The Bombay Salesian Society's

Don Bosco Institute of Technology, Mumbai, 400070.

(Affiliated to University of Mumbai)

Lab Name: JAVA Lab

(ITL304) ODD Semester

Academic Year: 2024-25

Name:- Tanvi Rupesh Patil

Batch:- B

Roll No.:- 41

Subject In-charge: Prof. Tayyabali Sayyad

Department Of Information Technology

INDEX

Program No. 1

Write a java program to demonstrate to passed by value and pass by reference

PROGRAM :-

```
class PassByValue{
    void meth(int i,int j){
        i *= 2;
        j  /= 2;
    }
}

class PassByRef{
    int a;
    int b;
    PassByRef(int i, int j){
        a = i;
        b = j;
    }
    void meth2(PassByRef obj){
        obj.a *= 2;
        obj.b /= 2;
    }
}

public class PassByValueAndRef{
```

```
    public static void main(String[] args){

        PassByValue obj = new PassByValue();
        int a = 10;
        int b = 20;
        System.out.println("a & b before Call by value "+ a +" "+b);
        obj.meth(a,b);
        System.out.println("a & b After call Call by value "+ a +" "+b);
        System.out.println(" ");

        PassByRef obj2 = new PassByRef(10,20);
        System.out.println("a & b before call by Reference "+ obj2.a +" " +
obj2.b);
        obj2.meth2(obj2);
        System.out.println("a & b After call by Reference "+ obj2.a
        +" " + obj2.b);
    }
}
```

OUTPUT :-

```
java -cp /tmp/JyZpsMnPKr/Main
a & b before Call by value 10 20
a & b After call Call by value 10 20

a & b before call by Reference 10 20
a & b After call by Reference 20 10

=== Code Execution Successful ===
```

Program No. 2

Q2. Write a java program to demonstrate to demonstrate static variables, methods, and blocks.

PROGRAM :-

```
public class Static_Demo{
        static int a = 3;
        static int b;

        static void meth(int x){
        System.out.println("x = "+x);
        System.out.println("a = "+a);
        System.out.println("b = "+b);
    }

    static{
        System.out.println("Static Block Initiated");
        b = a * 4;
    }

    public static void main(String[] args){
        meth(42);
    }
}
```

OUTPUT :-

```
java -cp /tmp/dvlPsBDL2e/Main
Static Block Initiated
x = 42
a = 3
b = 12

=== Code Execution Successful ===
```

Program No. 3

Write a java program to demonstrate inner class and outer classes .

PROGRAM :-

```
class Outer{
   int outer_x=100;

   void test(){
      Inner inner=new Inner();
      inner.display();

   }

   class Inner{
      int y=10;
      void display(){
         System.out.println("display: outer_x is "+outer_x);
      }
   }

   void showy(){
      //Inner inner = new Inner();  // Create an instance of Inner
      //System.out.println("value of y is " + inner.y);
   }
}


public class InnerClassDemo{
   public static void main(String[] args){
      Outer outer=new Outer();
      outer.test();
      //outer.showy();
   }
}
```

OUTPUT :-

```
java -cp /tmp/07YDxUexRK/InnerClassDemo
display: outer_x is 100

=== Code Execution Successful ===
```

Program No. 4

Write a java program to demonstrate accessing private members in the sub class using public methods.

PROGRAM :-

```
class Test5{
        int a;
        public int b;
        private int c;

        void setc(int i){
                c=i;
        }

        int getc(){
                return c;
        }
}

public class AccessTest{
        public static void main(String [] args){
                Test5 ob = new Test5();
                ob.a=10;
                ob.b=20;
                ob.setc(100);

                System.out.println("a is "+ob.a+"    b is "+ob.b+"    c is
"+ob.getc());
        }
 }
```

OUTPUT :-

```
java -cp /tmp/ygFLWW9CpK/AccessTest
a is 10    b is 20    c is 100

=== Code Execution Successful ===
```

Program No. 5

Write a java program to demonstrate all usage of super keyword.

