Week-3

Static keyword

In Java, **static keyword** is mainly used for memory management

It can be used with variables, methods, blocks and nested classes.

It is a keyword which is used to share the same variable or method of a given class.

Basically, static is used for a constant variable or a method that is same for every instance of a class. The main method of a class is generally labeled static.

In order to create a static member (block, variable, method, nested class), you need to precede its declaration with the keyword ***static***. When a member of the class is declared as static, it can be accessed before the objects of its class are created, and without any object reference.

Static block

Java supports a special block, called static block which can be used for static variable initializations of a class. This code inside static block is executed only once. Static block calling before main method

Syntax

static

{

//static block

}

Static Variable

When you declare a variable as static, then a single copy of the variable is created and divided among all objects at the class level. Static variables are global variables. Basically, all the instances of the class share the same static variable. Static variables can be created at class-level only.

Generally Static variables are called by

Classname.variablename;

Static method

When a method is declared with the ***static*** keyword, it is known as a static method. The most common example of a static method is the ***main( )*** method.

We can access static methods using classname

Static methods can be called by classname.methodname()

Syntax

Classname.methodname();

Static keyword example

```
//static  Block Level will execute before main
// static varaible can be called by classname.variable
//staticmethod canbe called by classname.methodname()
//static  Block Level will execute before main
class AccountHolder
{
long accountNumber;
static String bankName="SBI";
static long pinCode=530016L;
        static
        {
        System.out.println("library files loading");
        }
        static void pinCode()
        {
        System.out.println(pinCode);
        }

public static void main(String args[])
{
System.out.println(AccountHolder.bankName);
AccountHolder.pinCode();
}
```

}

Output

javac AccountHolder.java

java AccountHolder

library files loading

SBI

530016

Week-3 Programs

3.a) Create a class Box that uses a parameterized constructor to initialize the dimensions of a box. The dimensions of the Box are width, height, depth. The class should have a method that can return the volume of the box. Create an object of the Box class and test the functionalities.

```
import java.util.Scanner;
class Box {
   // Attributes to store dimensions of the box
    double width;
    double height;
    double depth;

   // Parameterized constructor to initialize dimensions
    Box(double width, double height, double depth) {
      this.width = width;
      this.height = height;
      this.depth = depth;
   }
```

```
    // Method to calculate the volume of the box
     double getVolume() {
        return width * height * depth;
    }

    public static void main(String[] args) {
        // Creating an object of the Box class
Scanner sc=new Scanner(System.in);

System.out.print("Enter width:");
double w=sc.nextDouble();
System.out.print("Enter height:");
double h=sc.nextDouble();
System.out.print("Enter depth:");
double d=sc.nextDouble();

        Box box = new Box(w,h,d);

        // Testing the functionality
        System.out.print("Volume of the box: " + box.getVolume());
    }
}
```

Output

```
javac Box.java
java Box
Enter width:3.5
```

Enter height:4.5

Enter depth:5.6

Volume of the box: 88.19999999999999

3.b) Create a new class called Calculator with the following methods:

a. A static method called powerInt(int num1,int num2)

i. This method should return num1 to the power num2.

b. A static method called powerDouble(double num1,double num2).

i. This method should return num1 to the power num2.

c. Invoke both the methods and test the functionality. Also count the number of objects created.

```
class Calculator {
   // Static variable to keep track of the number of objects created
    static int objectCount = 0;

   // Constructor that increments the object count
    Calculator() {
      objectCount++;
   }

   // Static method to calculate the power of two integers
    static int powerInt(int num1, int num2) {
      return (int) Math.pow(num1, num2); // Using Math.pow() to calculate power
```

```
    }

    // Static method to calculate the power of two doubles
     static double powerDouble(double num1, double num2) {
        return Math.pow(num1, num2); // Using Math.pow() to calculate power
    }

    // Static method to get the number of objects created

    public static void main(String[] args) {
        // Test the static methods without creating any object
        int resultInt = Calculator.powerInt(2, 3);  // 2^3 = 8
        double resultDouble = Calculator.powerDouble(2.5, 3.2);  // 2.5^3.2

        // Output the results
        System.out.println("2^3 using powerInt: " + resultInt);
        System.out.println("2.5^3.2 using powerDouble: " + resultDouble);

        // Creating objects to check the object count
        Calculator calc1 = new Calculator();
        Calculator calc2 = new Calculator();

        // Output the number of objects created
        System.out.println("Number of Calculator objects created: " + Calculator.objectCount);
    }
}
```

Output

javac Calculator.java

java Calculator

2^3 using powerInt: 8

2.5^3.2 using powerDouble: 18.767569280959865

Number of Calculator objects created: 2