List of Programs

Exercise 1

Method Overloading

Aim

Write a Java program to show method overloading.

Method Overloading

```
import java.util.Scanner;

class Triangle
{
  double a,b,c;

  Triangle()
  {
    this.a = 0;
    this.b = 0;
    this.c = 0;
  }

  Triangle(double a, double b, double c)
  {
    this.a = a;
    this.b = b;
    this.c = c;
  }

  Triangle(double b)
  {
    this.b = b;
  }

  //Overloaded Method
  double getArea()
  {
    double s = (a+b+c)/2;
    return Math.sqrt(s*(s-a)*(s-b)*(s-c));

  }

  //Overloaded Method
  double getArea(double h)
  {
    return (0.5*b*h);
  }
}
```

```
class Overloading
{
  public static void main(String[] args)
  {
    int choice;
    double a,b,c,h;
    Triangle t;
    System.out.println("1. Triangle with 3 sides known");
    System.out.println("2. Triangle with base and height known");
    System.out.println();
    System.out.println("Enter your choice: ");
    Scanner sc = new Scanner(System.in);
    choice = sc.nextInt();
    switch(choice)
    {
      case 1:
        System.out.println("Enter the three sides: ");
        a = sc.nextDouble();
        b = sc.nextDouble();
        c = sc.nextDouble();
        t = new Triangle(a,b,c);
        System.out.println("Area of triangle = " + t.getArea());
        break;
      case 2:
        System.out.println("Enter the base side: ");
        b = sc.nextDouble();
        t = new Triangle(b);
        System.out.println("Enter the height: ");
        h = sc.nextDouble();
        System.out.println("Area of triangle = " + t.getArea(h));
        break;
      default:
        System.out.println("Invalid Choice...");
    }
  }
}
```

Output

```
1. Triangle with 3 sides known
2. Triangle with base and height known

Enter your choice:
1
Enter the three sides:
12
12
12
Area of triangle = 62.353829072479584
```

Exercise 2

Inheritance

Aim

Write a Java program to show the implementation of inheritance.

Inheritance

```
class Shape
{
  String name;
  Shape(String name)
  {
    this.name = name;
  }
  void show()
  {
    System.out.println(this.name);
  }
}
class Rectangle extends Shape
{
  private int length,breadth;
  Rectangle(int l,int b, String name)
  {
    super(name);
    this.length = l;
    this.breadth = b;
  }
  int getArea()
  {
    return this.length*this.breadth;
  }
  int getPerimeter()
  {
    return 2*(this.length*this.breadth);
  }
}
class Circle extends Shape
{
  private double radius;
  Circle(double r, String name)
  {
    super(name);
    this.radius = r;
  }
```

```
  double getArea()
  {
    return 3.14*this.radius*this.radius;
  }
  double getPerimeter()
  {
    return 2*3.14*this.radius;
  }
}
class Cylinder extends Circle
{
  double height;
  Cylinder(double r, double h, String name)
  {
    super(r,name);
    this.height= h;
  }
  double getVolume()
  {
    double volume;
    volume = getArea()*height;
    return volume;
  }
}
class Inheritance
{
  public static void main(String[] args)
  {
    Rectangle r = new Rectangle(5,10,"R1");
    Circle c = new Circle(10,"C1");
    Cylinder cyl = new Cylinder(10,20,"CYL1");
    System.out.print("Rectangle ");
    r.show();
    System.out.println("Area of rectangle " + r.getArea());
    System.out.print("Circle ");
    c.show();
    System.out.println("Area of circle " + c.getArea());
    System.out.print("Cylinder ");
    cyl.show();
    System.out.println("Volume of cylinder " + cyl.getVolume());
  }
}
```

Output

```
Rectangle R1
Area of rectangle 50
Circle C1
Area of circle 314.0
Cylinder CYL1
Volume of cylinder 6280.0
```

Exercise 3

Method Overriding

Aim

Write Java Program to show method overriding. (Exercise to understand Polymorphism)

