Day-7

Static

The static can be:

1. Static Variables
2. Static Methods
3. Static Blocks Of Code.

Let’s look at static variables and static methods first.

What is Static Variable in Java?

Static variable in Java is variable which belongs to the class and initialized only once at the start of the execution. It is a variable which belongs to the class and not to object(instance). Static variables are initialized only once, at the start of the execution. These variables will be initialized first, before the initialization of any instance variables.

- A single copy to be shared by all instances of the class
- A static variable can be accessed directly by the class name and doesn’t need any object

  Syntax:
- `<class-name>.<variable-name>`

Example: How to call static variables & methods
Step 1) Copy the following code into a editor

```
public class Demo{
  public static void main(String args[]){
    Student s1 = new Student();
    s1.showData();
    Student s2 = new Student();
    s2.showData();
    //Student.b++;
    //s1.showData();
  }
}

class Student {
```

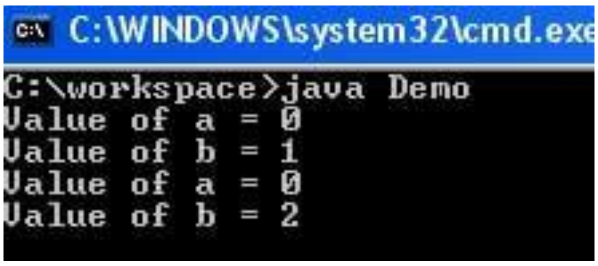
```
int a; //initialized to zero
static int b; //initialized to zero only when class is loaded not for each object created.

  Student(){
   //Constructor incrementing static variable b
   b++;
  }

   public void showData(){
      System.out.println("Value of a = "+a);
      System.out.println("Value of b = "+b);
   }
//public static void increment(){
//a++;
//}

}
```

Expected output show below



```
C:\workspace>java Demo
Value of a = 0
Value of b = 1
Value of a = 0
Value of b = 2
```

What is Static Block in Java?

The static block is a block of statement inside a Java class that will be executed when a class is first loaded into the JVM. A static block helps to initialize the static data members, just like constructors help to initialize instance members.

```
class Test{
 static {
 //Code goes here
 }
}
```

Following program is the example of java static block.

Example: How to access static block

```
public class Demo {
 static int a;
 static int b;
 static {
    a = 10;
    b = 20;
 }
 public static void main(String args[]) {

  System.out.println("Value of a = " + a);
  System.out.println("Value of b = " + b);

            }
}
```

you will get following output of the program.

```
Value of a = 10
Value of b = 20
```

Static Method
In Java, public methods belong to an instance of class but if we talk about static method, they belong to a class not to an instance of a class. There is no need of creating an instance of the class to invoke a static method. The static member can access only the static data member and can change its value.

Non-static Method
All the methods without having static keyword before method name are referred to as Non-static methods. There is no need to create an instance of the class for accessing the static method and static variable. The non-static methods are used dynamic or runtime binding. Unlike static method, we can override the non-static method.

Objects and Classes in Java

learn about Java objects and classes. In object-oriented programming technique, we design a program using objects and classes.

An object in Java is the physical as well as a logical entity, whereas, a class in Java is a logical entity only.

What is an object in Java

An entity that has state and behavior is known as an object e.g., chair, bike, marker, pen, table, car, etc. It can be physical or logical (tangible and intangible). The example of an intangible object is the banking system.

For Example, Pen is an object. Its name is Reynolds; color is white, known as its state. It is used to write, so writing is its behavior.

An object is an instance of a class. A class is a template or blueprint from which objects are created. So, an object is the instance(result) of a class.

Object Definitions:

- An object is a real-world entity.
- An object is a runtime entity.
- The object is an entity which has state and behavior.
- The object is an instance of a class.

What is a class in Java

A class is a group of objects which have common properties. It is a template or blueprint from which objects are created. It is a logical entity. It can't be physical.

A class in Java can contain:

- Fields
- Methods
- Constructors
- Blocks
- Nested class and interface

Syntax to declare a class:

```
class <class_name>{

    field;

    method;

}
```

Object and Class Example: main within the class

In this example, we have created a Student class which has two data members id and name. We are creating the object of the Student class by new keyword and printing the object's value.

Here, we are creating a main() method inside the class.

File: Student.java

```java
//Java Program to illustrate how to define a class and fields
//Defining a Student class.
class Student{
 //defining fields
 int id;//field or data member or instance variable
 String name;
 //creating main method inside the Student class
 public static void main(String args[]){
  //Creating an object or instance
  Student s1=new Student();//creating an object of Student
  //Printing values of the object
  System.out.println(s1.id);//accessing member through reference variable
  System.out.println(s1.name);
 }
}
```

Output:

```
0
null
```

Java String class methods

The java.lang.String class provides many useful methods to perform operations on sequence of char values.