PROGRAM :-

```
class Parent {
   int a;
   int b;

   Parent(int x, int y) {
      a = x;
      b = y;
   }

   void display() {
      System.out.println("Parent A: " + a);
      System.out.println("Parent B: " + b);
   }
}

class Child extends Parent {
   int c;

   Child(int x, int y, int z) {
      super(x, y);
      c = z;
   }

   void display() {
      super.display();
      System.out.println("Child C: " + c);
   }
}

public class SuperUse {
   public static void main(String args[]) {
      Child child = new Child(10, 20, 30);
      child.display();
```

```
    }
}
```

OUTPUT :-

```
java -cp /tmp/ohHo9doqIz/SuperUse
Parent A: 10
Parent B: 20
Child C: 30

=== Code Execution Successful ===
```

Program No. 6

Write a java program to demonstrate dynamic method dispatch in the inheritance.

PROGRAM :-

```
class A7{
	void callme(){
		System.out.println("A's call");
	}
}

class B7 extends A7{
	void callme(){
		System.out.println("B's call");
	}
}

class C7 extends A7{
	void callme(){
		System.out.println("C's call");
	}
}

public class Dispatch{
	public static void main(String [] args){
		A7 a=new A7();
		B7 b=new B7();
		C7 c=new C7();

		A7 r;

		r=a;
		r.callme();

		r=b;
		r.callme();
```

```
        r=c;
        r.callme();
    }
}
```

Output :-

```
java -cp /tmp/soSqksJWnE/Dispatch
A's call
B's call
C's call

=== Code Execution Successful ===
```

Program no. 7

Write a java program to demonstrate shared constants in interfaces

PROGRAM :-

```
interface ShapeConstants {
   double PI = 3.14;
   int MAX_SHAPES = 10;
}

class Circle implements ShapeConstants {
   double radius;

   Circle(double r) {
      radius = r;
   }

   double area() {
      return PI * radius * radius;
   }
}

class Rectangle implements ShapeConstants {
   double length;
   double width;

   Rectangle(double l, double w) {
      length = l;
      width = w;
   }

   double area() {
      return length * width;
   }
}

public class SharedConstantsDemo {
```

```
    public static void main(String[] args) {
        Circle circle = new Circle(5);
        Rectangle rectangle = new Rectangle(4, 6);

        System.out.println("Area of the Circle: " + circle.area());
        System.out.println("Area of the Rectangle: " + rectangle.area());

        System.out.println("Maximum number of shapes allowed: " +
ShapeConstants.MAX_SHAPES);
    }
}
```

OUTPUT :-

```
java -cp /tmp/WDA5lrAkdr/SharedConstantsDemo
Area of the Circle: 78.5
Area of the Rectangle: 24.0
Maximum number of shapes allowed: 10

=== Code Execution Successful ===
```

Program no. 8

Write a java program to demonstrate default, public, private and protected scope in packages.

PROGRAM :-

```
package Package_A;

public class Class_A {
   public String publicField = "publicField";
   protected String protectedField = "protectedField";     String
DefaultField = "DefaultField";
   private String privateField = "privateField";

   public void publicmethod(){
      System.out.println("public method");
   }
   protected void protectedmethod(){
      System.out.println("protectedmethod");
   }
    void defaultmethod(){
      System.out.println("defaultmethod");
   }
   private void privatemethod(){
      System.out.println("privatemethod");
   }
   public void accessPrivateMembers(){
      System.out.println("accessing within class A:");
System.out.println(privateField);        publicmethod();
   } }
package
Package
_A;

public class Class_C {
```

```
   public void accessmembers(){

      Class_A classA = new Class_A();
      System.out.println("accessing members from class C");

      System.out.println(classA.publicField);
      System.out.println(classA.protectedField);
      System.out.println(classA.DefaultField);

      classA.publicmethod();
classA.protectedmethod();
      classA.defaultmethod();
   } }


package Package_B;
import Package_A.Class_A;

public class Class_B extends Class_A {     public
void accessmembers(){
System.out.println(publicField);
      System.out.println(protectedField);

      publicmethod();
      protectedmethod();
   } }


import Package_A.Class_A; import
Package_B.Class_B; import
Package_A.Class_C;

public class PackageDemo {

   public static void main(String[] args) {
      Class_A a = new Class_A();
      Class_B b = new Class_B();
      Class_C c = new Class_C();
```

```
        System.out.println("direct access from class_A");
System.out.println(a.publicField);        a.publicmethod();
        b.accessmembers();
        c.accessmembers();
        a.accessPrivateMembers();
    } }
```

Output :- direct access from

class_A publicField public method

publicField protectedField public

method protectedmethod

accessing members from class C

publicField protectedField

DefaultField public method

protectedmethod defaultmethod

accessing within class A:

privateField public

method

Process finished with exit code 0

Program no. 9

Write a java program print even and odd numbers using Multithreading.