Method Overriding

```
class Shape
{
  String name;
  Shape(String name)
  {
    this.name = name;
  }
  void show()
  {
    System.out.println("Shape " + this.name);
  }

  void findArea()
  {
    System.out.println("No area");
  }
}

class Rectangle extends Shape
{
  private int length,breadth;

  Rectangle(int l,int b, String name)
  {
    super(name);
    this.length = l;
    this.breadth = b;
  }

  void findArea()
  {
    System.out.println(this.length*this.breadth);
  }

  void show()
  {
    System.out.println("Rectangle " + this.name);
  }
}
```

```
class Circle extends Shape
{
  private double radius;

  Circle(double r, String name)
  {
    super(name);
    this.radius = r;
  }
  void findArea()
  {
    System.out.println(3.14*this.radius*this.radius);
  }

  void show()
  {
    System.out.println("Circle " + this.name);
  }
}

class Overriding
{
  public static void main(String[] args)
  {
    Shape[] shapes =  new Shape[3];

    shapes[0] = new Shape("Shape 1");
    shapes[1] = new Rectangle(10,10,"Rectangle 1");
    shapes[2] = new Circle(10,"Circle 1");

    for(Shape s: shapes)
    {
      s.show();
      s.findArea();
    }
  }
}
```

Output

```
Shape Shape 1
No area
Rectangle Rectangle 1
100
Circle Circle 1
314.0
```

Exercise 4

Interface

Aim

Write a java program to implement interface.

Interface

```
import java.util.Scanner;

interface IStack
{
  void push(int item);
  int pop();
  void display();
}

class FixedStack implements IStack
{
  private int s[];
  private int top;

  FixedStack(int size)
  {
    s = new int[size];
    top = -1;
  }

  public void push(int item)
  {
    if(top == s.length - 1)
    {
      System.out.println("Stack Full...");
    }
    else
    {
      s[++top] = item;
    }
  }

  public int pop()
  {
    if(top == -1)
    {
      System.out.println("Stack Empty...");
      return 0;
    }
```

```
      else
      {
        return s[top--];
      }
    }
    public void display()
    {
      System.out.println("Current Stack");
      for(int i = top;i>=0;i--)
      {
        System.out.println(s[i]);
      }
    }
  }

  class DynamicStack implements IStack
  {
    private int s[];
    private int top;

    DynamicStack(int size)
    {
      s = new int[size];
      top = -1;
    }

    public void push(int item)
    {
      if(top == s.length - 1)
      {
        int temp[] = new int[s.length*2];
        for(int i=0;i<s.length;i++)
        temp[i] = s[i];
        s = temp;
      }
      s[++top] = item;
    }

    public int pop()
    {
      if(top == -1)
      {
        System.out.println("Stack Empty...");
        return 0;
      }
      else
      {
        return s[top--];
      }
    }
    public void display()
```

```
    {
      System.out.println("Current Stack");
      for(int i = top;i>=0;i--)
      {
        System.out.println(s[i]);
      }
    }
  }


  class Stack
  {
    void pushInto(IStack st)
    {
      int item;
      Scanner sc = new Scanner(System.in);
      System.out.println("Enter the item to be pushed");
      item = sc.nextInt();
      st.push(item);
    }

    void popFrom(IStack st)
    {
      System.out.println("Item popped = " + st.pop());
    }

    void display(IStack st)
    {
      st.display();
    }
  }

  class StackOperation
  {
    public static void main(String[] args)
    {
      int choice;
      Scanner sc = new Scanner(System.in);
      IStack is = null;
      Stack s = new Stack();

      System.out.println("Stack Types");
      System.out.println("************");
      System.out.println("1. Fixed Stack");
      System.out.println("2. Dynamic Stack");
      System.out.println("Enter your choice: ");
      choice = sc.nextInt();
      switch(choice)
      {
        case 1:
          System.out.println("Enter the size of the stack: ");
```

```
            int size = sc.nextInt();
            is = new FixedStack(size);
            break;
          case 2:
            is = new DynamicStack(5);
            break;
        }

        do
        {
          System.out.println("Stack Operations");
          System.out.println("*****************");
          System.out.println("1. Push");
          System.out.println("2. Pop");
          System.out.println("3. Display");
          System.out.println("4. Exit");
          choice = sc.nextInt();
          switch(choice)
          {
            case 1:
              s.pushInto(is);
              break;
            case 2:
              s.popFrom(is);
              break;
            case 3:
              s.display(is);
              break;
            case 4:
              System.out.println("Thank You...");
              break;
            default:
              System.out.println("Invalid Choice!");
          }
        }while(choice != 4);
      }
    }
```

