**MCA FIRST SEMSTER LAB PROGRAMS-2021**

**OBJECT ORIENTED PROGRAMMING USING JAVA**

**OOP USING JAVA LAB PROGRAMS-2021**

**LAB1:**

**AIM: Write a Java Program which performs sorting of group of integer values using bubble sort technique.**

**Procedure:**

Step 1: Start

Step 2: Declare i, j. num, temp variables as Integer datatype

Step 3: Take Scanner class and declare Input as a object

Step 4: Print “Enter the number of integers to sort:"

Step 5: Declare array using int array[] = new int[num]

Step 6: Read elements in to num variable. Declare array and take N elements into array.

Step 7: Having bubble sort logic as given below

Step 7.1: Using for loop for i=0 to i<num-1

Step 7.2 : Using for loop for j=0 to j<num-i-1

Step 8: if (array[j] > array[j+1])

temp = array[j]

array[j] = array[j+1]

array[j+1] = temp

Step 9: Print “Sorted list of integers”

Step 10: Stop

**Program:**

import java.util.Scanner;

class BubbleSortExample {

public static void main(String []args) {

int num, i, j, temp;

Scanner input = new Scanner(System.in);

System.out.println("Enter the number of integers to sort:");

num = input.nextInt();

int array[] = new int[num];

System.out.println("Enter " + num + " integers: ");

for (i = 0; i < num; i++)

array[i] = input.nextInt();

for (i = 0; i < ( num - 1 ); i++) {

for (j = 0; j < num - i - 1; j++) {

if (array[j] > array[j+1])

{

temp = array[j];

array[j] = array[j+1];

array[j+1] = temp;

}

}

}

System.out.println("Sorted list of integers:");

for (i = 0; i < num; i++)

System.out.println(array[i]);

}

}

**Output:**

Enter the number of integers to sort:

6

Enter 6 integers:

12

6

78

9

45

08

Sorted list of integers:

6

8

9

12

45

78

**LAB2:**

**AIM: Write a Java Program for testing a string whether it is a palindrome or not.**

**Procedure:**

Step1: start

Step2: : Declare str,rev variables as String datatype

Step 3: Take Scanner class and declare Input as a object

Step 4: Print “Enter a string"

Step 5: Take nextLine() method.

Step 6: Declare int length = str.length();

Step 7: Using for loop

for ( int i = length - 1; i >= 0; i-- )

rev = rev + str.charAt(i)

Step 8: if (str.equals(rev))

Step 9: Print “String is a palindrome

Step10: else, Print “Sring is not a palindrome.

Step 11: Stop

**Program:**

import java.util.Scanner;

class ChkPalindrome

{

public static void main(String args[])

{

String str, rev = "";

Scanner sc = new Scanner(System.in);

System.out.println("Enter a string:");

str = sc.nextLine();

int length = str.length();

for ( int i = length - 1; i >= 0; i-- )

rev = rev + str.charAt(i);

if (str.equals(rev))

System.out.println(str+" is a palindrome");

else

System.out.println(str+" is not a palindrome");

}

}

**Output:**

Enter a string:

radar

radar is a palindrome

**LAB 3:**

**AIM: Write a Java Program to illustrate constructors.**

**Procedure:**

Step1: start

Step2: Declare languages variable as Sting datatype.

Step3: Take constructor as Main().

Step4: Declare lang as String datatype in constructor.

Step5: if languages=lang , print “programming languages”

Step6: create three objects as obj1,obj2 and obj3.

Step7:Stop.

**Program:**

class Main

{

String languages;

Main(String lang)

{

languages = lang;

System.out.println(languages + " Programming Language");

}

public static void main(String[] args)

{

Main obj1 = new Main("Java");

Main obj2 = new Main("Python");

Main obj3 = new Main("C");

}

}

**Output**:

Java Programming Language

Python Programming Language

C Programming Language

**LAB 4:**

**AIM: Write a Java Program for a method without parameters but with return type.**

**Procedure:**

Step1: start

Step2: Declare radius as double datatype.

Step3: Take class name as Circle

Step4: Take method name as area().

Step5:  Decalre      double r = 7;

   double ar = 3.14 \* r \* r;

    return ar;

Step6: Take class name as Methodreturn

Step7: Create object “c” for Cirle class

Step8:  double area;

   area = c.area();

Step9:Print “Area of a circle”.

Step10: stop.

**Program:**

|  |
| --- |
|  |

|  |
| --- |
| class Circle  {         double radius;             double area()       {           double r = 7;     double ar = 3.14 \* r \* r;      return ar;       }  }    class Methodreturn  {            public static void main(String args[])      {                  Circle c = new Circle();                    double area;     area = c.area();     System.out.println("Area of circle is : " + area);      }  } |

**Output**

Area of circle is : 153.86

**LAB 5:**

**AIM:** Write a Java Program to illustrate overloading methods in Java.

**Procedure:**

Step1: Start

Step2: Take class name as Displayoverloading

Step3: Take method name as disp()

Step4: Use “c” as a character datatype in method as aparameter.