No.	Method	Description
1	char charAt(int index)	It returns char value for the particular index
2	int length()	It returns string length
3	static String format(String format, Object... args)	It returns a formatted string.

4	static String format(Locale l, String format, Object... args)	It returns formatted string with given locale.
5	String substring(int beginIndex)	It returns substring for given begin index.
6	String substring(int beginIndex, int endIndex)	It returns substring for given begin index and end index.
7	boolean contains(CharSequence s)	It returns true or false after matching the sequence of char value.
8	static String join(CharSequence delimiter, CharSequence... elements)	It returns a joined string.
9	static String join(CharSequence delimiter, Iterable<? extends CharSequence> elements)	It returns a joined string.
10	boolean equals(Object another)	It checks the equality of string with the given object.
11	boolean isEmpty()	It checks if string is empty.
12	String concat(String str)	It concatenates the specified string.
13	String replace(char old, char new)	It replaces all occurrences of the specified char value.
14	String replace(CharSequence old, CharSequence new)	It replaces all occurrences of the specified CharSequence.
15	static String equalsIgnoreCase(String another)	It compares another string. It doesn't check case.
16	String[] split(String regex)	It returns a split string matching regex.
17	String[] split(String regex, int limit)	It returns a split string matching regex and limit.
18	String intern()	It returns an interned string.

19	int indexOf(int ch)	It returns the specified char value index.
20	int indexOf(int ch, int fromIndex)	It returns the specified char value index starting with given index.
21	int indexOf(String substring)	It returns the specified substring index.
22	int indexOf(String substring, int fromIndex)	It returns the specified substring index starting with given index.
23	String toLowerCase()	It returns a string in lowercase.
24	String toLowerCase(Locale l)	It returns a string in lowercase using specified locale.
25	String toUpperCase()	It returns a string in uppercase.
26	String toUpperCase(Locale l)	It returns a string in uppercase using specified locale.
27	String trim()	It removes beginning and ending spaces of this string.
28	static String valueOf(int value)	It converts given type into string. It is an overloaded method.

Java Arrays

Normally, an array is a collection of similar type of elements which has contiguous memory location.

Java array is an object which contains elements of a similar data type. Additionally, The elements of an array are stored in a contiguous memory location. It is a data structure where we store similar elements. We can store only a fixed set of elements in a Java array.

Array in Java is index-based, the first element of the array is stored at the 0th index, 2nd element is stored on 1st index and so on.

Types of Array in java

There are two types of array.

- **Single Dimensional Array**
- **Multidimensional Array**

Single Dimensional Array in Java

Syntax to Declare an Array in Java

```
dataType[] arr; (or)
dataType []arr; (or)
dataType arr[];
```

Instantiation of an Array in Java

```
arrayRefVar=new datatype[size];
```

Example of Java Array

Let's see the simple example of java array, where we are going to declare, instantiate, initialize and traverse an array.

```
//Java Program to illustrate how to declare, instantiate, initialize
//and traverse the Java array.
class Testarray{
public static void main(String args[]){
int a[]=new int[5];//declaration and instantiation
a[0]=10;//initialization
a[1]=20;
a[2]=70;
a[3]=40;
a[4]=50;
//traversing array
for(int i=0;i<a.length;i++)//length is the property of array
System.out.println(a[i]);
}}
```

Output:

```
10
20
70
40
50
```

Multidimensional Array in Java

In such case, data is stored in row and column based index (also known as matrix form).

Syntax to Declare Multidimensional Array in Java

```
dataType[][] arrayRefVar; (or)
dataType [][]arrayRefVar; (or)
dataType arrayRefVar[][]; (or)
dataType []arrayRefVar[];
```

Example to instantiate Multidimensional Array in Java

```
int[][] arr=new int[3][3];//3 row and 3 column
```

Example to initialize Multidimensional Array in Java

```
arr[0][0]=1;
arr[0][1]=2;
arr[0][2]=3;
arr[1][0]=4;
arr[1][1]=5;
arr[1][2]=6;
arr[2][0]=7;
arr[2][1]=8;
arr[2][2]=9;

```

Example of Multidimensional Java Array

Let's see the simple example to declare, instantiate, initialize and print the 2Dimensional array.

```
//Java Program to illustrate the use of multidimensional array
```

```java
class Testarray3{
public static void main(String args[]){
//declaring and initializing 2D array
int arr[][]={{1,2,3},{2,4,5},{4,4,5}};
//printing 2D array
for(int i=0;i<3;i++){
 for(int j=0;j<3;j++){
   System.out.print(arr[i][j]+" ");
 }
 System.out.println();
}
}}
```

Output:

```
1 2 3
2 4 5
4 4 5
```