Output

```
Stack Types
************
1. Fixed Stack
2. Dynamic Stack
Enter your choice:
1
Enter the size of the stack:
3
Stack Operations
*****************
1. Push
2. Pop
```

```
3. Display
4. Exit
1
Enter the item to be pushed
11
Stack Operations
*****************
1. Push
2. Pop
3. Display
4. Exit
1
Enter the item to be pushed
22
Stack Operations
*****************
1. Push
2. Pop
3. Display
4. Exit
3
Current Stack
22
11
Stack Operations
*****************
1. Push
2. Pop
3. Display
4. Exit
2
Item popped = 22
Stack Operations
*****************
1. Push
2. Pop
3. Display
4. Exit
3
Current Stack
11
Stack Operations
*****************
1. Push
2. Pop
3. Display
4. Exit
```

Exercise 5

Exception Handling

Aim

Write a java program that handles various exceptions. Use try, catch and finally statements.

Exception Handling

```
import java.util.Scanner;
class UnderAgeException extends Exception
{
  int age;
  UnderAgeException(int age)
  {
    this.age = age;
  }
  public String toString()
  {
    return "Your age is " + age + " years";
  }
}

class ExceptionsDemo
{
  public static void main(String[] args)
  {
    Scanner sc= new Scanner(System.in);
    System.out.println("Enter your age: ");
    int age = sc.nextInt();
    try
    {
      vote(age);
      System.out.println("Enter two numbers: ");
      int a = sc.nextInt();
      int b = sc.nextInt();
      double ans = a/b;
      System.out.println(a + "/" + b + " = "  + ans);
    }
    catch(UnderAgeException e)
    {
      System.out.println("Error: " + e);
    }
    catch(ArithmeticException e)
    {
      System.out.println("Error: " + e);
    }
```

```
      finally
      {
        System.out.println("Exiting...\nThank you...");
      }
    }
    static void vote(int age) throws UnderAgeException
    {
      if(age >= 18)
      {
        System.out.println("Welcome...");
      }
      else
      {
        throw new UnderAgeException(age);
      }
    }
}
```

Output

```
Enter your age:
15
Error: Your age is 15 years
Exiting...
Thank you...

Enter your age:
25
Welcome...
Enter two numbers:
30
3
30/3 = 10.0
Exiting...
Thank you...
```

Exercise 6

Threads using Runnable Interface

Aim

Write a java program to demonstrate threads using runnable interface.

Threads

```
class NewThread implements Runnable
{
  Thread t;
  NewThread()
  {
    t = new Thread(this,"Demo Thread");
    System.out.println("Child Thread: " + t);
  }
  public void run()
  {
    try
    {
      for(int i=5;i>=0;i--)
      {
        System.out.println("Child Thread: " + i);
        Thread.sleep(1000);
      }
    }
    catch(InterruptedException e)
    {
      System.out.println("Child Interrupted...");
    }
    System.out.println("Exiting child thread");
  }
}
class ThreadDemo
{
  public static void main(String args[])
  {
    NewThread nt = new NewThread();
    nt.t.start();
    try
    {
      for(int i=5;i>=0;i--)
      {
        System.out.println("Main Thread: " + i);
        Thread.sleep(2000);
      }
    }
```

```
      catch(InterruptedException e)
      {
        System.out.println("Main thread interrupted...");
      }
      System.out.println("Exiting main thread");
    }
  }
```

