Object Oriented Programming with Java
Practical-3

Introduction to Objects and Classes in Java

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`:

Main.java

Create a class named "`Main`" with a variable x:

```
public class Main {
  int x = 5;
}
```

Create an Object

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

To create an object of `Main`, 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 Main {
  int x = 5;

  public static void main(String[] args) {
    Main myObj = new Main();
    System.out.println(myObj.x);
  }
}
```

Multiple Objects

You can create multiple objects of one class:

Example

Create two objects of `Main`:

```
public class Main {
  int x = 5;

  public static void main(String[] args) {
    Main myObj1 = new Main();  // Object 1
    Main myObj2 = new Main();  // Object 2
    System.out.println(myObj1.x);
    System.out.println(myObj2.x);
  }
}
```

Java 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:

Example

Create a constructor:

```
// Create a Main class
public class Main {
  int x;  // Create a class attribute

  // Create a class constructor for the Main class
  public Main() {
    x = 5;  // Set the initial value for the class attribute x
  }

  public static void main(String[] args) {
    Main myObj = new Main(); // Create an object of class Main (This will call the constructor)
    System.out.println(myObj.x); // Print the value of x
  }
}

// Outputs 5
```

Note that the constructor name must **match the class name**, and it cannot have a **return type** (like `void`).

Also note that the constructor is called when the object is created.

All classes have constructors by default: if you do not create a class constructor yourself, Java creates one for you. However, then you are not able to set initial values for object attributes.

Constructor Parameters

Constructors can also take parameters, which is used to initialize attributes.

The following example adds an `int y` parameter to the constructor. Inside the constructor we set x to y (x=y). When we call the constructor, we pass a parameter to the constructor (5), which will set the value of x to 5:

Example

```
public class Main {
  int x;

  public Main(int y) {
    x = y;
  }

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

// Outputs 5
```

You can have as many parameters as you want:

Example

```
public class Main {
  int modelYear;
  String modelName;

  public Main(int year, String name) {
    modelYear = year;
    modelName = name;
  }

  public static void main(String[] args) {
    Main myCar = new Main(1969, "Mustang");
    System.out.println(myCar.modelYear + " " + myCar.modelName);
  }
}

// Outputs 1969 Mustang
```

String Constructor

String is a sequence of characters. In java, objects of String are immutable which means a constant and cannot be changed once created.
Creating a String

There are two ways to create string in Java:

- *String literal*

```
String s = “DepartmentofMCA”;
```

- **Using *new* keyword**

```
String s = new String (“DepartmentofMCA”);
```

String() Constructs a new empty String.

String(String) Constructs a new String that is a copy of the specified String.

String(char[]) Constructs a new String whose initial value is the specified array of characters.

String(char[], int, int) Constructs a new String whose initial value is the specified sub array of characters.

String(byte[], int, int) Constructs a new String whose initial value is the specified sub array of bytes.

String(byte[], int) Constructs a new String whose initial value is the specified array of bytes.

Java String Methods

charAt(int) Returns the character at the specified index.

compareTo(String) Compares this String to another specified String.

concat(String) Concatenates the specified string to the end of this String.

copyValueOf(char[], int, int) Returns a String that is equivalent to the specified character array.

copyValueOf(char[]) Returns a String that is equivalent to the specified character array.

endsWith(String) Determines whether the String ends with some suffix.

equals(Object) Compares this String to the specified object.

equalsIgnoreCase(String) Compares this String to another object.

getChars(int, int, char[], int) Copies characters from this String into the specified character array.

indexOf(String) Returns the index within this String of the first occurrence of the specified substring.

indexOf(String, int) Returns the index within this String of the first occurrence of the specified substring.

lastIndexOf(int, int) Returns the index within this String of the last occurrence of the specified character.

lastIndexOf(String) Returns the index within this String of the last occurrence of the specified substring.

length() Returns the length of the String.

replace(char, char) Converts this String by replacing all occurences of oldChar with newChar.

startsWith(String) Determines whether this String starts with some prefix.

toCharArray() Converts this String to a character array.

toLowerCase() Converts all of the characters in this String to lower case.

toString() Converts this String to a String.

toUpperCase() Converts all of the characters in this String to upper case.

trim() Trims leading and trailing whitespace from this String.

valueOf(Object) Returns a String that represents the String value of the object.