.out.println("Make of the Car: " + iMake);
        System.out.println("Transmission of the Car: " + sTransmission);

   }

}
```

Main Method

```
public class ConstructorTestExercise {

        public static void main(String[] args) {
                //Use the Car keyword to declare Car Class variable
                Car Toyota = new Car();;

                Toyota.iDoors = 4;
                Toyota.iHighestSpeed = 200;
                Toyota.iTyres = 4;
                Toyota.sColor = "Black";
                Toyota.sTransmission = "Manual";
                Toyota.bLeftHandDrive = true;

                //Using Car class method
                Toyota.DisplayCharacterstics();

        }

}
```

Another situation can be where few class variables are very important for the class object and programmer wants to make sure that so ever use the class and creates the object of the class, does not forget to initialize these important variables.

String Class

What is String in Java?

String is nothing but a sequence of character. String is a Class and not a data type like 'int & char'. So the String class represents the characters of String and all string objects in java are the instance of this class. String is the most usable class in java programming and in Selenium WebDriver.

The most important fact about String class that it is an immutable class, which means once it is created a string object, cannot be changed.

Creating String

The way we declare the any other data type, the same way we declare String as well. String is always being described in the double quotes.

```
String WebSite = "www.technosolution.org"
```

As with any other object, you can also create String objects by using the new keyword and a constructor.

```
char[] webSiteName = { 'T', 'E', 'C', 'H', 'N', 'O', 'S', 'O', 'L', 'U', 'T', 'I', 'O', 'N'};
String webSite = new String(webSiteName);
System.out.println( webSite );
```

This would produce: ***TECHNOSOLUTION***

String Manipulations

As we have already discussed above that string is the most important class in the Java and it has got many methods. You will be using all these methods very frequently in Selenium Automation or Java Programming.

String Length:

This *length* method returns the ***number of characters*** in the string. It count every alphanumeric char, special char or even spaces.

```
public class StringTestExercise {

        public static void main(String[] args) {
                //Declaring the String and Int Variable
                String sPopularTopic = "Selenium Automation Framework";
                int ilength = sPopularTopic.length();

                //Print the value of String variable & int Variable
                System.out.println("Popular Topic of Techno Solution: " +
sPopularTopic);
                System.out.println("Length of the Popular Topic: " +  ilength);

        }
```

The above will produce:

Popular Topic of Techno Solution: Selenium Automation Framework
Length of the Popular Topic: 29

Concatenating String

String class gives you the method to ***join*** two strings.

```
public class StringTestExercise {

	public static void main(String[] args) {
		//Declaring the String Variables
		String sPopularTopic_1 = "Selenium Automation Framework";
		String sPopularTopic_2 = "Basic Java Tutorial";
		String sSpace = " ";

		//Print the value of Concat String
		//String1.concat(String2);
		System.out.println("First Output");
		System.out.println(sPopularTopic_1.concat(sPopularTopic_2));

		//Another way of using String Concat
		//"String1".concat(String2);
		System.out.println("Second Output");
		System.out.println("Selenium Automation Framework".concat(sPopularTopic_2));

		//Another way of using String Concat
		//"String1" + "String2";
		System.out.println("Third Output");
		System.out.println("Selenium Automation Framework" + "Basic Java Tutorial");

		//Another way of using String Concat
		//"String1" + String + "String2";
		System.out.println("Fourth Output");
		System.out.println("Selenium Automation Framework" +sSpace+ "Basic Java
Tutorial");

	}

}
```

The above code would produce:

Selenium Automation FrameworkBasic Java Tutorial
Second Output
Selenium Automation FrameworkBasic Java Tutorial
Third Output
Selenium Automation FrameworkBasic Java Tutorial
Fourth Output
Selenium Automation Framework Basic Java Tutorial

Various String Methods

Let's take an example and cover all the important String operation.

```
public class StringTestExercise {

	public static void main(String[] args) {
		//Declaring the String
		String sPopularTopic_1 = "Selenium Automation Framework";
		String sPopularTopic_2 = "Basic Java Tutorial";

		//Compare two String: This would compare two strings
		//If the twos strings are equal, it will return 'true' else it will return 'false'
		boolean bCompareResult = sPopularTopic_1.equals(sPopularTopic_2);
		System.out.println("The result of String Comparison is : " + bCompareResult);

		//Character at: This would return the single character at index value from the String
		char cIndex = sPopularTopic_1.charAt(5);
		System.out.println("The fifth character of Popular Topic 1 is : " + cIndex);

		//Contains: This would return 'true' if the passed string is contained in the another
String
		boolean bContainResult = sPopularTopic_1.contains("Framework");
		System.out.println("The result of string Framework is contained in the String
sPopularTopic_1 is : " + bContainResult);

		//Index of: This would return the starting index of the passed string
		int iIndex = sPopularTopic_1.indexOf("Framework");
		System.out.println("The start index of the string Framework is : " + iIndex);

		//Sub String First index: This would return the sub string of the string from the passed
index number
		String sSubString = sPopularTopic_1.substring(iIndex);
		System.out.println("The sub string from the index number is : " + sSubString);

		//Sub String First & Last index: This would return the sub string from first index to end
index
		sSubString = sPopularTopic_1.substring(8, 19);
		System.out.println("The sub string of Popular Topic 1 from index 8 to 18 is : " +
sSubString);

		//To Lower Case: It would return the complete string in the lower case
		String sLowerCase = sPopularTopic_1.toLowerCase();
		System.out.println("The lower case of the Popular Topic 1 is : " + sLowerCase);

		//Split: It breaks the string in to two parts from the passed argument and store it in to
array
		String [] aSplit = sPopularTopic_1.split("Automation");
		System.out.println("The first part of the array is : " + aSplit[0]);
		System.out.println("The last part of the array is : " + aSplit[1]);

	}
}
```

The above code would produce:

The result of String Comparison is : false
The fifth character of Popular Topic 1 is : i
The result of string Framework is contained in the String sPopularTopic_1 is : true

The start index of the string Framework is : 20
The sub string from the index number is : Framework
The sub string of Popular Topic 1 from index 8 to 18 is : Automation
The lower case of the Popular Topic 1 is : selenium automation framework
The first part of the array is : Selenium
The last part of the array is : Framework

Still there are many more methods are left with String class, take a look over all the methods, so that you would be able to know what you are capable of doing with String object. When you type any string variable and type dot, it populates all the available methods of the object.