Output

```
Child Thread: Thread[Demo Thread,5,main]
Main Thread: 5
Child Thread: 5
Child Thread: 4
Main Thread: 4
Child Thread: 3
Child Thread: 2
Main Thread: 3
Child Thread: 1
Child Thread: 0
Main Thread: 2
Exiting child thread
Main Thread: 1
Main Thread: 0
Exiting main thread
```

Exercise 7

Multiple Threads

Aim

Write a java program that creates three threads. First thread displays “Good Morning” every one second, the second thread displays “Hello” every two seconds and the third thread displays “Welcome” every three seconds.

Multiple Threads

```
class MessageThread extends Thread
{
  String message;
  int timer;
  String name;
  MessageThread(String name, String message, int timer)
  {
    super(name);
    this.name = name;
    this.message = message;
    this.timer = timer;
  }
  public void run()
  {
    try
    {
      for(int i=0;i<5;i++)
      {
        System.out.println(this.message);
        Thread.sleep(this.timer);
      }
    }
    catch(InterruptedException e)
    {
      System.out.println("Child Interrupted...");
    }
    System.out.println("Exiting " + this.name);
  }
}

class ThreadDemo
{
  public static void main(String args[])
  {
    MessageThread mt1 = new MessageThread
                ("Morning Thread", "Good Morning",1000);
    MessageThread mt2 = new MessageThread
                ("Hello Thread", "Hello",2000);
```

```
        MessageThread mt3 = new MessageThread
                    ("Welcome Thread", "Welcome",3000);

        mt1.start();
        mt2.start();
        mt3.start();
        try
        {
          mt1.join();
          mt2.join();
          mt3.join();
        }
        catch(InterruptedException e)
        {
          System.out.println("Main thread interrupted...");
        }
        System.out.println("Exiting main thread");
      }
    }
```

Output

```
Good Morning
Welcome
Hello
Good Morning
Hello
Good Morning
Welcome
Good Morning
Hello
Good Morning
Exiting Morning Thread
Welcome
Hello
Hello
Welcome
Exiting Hello Thread
Welcome
Exiting Welcome Thread
Exiting main thread
```

Exercise 8

Packages

Aim

Write a program to show an implementation of Packages.

Rectangle Class

```
package shape;
public class Rectangle
{
  private int length,breadth;

  public Rectangle(int l,int b)
  {
    this.length = l;
    this.breadth = b;
  }

  public void findArea()
  {
    System.out.println(this.length*this.breadth);
  }

  public void findPerimeter()
  {
    System.out.println(2*(this.length+this.breadth));
  }
}
```

Circle Class

```
package shape;
public class Circle
{
  private double radius;

  public Circle(double r)
  {
    this.radius = r;
  }

  public void findArea()
  {
    System.out.println(3.14*this.radius*this.radius);
  }
```

```
  public void findPerimeter()
  {
    System.out.println(2*3.14*this.radius);
  }
}
```

Geometry Class

```
import shape.*;
public class Geometry
{
  public static void main(String[] args)
  {
    Rectangle r =  new  Rectangle(10,20);
    r.findArea();
    r.findPerimeter();

    Circle c =  new Circle(10.0);
    c.findArea();
    c.findPerimeter();
  }
}
```

Output

```
200
60
314.0
62.800000000000004
```

Exercise 9

Abstract Classes

Aim

Write a java program to implement abstract classes.

