Lab 06
Static Member and Methods

Objective(s):

1. Static Member
2. Static Methods
3. Types of Static Methods
4. Static Code
5. Lab tasks
6. Post Lab questions

1: Static Members

When a variable is declared as static, then a single copy of the variable is created and shared among all objects at the class level. Static variables are, essentially, global variables. All instances of the class share the same static variable. Static variables and methods in Java provide several advantages, including memory efficiency, global access, object independence, performance, and code organization.

- The static variable gets memory only once in the class area at the time of class loading.

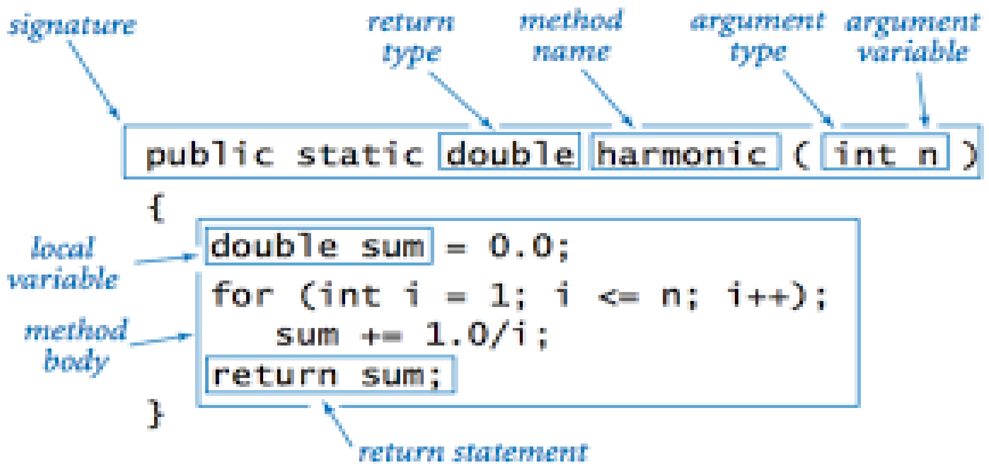
```
//Java Program to demonstrate the use of static variable
class Student{
  int rollno;//instance variable
  String name;
  static String college ="ITS";//static variable
  //constructor
  Student(int r, String n){
  rollno = r;
  name = n;
  }
  //method to display the values
  void display (){System.out.println(rollno+" "+name+" "+college);}
}
//Test class to show the values of objects
public class TestStaticVariable1{
 public static void main(String args[]){
 Student s1 = new Student(111,"Karan");
 Student s2 = new Student(222,"Aryan");
 //we can change the college of all objects by the single line of code
 //Student.college="BBDIT";
 s1.display();
 s2.display();
 }
}
```

2: Static Methods

A method (function) is a group of statements that is executed when it is called from some point of the program.

Static methods are the methods in Java that can be called without creating an object of class. They are referenced by the class name itself or reference to the Object of that class. Static method belongs to the class rather than the object of a class

The following is its format:



Where:

- ***return_type*** is the data type specifier of the data returned by the function.
- ***method_name*** is the identifier by which it will be possible to call the function.
- ***parameters*** (as many as needed): Each parameter consists of a data type specifier followed by an identifier, like any regular variable declaration (for example: int x) and which acts within the function as a regular local variable. They allow to pass arguments to the function when it is called. The different parameters are separated by commas.
- ***statements*** is the function's body. It is a block of statements surrounded by {}

```
//Java Program to demonstrate the use of a static method.
class Student{
int rollno;
String name;
static String college = "ITS";
//static method to change the value of static variable
static void change(){
college = "BBDIT";
}
//constructor to initialize the variable
Student(int r, String n){
rollno = r;
```

```java
name = n;
}
//method to display values
void display(){System.out.println(rollno+" "+name+" "+college);}
}
//Test class to create and display the values of object
public class TestStaticMethod{
public static void main(String args[]){
Student.change();//calling change method
//creating objects
Student s1 = new Student(111,"Karan");
Student s2 = new Student(222,"Aryan");
Student s3 = new Student(333,"Sonoo");
//calling display method
s1.display();
s2.display();
s3.display();
}
}
```

Restrictions

a. The static method cannot use non static data member or call non-static method directly.
b. this and super cannot be used in static context.

Why is the Java main method static?