Step5: print” character”

Step6: Take method name as disp().

Step7: Use “c” as character datatype and “num” as a integer datatype in method as

two parameters.

Step8: Print “character and Number”.

Step9: Take main class name as Sample.

Step10: Create object “obj” for Displayoverloading class.

Step11: obj.disp('a');

obj.disp('a',10);

Step12: stop

**Program:**

class Displayoverloading

{

public void disp(char c)

{

System.out.println(c);

}

public void disp(char c, int num)

{

System.out.println(c + " "+num);

}

}

class Sample

{

public static void main(String args[])

{

Displayoverloading obj = new Displayoverloading();

obj.disp('a');

obj.disp('a',10);

}

}

**Output:**

a

a 10

**LAB 6:**

**AIM: Write a Java Program to illustrate overriding methods in Java.**

**Procedure:**

Step1: start

Step2: Take class name as Bank

Step3: Declare getRateOfInterest as a method name

Step4: return o

Step5: Take SBI as another class name.

Step6: return 8

Step7: Take ICICIas another class name.

Step8: return 7

Step9: Take AXIS as another class name.

Step10: return 9.

Step11: Take three objects as “s” for SBI class, “i” for ICICI class and “a” for

AXIS class.

Step12: Print Rateofinterest for SBI,ICICI and AXIS class.

Step13: Stop.

**Program:**

class Bank

{

int getRateOfInterest()

{

return 0;

}

}

class SBI extends Bank

{

int getRateOfInterest()

{

return 8;

}

}

class ICICI extends Bank

{

int getRateOfInterest()

{

return 7;

}

}

class AXIS extends Bank

{

int getRateOfInterest()

{

return 9;

}

}

class Test2

{

public static void main(String args[])

{

SBI s=new SBI();

ICICI i=new ICICI();

AXIS a=new AXIS();

System.out.println("SBI Rate of Interest: "+s.getRateOfInterest());

System.out.println("ICICI Rate of Interest: "+i.getRateOfInterest());

System.out.println("AXIS Rate of Interest: "+a.getRateOfInterest());

}

}

**Output:**

SBI Rate of Interest: 8

ICICI Rate of Interest: 7

AXIS Rate of Interest: 9

**LAB 7:**

**AIM:** Write a Java Program to illustrate the implementation of multiple inheritance using interfaces in Java.

**Procedure:**

Step1: start

Step2: Take class name as MotorBike.

Step3: Declare speed variable as integer datatype.

Step4: Declare totalDistance as a methodname.

Step5: use interface name as Cycle.

Step6: Declare distance as integer datatype

Step7: Take speed() as another methodname.

Step8: Use implements keyword inherit the two classes properties in another

class name as TwoWheeler.

Step9: Declare totaldistance, avgspeed as integer datatype.

Step10: totalDistance=speed\*distance;

Step11: Print “totaldistance”.

Step12: avgSpeed=totalDistance/speed;

Step13: Print “avgspeed”.

Step14: Declare “ t1” as a object for TwoWheeler class.

Step15: Stop.

**Program**

|  |
| --- |
| interface MotorBike  {  int speed=50;  public void totalDistance();  }  interface Cycle  {  int distance=150;  public void speed();  }  public class TwoWheeler implements MotorBike,Cycle  {  int totalDistance;  int avgSpeed;  public void totalDistance()  {  totalDistance=speed\*distance;  System.out.println("Total Distance Travelled : "+totalDistance);  }  public void speed()  {  int avgSpeed=totalDistance/speed;  System.out.println("Average Speed maintained : "+avgSpeed);  }  public static void main(String args[])  {  TwoWheeler t1=new TwoWheeler();  t1.totalDistance();  t1.speed();  }  } |

|  |
| --- |
|  |

**Output:**

Total Distance Travelled : 7500

Average Speed maintained : 150

**Lab 8:**

**AIM:** Write a Java Program to illustrate the implementation of abstract class.

**Procedure:**

Step1: start

Step2: Take class name as MyClass by using abstract keyword.

Step3: Declare speed variable as integer datatype.

Step4: Declare disp as a methodname.

Step5: Print “Concrete method of parent class “

Step6: Declare another methodname as disp2by using abstract keyword.

Step7: Take another class name as Demo.

Step8: Inside Demo class declare method name as a disp2.

Step9: Print “overriding abstract method”.

Step10: Take main method. Inside main method create an object for Demo

class

Step11: Demo obj = new Demo();

obj.disp2();

Step12: Stop.

**Program:**

abstract class MyClass

{

public void disp()

{

System.out.println("Concrete method of parent class");

}

abstract public void disp2();

}

class Demo extends MyClass

{

public void disp2()

{

System.out.println("overriding abstract method");

}

public static void main(String args[])

{

Demo obj = new Demo();

obj.disp2();

}

}

**Output:**

overriding abstract method

**LAB 9:**

**AIM:** Write a Java Program to access the super class method and instance variable by using super key word from sub class.