Abstract Class

```
abstract class Shape
{
  String name;
  Shape(String name)
  {
    this.name = name;
  }
  void show()
  {
    System.out.println(this.name);
  }
  abstract int getArea();
}
class Rectangle extends Shape
{
  private int length,breadth;
  Rectangle(int l,int b, String name)
  {
    super(name);
    this.length = l;
    this.breadth = b;
  }
  int getArea()
  {
    return  this.length*this.breadth;
  }
}
class Inheritance
{
  public static void main(String[] args)
  {
    Rectangle r = new Rectangle(5,10,"R1");
    System.out.print("Rectangle ");
    r.show();
    System.out.print("Area = " + r.getArea());
  }
}
```

Output

```
Rectangle R1
Area = 50
```

Exercise 10

Calculator using Swing

Aim

Write a Java program using Java Swing to create a simple calculator. Arrange Buttons for digits and the + - * % operations properly. Add a text field to display the result. Handle any possible exceptions like divide by zero.

Calculator

```
btn0.addActionListener(new ActionListener() {
  public void actionPerformed(ActionEvent arg0) {
    if(lblDisplay.getText()=="0") {
      lblDisplay.setText("0");
    }
    else {
      lblDisplay.setText(lblDisplay.getText() + "0");
    }
  }
});
btn1.addActionListener(new ActionListener() {
  public void actionPerformed(ActionEvent arg0) {
    if(lblDisplay.getText()=="0") {
      lblDisplay.setText("1");
    }
    else {
      lblDisplay.setText(lblDisplay.getText() + "1");
    }
  }
});
btn2.addActionListener(new ActionListener() {
  public void actionPerformed(ActionEvent arg0) {
    if(lblDisplay.getText()=="0") {
      lblDisplay.setText("2");
    }
    else {
      lblDisplay.setText(lblDisplay.getText() + "2");
    }
  }
});
btn3.addActionListener(new ActionListener() {
  public void actionPerformed(ActionEvent arg0) {
    if(lblDisplay.getText()=="0") {
      lblDisplay.setText("3");
    }
    else {
      lblDisplay.setText(lblDisplay.getText() + "3");
    }
```

```
    }
  });
  btn4.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent arg0) {
      if(lblDisplay.getText()=="0") {
        lblDisplay.setText("4");
      }
      else {
        lblDisplay.setText(lblDisplay.getText() + "4");
      }
    }
  });
  btn5.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent arg0) {
      if(lblDisplay.getText()=="0") {
        lblDisplay.setText("5");
      }
      else {
        lblDisplay.setText(lblDisplay.getText() + "5");
      }
    }
  });
  btn6.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent arg0) {
      if(lblDisplay.getText()=="0") {
        lblDisplay.setText("6");
      }
      else {
        lblDisplay.setText(lblDisplay.getText() + "6");
      }
    }
  });
  btn7.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent arg0) {
      if(lblDisplay.getText()=="0") {
        lblDisplay.setText("7");
      }
      else {
        lblDisplay.setText(lblDisplay.getText() + "7");
      }
    }
  });
  btn8.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent arg0) {
      if(lblDisplay.getText()=="0") {
        lblDisplay.setText("8");
      }
      else {
        lblDisplay.setText(lblDisplay.getText() + "8");
      }
    }
```

```
});
btn9.addActionListener(new ActionListener() {
  public void actionPerformed(ActionEvent arg0) {
    if(lblDisplay.getText()=="0") {
      lblDisplay.setText("9");
    }
    else {
      lblDisplay.setText(lblDisplay.getText() + "9");
    }
  }
});

btnDecimal.addActionListener(new ActionListener() {
  public void actionPerformed(ActionEvent arg0) {
    if(lblDisplay.getText().indexOf('.') == -1){
      lblDisplay.setText(lblDisplay.getText() + ".");
    }
  }
});

btnAdd.addActionListener(new ActionListener() {
  public void actionPerformed(ActionEvent arg0) {
    fno = Double.parseDouble(lblDisplay.getText());
    lblDisplay.setText("0");
    operator = 1;
  }
});

btnSub.addActionListener(new ActionListener() {
  public void actionPerformed(ActionEvent arg0) {
    fno = Double.parseDouble(lblDisplay.getText());
    lblDisplay.setText("0");
    operator = 2;
  }
});

btnMultiply.addActionListener(new ActionListener() {
  public void actionPerformed(ActionEvent arg0) {
    fno = Double.parseDouble(lblDisplay.getText());
    lblDisplay.setText("0");
    operator = 3;
  }
});

btnDivide.addActionListener(new ActionListener() {
  public void actionPerformed(ActionEvent arg0) {
    fno = Double.parseDouble(lblDisplay.getText());
    lblDisplay.setText("0");
    operator = 4;
  }
});
```

```
btnClear.addActionListener(new ActionListener() {
  public void actionPerformed(ActionEvent arg0) {
    lblDisplay.setText("0");
  }
});

btnSign.addActionListener(new ActionListener() {
  public void actionPerformed(ActionEvent arg0) {
    if(lblDisplay.getText().indexOf('-') == -1) {
      lblDisplay.setText("-" + lblDisplay.getText());
    }
    else
    {
      lblDisplay.setText(lblDisplay.getText().substring(1));
    }
  }
});

btnEquals.addActionListener(new ActionListener() {
  public void actionPerformed(ActionEvent arg0) {
    sno = Double.parseDouble(lblDisplay.getText());
    double result=0;

    switch(operator)
    {
      case 1:
      result = fno + sno;
      break;
      case 2:
      result = fno - sno;
      break;
      case 3:
      result = fno * sno;
      break;
      case 4:
      result = fno / sno;
      break;
    }
    lblDisplay.setText(String.valueOf(result));
  }
});
```