PROGRAM :-

```
class EvenNumbers extends Thread {
   public void run() {
      for (int i = 0; i <= 20; i += 2) {
         System.out.println("Even: " + i);
         try {
            Thread.sleep(500);
         } catch (InterruptedException e) {
            System.out.println(e);
         }
      }
   }
}

class OddNumbers extends Thread {
   public void run() {
      for (int i = 1; i <= 20; i += 2) {
         System.out.println("Odd: " + i);
         try {
            Thread.sleep(500);
         } catch (InterruptedException e) {
            System.out.println(e);
         }
      }
   }
}

public class EvenOddThreadDemo {
   public static void main(String[] args) {
      EvenNumbers evenThread = new EvenNumbers();
      OddNumbers oddThread = new OddNumbers();

      evenThread.start();
```

```
        oddThread.start();
    }
}
```

OUTPUT :-

```
java -cp /tmp/FkXE5ysiyN/EvenOddThreadDemo
Even: 0
Odd: 1
Even: 2
Odd: 3
Even: 4
Odd: 5
Even: 6
Odd: 7
Even: 8
Odd: 9
Even: 10
Odd: 11
Even: 12
Odd: 13
Even: 14
Odd: 15
Even: 16
Odd: 17
Even: 18
Odd: 19
Even: 20

=== Code Execution Successful ===
```

Program no. 10

Write a java program to demonstrate how to read from a text file using FileReader and write to a text file using FileWriter.

PROGRAM :-

```
import java.io.FileReader; import
java.io.FileWriter;
import java.io.IOException;

public class Reader {
   public static void main(String[] args) {
      FileReader fr = null;
       try {
fr = new
FileReader("C:\\Users\\Parth\\IdeaProjects\\MiniProject_sem3_2024\\java
programs\\DBIT_JAVA\\src\\file.txt");
         int data;
         while ((data = fr.read()) != -1) {
System.out.print((char) data);
         }
      } catch (IOException e) {
         System.out.println("An error occurred: " + e.getMessage());
      } finally {

         if (fr != null) {
            try {
               fr.close();
            } catch (IOException e) {
               System.out.println("Error closing file: " + e.getMessage());
            }
         }
      }

      FileWriter fw = null;
try{
```

```
        fw = new FileWriter("output.txt");
        fw.write("Hello File");
    }catch(IOException e){
        System.out.println("An error occured: "+e.getMessage());
    }finally {

        if (fw != null) {
            try {
                fw.close();
            } catch (IOException e) {
                System.out.println("Error closing file: " + e.getMessage());
            }
        }
    }

} }
```

Output :- hi

friends!!

Process finished with exit code 0

Program no. 11

Write a java program to demonstrate reading a file using FileInputStream and writing to another file using FileOutputStream.

PROGRAM :-

```
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;

public class FileIOExample {
   public static void main(String[] args) {

      FileInputStream fis = null;
      try {
fis = new
FileInputStream("C:\\Users\\Parth\\IdeaProjects\\MiniProject_sem3_20
24\\ja va programs\\DBIT_JAVA\\src\\file.txt");            int data;
         System.out.println("Contents of file.txt:");
         while ((data = fis.read()) != -1) {
System.out.print((char) data);
         }
         System.out.println();
      } catch (IOException e) {
         System.out.println("An error occurred while reading the file:
" + e.getMessage());        } finally {

         if (fis != null) {
            try {
               fis.close();
            } catch (IOException e) {
               System.out.println("Error closing file input stream: " +
e.getMessage());
            }
         }
      }
```

```
        FileOutputStream fos = null;
try {
            fos = new FileOutputStream("output.txt");
            String dataToWrite = "Hello File using FileOutputStream!";
            fos.write(dataToWrite.getBytes());
            System.out.println("Data has been written to output.txt
successfully!");
        } catch (IOException e) {
            System.out.println("An error occurred while writing to the file:
" + e.getMessage());        } finally {

            if (fos != null) {
                try {
                    fos.close();
                } catch (IOException e) {
                    System.out.println("Error closing file output stream: " +
e.getMessage());
                }
            }
        }
    }
}
```

OUTPUT :-

Contents of file.txt:

hi friends!!

Data has been written to output.txt successfully!

Process finished with exit code 0