**Procedure:**

Step1: start

Step2: Take class name as Parent.

Step3: Inside Parent class Declare “x” variable as integer datatype and

assigns a value to 5.

Step4: Declare show as a methodname.

Step5: Print “Method of parent class using SuperKeyword “

Step6: Take another class name as Child.

Step7: Inside Child cass declare another integer variable as “x” and assigns a

value to 9.

Step8: Inside Child class declare method name as show.

Step9: Print “Instance Variable of Parent class using Super

Keyword :" + super.x ”.

Step10: Take main method. Inside main method create an object for Child

class

Step11: Child ob = new Child();

ob.show();

Step12: Stop.

**Program:**

class Parent

{

int x = 5;

void show()

{

System.out.println("Method of parent class using SuperKeyword");

}

}

class Child extends Parent

{

int x = 9;

void show()

{

System.out.println("Instance Variable of Parent class using Super

Keyword :" + super.x);

System.out.println("Instance variable of Child class :"+x);

super.show();

System.out.println("Method of Child class ");

}

public static void main(String args[])

{

Child ob = new Child();

ob.show();

}

}

**Output:**

$ javac Child.java

$ java Child

Instance Variable of Parent class using Super Keyword :5

Instance variable of Child class :9

Method of parent class using Super Keyword

Method of Child class

**LAB 10:**

**AIM:** Write a Java Program which tells the use of try, catch and finally block.

**Procedure:**

Step1: start

Step2: Take class name as Main.

Step3: Inside Main class nd take main mehod().

Step4: Use try , catch and finally keywords.

Step5: Inside try block, declare

int divideByZero = 5 / 0;

Step6: Inside catch block, declare as ArithmeticException , use an object “e”

Step7: Print “ArithmeticException => " + e.getMessage());”.

Step8: Use finally bock

Step9: Print “This is the finally block ”

Step10: Stop.

**Program**

class Main

{

public static void main(String[] args)

{

try

{

// code that generates exception

int divideByZero = 5 / 0;

}

catch (ArithmeticException e)

{

System.out.println("ArithmeticException => " + e.getMessage());

}

finally

{

System.out.println("This is the finally block");

}

}

}

**Output**

ArithmeticException => / by zero

This is the finally block

# LAB 11:

# AIM: Write a Java Program to perform single task by multiple

# threads?

**Procedure:**

Step1: start

Step2: Take class name as TestMultitasking1.

Step3: Declare speed variable as integer datatype and extends to thread class.

Step4: Declare run as a methodname.

Step5: Print “task one “

Step6: Declare main method name as main().

Step7: Create three objects for TestMultitasking1 class.

TestMultitasking1 t1=new TestMultitasking1();

   TestMultitasking1 t2=new TestMultitasking1();

   TestMultitasking1 t3=new TestMultitasking1();

Step8:   take start mehod and use dot operator.

t1.start();

   t2.start();

   t3.start();

Step9: Stop.

**Program:**

class TestMultitasking1 extends Thread

{

  public void run()

{

    System.out.println("task one");

  }

  public static void main(String args[])

{

  TestMultitasking1 t1=new TestMultitasking1();

  TestMultitasking1 t2=new TestMultitasking1();

  TestMultitasking1 t3=new TestMultitasking1();

  t1.start();

  t2.start();

  t3.start();

 }

}

**Output:**

task one

task one

task one

**LAB 12:**

**AIM**: Write Java Program to creates an applet with some background color and foreground color with a message. The message string is stored in msg and is displayed in paint() method.

**Procedure:**

Step1: start

Step2: Use import statement and import three packages as,

import java.applet.Applet;

import java.awt.Color;

import java.awt.Graphics; for applets,

Step3: Take class name as SetBackColor and extends to Applet class.

Step4: Declare method name as init().

Step5: setBackground(Color.cyan);

setForeground(Color.red);

Step6: Take another merhodname as paint()

Step7: Declare g.drawString("Hello Java",50,50);

Step8: Take another HTML file use applet code as

applet code="SetBackColor" width=200 height=200>

Step9: Stop

**Program:**

**SetBackColor.java**

import java.applet.Applet;

import java.awt.Color;

import java.awt.Graphics;

public class SetBackColor extends Applet

{

public void init()

{

setBackground(Color.cyan);

setForeground(Color.red);

}

public void paint(Graphics g)

{

g.drawString("Hello Java",50,50);

}

}

**SetBackColor.html**

<html>

<head>

<h1>Applets </h1>

</head>

<body>

<applet code="SetBackColor" width=200 height=200>

</applet>

</body>

</html>

**OUTPUT**  
  
javac SetBackColor.java

Appletviewer SetBackColor.html

****