Exercise 11

Database Connectivity using JDBC

Aim

Write a Java program to display all records from a table using Java Database Connectivity(JDBC).

JDBC

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class StudentDatabase
{
  public static void main(String[] args) throws SQLException
  {
    try
    {
      Connection con=DriverManager.getConnection("jdbc:mysql://localhost/
          student","student","student");
      String sql = "select * from student";
      Statement statement = con.createStatement();
      ResultSet result = statement.executeQuery(sql);
      System.out.println("Reg.No.\tName \tPgm \t Marks");
      while(result.next())
      {
        int rollno = result.getInt("regno");
        String  name = result.getString("name");
        String  programme = result.getString("programme");
        int marks = result.getInt("marks");
        System.out.print(rollno + "\t");
        System.out.print(name + "\t");
        System.out.print(programme+ "\t");
        System.out.print(marks+ "\t");
        System.out.println();
      }
    }
    catch(SQLException e)
    {
      System.out.println(e);
    }
  }
}
```

Output

```
Reg.No. Name  Pgm    Marks
1       abc   cs      900
2       def   math    890
3       pqr   cs      990
```

Exercise 12

Generics

Aim

Write a program that demonstrates generics.

Generic Stack

```
import java.util.*;
class Stack<T>
{
  private ArrayList<T> s;
  private int top;

  Stack()
  {
    s = new ArrayList<T>();
    top = -1;
  }

  public void push(T item)
  {
    s.add(item);
    top++;
  }

  public T pop()
  {
    if(top == -1)
    {
      System.out.println("Stack Empty...");
      return null;
    }
    top--;
    return s.get(top);
  }
  public void display()
  {
    System.out.println("Current Stack");
    for(int i = top;i>=0;i--)
    {
      System.out.println(s.get(i));
    }
  }
}
```

```
class StackOperation
{
  public static void main(String[] args)
  {
    Stack<Integer> s1 = new Stack<Integer>();
    s1.push(11);
    s1.push(22);
    s1.push(33);
    s1.display();
    s1.pop();
    s1.display();

    Stack<Double> s2 = new Stack<Double>();
    s2.push(11.5);
    s2.push(22.5);
    s2.push(33.3);
    s2.display();
    s2.pop();
    s2.display();
  }
}
```

Output

```
Current Stack
33
22
11
Current Stack
22
11
Current Stack
33.3
22.5
11.5
Current Stack
22.5
11.5
```