It is because the object is not required to call a static method. If it were a non-static method, JVM creates an object first then call main() method that will lead the problem of extra memory allocation

2: Type of static Methods

Methods without Return Type and No Parameter

```
public static void method_name ( )
```

What we will do in this case for similar functionality code, create a function named drawLine() and call that function at repeated code locations.

```java
public class Lab04 {
    public static void main(String [] args){
    drawLine();
    System.out.println("Object Oriented Programming");
```

```
     drawLine();
     System.out.println("Lab 04");
     drawLine();
     System.out.println("Department of Computer Sciences");
     drawLine();
     }
    public static void drawLine(){
       for(int i=1;i<30;i++)
         System.out.print("*");
         System.out.println();
     }
}
```

Methods without Return Type and with Parameters

```
public static void method_name ( par1, par2, par3 )
```

drawLine method with parameter to draw line according to the size provided in an argument.

```
public class Lab04 {
public static void main(String [] args){
        drawLine(30);
        System.out.println("Object Oriented Programming");
        drawLine(10);
        System.out.println("Lab 04");
        drawLine(10);
        System.out.println("Department of Computer Sciences");
        drawLine(30);
    }
public static void drawLine(int n){
   for(int i=1;i<n;i++)
        System.out.print("*");
        System.out.println();
    }
}
```

Methods With Return Type but no Parameter

```
public static return_type method_name ( )
public class GetNameExample {
  public static void main(String [] args) {
    System.out.println("The  University Name is "+getUniversityName());
  }//main ends
  public static String getUniversityName(){
    return "Sukkur IBA University";
  }
}//class ends
```

Methods With Return Type and Parameters

```
public static return_type method_name (par1, par2, par3)
```

```
public class ArraySum {
   public static void main(String [] args) {
      int [] a = {2,3,5,8,4,9,7,6,7,8};
      int sum = findSum(a);//invoking method with array argument
      System.out.println("The result is "+sum);
   }//main ends
   public static int findSum(int[] b){
      int total=0;
      for(int i=0; i<b.length;i++){
          total+=b[i]; return total;
      }
   }
}//class ends
```

There are two types of parameter passing:

1. **Pass by Value**

```
public static void sum(int a, int b)
```

- Call By value (copy of value) is when primitive data types are passed in the method call.

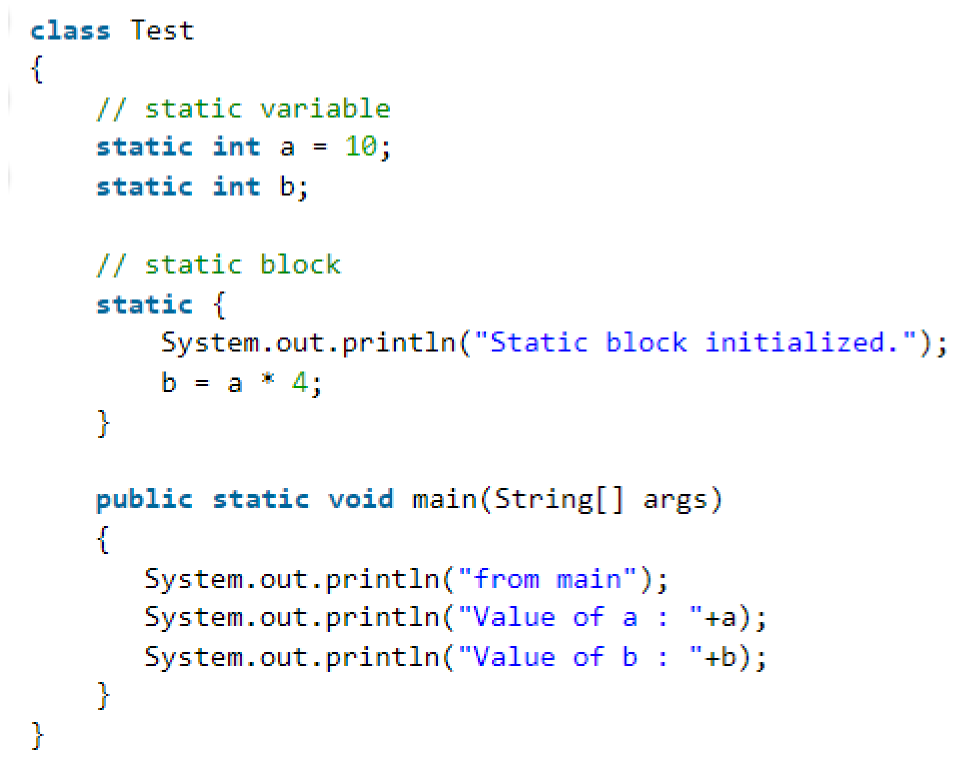
2. **Pass by Reference (value of memory address location)**

```
public static void displayCars(Car car)
```

- Objects and object variables are passed by reference or address.

3: Static Block

These are a block of codes with a static keyword. In general, these are used to initialize the static members. JVM executes static blocks before the main method at the time of class loading.



```
class Test
{
    // static variable
    static int a = 10;
    static int b;

    // static block
    static {
        System.out.println("Static block initialized.");
        b = a * 4;
    }

    public static void main(String[] args)
    {
       System.out.println("from main");
       System.out.println("Value of a : "+a);
       System.out.println("Value of b : "+b);
    }
}
```

Lab Tasks

Exercise1: You are developing a system to manage student information for a school. Each student has a name, age, and grade. Additionally, you need to keep track of the total number of students enrolled in the school. You want to ensure that this count is automatically updated whenever a new student object is created.

1. Design a Java class named **Student** with private instance variables for **name**, **age**, and **grade**.
2. Implement setter methods to set the values for these variables.
3. Implement getter methods to retrieve the values of these variables.
4. Add a static variable **studentCount** to keep track of the total number of students.

5. Implement a static method **getStudentCount()** to retrieve the total student count.
6. Use a static initialization block to initialize **studentCount** to 0.
7. Modify the constructor of the **Student** class to increment **studentCount** whenever a new student object is created.
8. Create class studentDemo to create a three new student object, set their information, and retrieve the total student count. Exercise 2
   Book.java

Exercise 2: You are creating a library management system to track borrowed books. Each book has a title, author, publication year, and a late fee rate applicable to all books. You want to ensure that the late fee rate remains the same for all books and cannot be modified per book

Design a Java class named Book with private instance variables for title, author, and publicationYear.

Implement setter methods to set the values for title, author, and publicationYear.

Implement getter methods to retrieve the values of title, author, and publicationYear.

Implement a static method setLateFeeRate(double lateFeeRate) to set the late fee rate applicable to all books.

Implement a static method getLateFeeRate() to retrieve the late fee rate.

Use a static initialization block to initialize the late fee rate to a default value.

Provide an example of how you would use this class to create a new book, set its information, set the late fee rate, and calculate the late fee for a borrowed book.

Exercise 3: You are tasked with creating a program to manage bank accounts for a financial institution. Each bank account has an account number, account holder name, balance, and an annual interest rate. You want to ensure that the interest rate is the same for all accounts and cannot be modified by individual account holders.

1. Design a Java class named BankAccount with private instance variables for accountNumber, accountHolderName, balance, and annualInterestRate.

2. Implement setter methods to set the values for accountNumber, accountHolderName, and balance.

3. Implement getter methods to retrieve the values of accountNumber, accountHolderName, and balance.

4. Implement a static method setAnnualInterestRate(double rate) to set the annual interest rate for all accounts.

5. Implement a static method getAnnualInterestRate() to retrieve the annual interest rate.

6. Use a static initialization block to initialize the annual interest rate to a default value.

7. Implement a method calculateInterest() to calculate the interest earned by the account based on its balance and the annual interest rate. This method should return the calculated interest amount.

8. Create another class to use this class and create five bank account, set its information, set the annual interest rate, deposit money into the account, and calculate the interest earned.

Exercise 4:

1. Design a Java class named Employee with private instance variables for employeeID, name, position, and salary.
2. Implement a constructor to initialize these variables.
3. Implement setter and getter methods for these variables.
4. Implement a static method named **promoteEmployee(Employee employee, String newPosition)** that takes an employee object and a new position as parameters and returns a new employee object with the updated position.
5. Implement a static method named **calculateBonus(Employee employee, double bonusPercentage)** that takes an employee object and a bonus percentage as parameters and returns a new employee object with the updated salary after applying the bonus.

6. Provide an example of how you would use these classes to create a new employee, promote the employee to a new position, calculate the bonus for the employee, and retrieve employee information.

Exercise 5:

1. Design a Java class named Course with private instance variables for courseCode, title, instructor, and duration.
2. Implement a constructor to initialize these variables.
3. Implement setter and getter methods for these variables.
4. Implement a nested class named Student to represent a student enrolled in the course. This class should have private instance variables for studentID, name, and progress.
5. Implement a constructor for the Student class to initialize these variables.
6. Implement setter and getter methods for these variables in the Student class.
7. Implement a method in the Course class named enrollStudent(Student student) to enroll a student in the course.
8. Implement a method in the Course class named trackProgress(Student student, double progress) to track the progress of a student in the course.
9. Provide an example of how you would use these classes to create a new course, enroll students, track their progress, and retrieve course information.

Post lab questions to ponder

1. Can constructors be static in java? Try it out and justify.
2. Why use iterations when we have recursion and vice versa?
3. Can we overload by return type?
4. Can you have static classes in java? if yes then justify?

END