RAMAIAH Institute of Technology

Department of Computer Science & Engineering

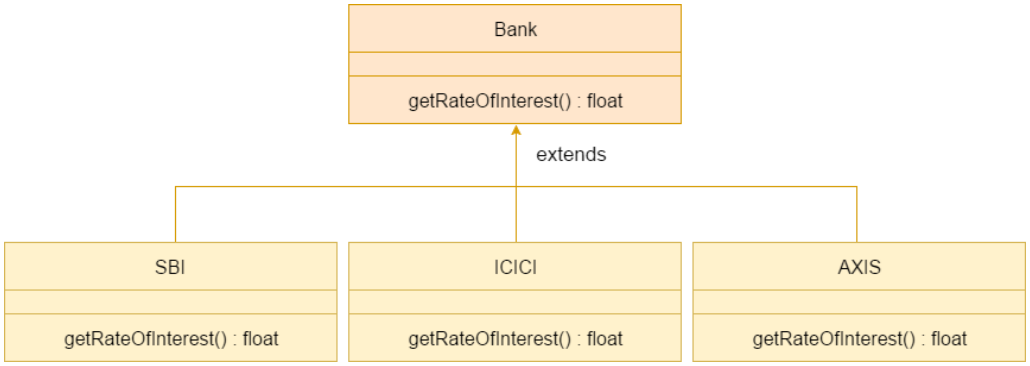
LAB MANUAL FOR III SEM (CSE) (Autonomous Syllabus)

Subject Code: CSL37 **TERM: Oct 2022-Jan 2023** **I.A.Marks : 50**
Subject Name: Object oriented Programming Laboratory **Exam Hours: 03**
Credits: 0:0:1 **Exam Marks: 50**

Design, develop, and implement the following programs in Java

PART - A

1. **Write a java program that demonstrate Method overriding:**
   a. **Create a class Bank that provides method to get the rate of interest called "getRateOfinterest() which is of type float.**
   b. **The rate of interest varies according to banks. SBI, ICICI and AXIS banks could provide 8%, 7%, and 9% rate of interest.**
   c. **Create Inheritance mechanism where SBI, ICICI and AXIS bank child classes extends Bank parent class that contains getRateOfinterest() method.**
   d. **Override getRateOfinterest() method in the child classes with updated interests as 8%, 7%, and 9%.**
   e. **Finally create a driver class BankDetails that can display the rate of interest of all the banks.**




```
//Java Program to demonstrate the real scenario of Java Method Overriding
//where three classes are overriding the method of a parent class.
//Creating a parent class.
class Bank{
int getRateOfInterest(){return 0;}
}
//Creating child classes.
class SBI extends Bank{
int getRateOfInterest(){return 8;}
}
```

```
class ICICI extends Bank{
int getRateOfInterest(){return 7;}
}
class AXIS extends Bank{
int getRateOfInterest(){return 9;}
}
//Test class to create objects and call the methods
class BankDetails{
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());
}
}
```

2. **Write a Java Program that does the following related to Inheritance:**

a. **Create an abstract class called 'Vehicle' which contains the 'hashelmet','year of manufacture' and two abstract methods 'getData()' and 'putData()'. Demonstrate the error when attempt is made to create objects of 'Vehicle'.**

b. **Have two derived classes 'TwoWheeler' and 'FourWheeler'. 'FourWheeler' is a final class. Demonstrate the error when attempt is made to inherit from 'FourWheeler'.**

c. **Your abstract class should haveoverloadedconstructors that initializes 'hashelmet' and 'year of manufacture for TwoWheeler and FourWheeler respectively.**

d. **'TwoWheeler' has data elements 'Brand', 'Cost', 'EngineType' (possible values "2 stroke", "4 stroke"), and 'Color' which are private, protected, 'friendly/default' and public respectively. Demonstrate the various ways in which the two abstract methods can be dealt 'getData()' and 'putData()' can be dealt with by the derived classes, 'TwoWheeler' and 'FourWheeler'.**

Solution:

```
import java.util.*;

abstract class Vehicle
{
   boolean hashelmet;
   int yom;
   abstract void getData();
   abstract void putData();
```

```
    Vehicle(boolean h,int n)
    {
       hashelmet=h;
       yom=n;
    }
}

class TwoWheeler extends Vehicle
{
    private String Brand;
    protected int Cost;
    String EngineType;
    public String Color;

    TwoWheeler(int n)
    {
       super(true,n);
    }

    void getData()
    {
       Scanner sc=new Scanner(System.in);
       System.out.println("Enter Two Wheeler details: Its Brand name,Cost,EngineType
and Colour");
       Brand=sc.next();
       Cost=sc.nextInt();
       EngineType=sc.next();
       Color=sc.next();
    }

    void putData()
    {
       System.out.println("\n");
          System.out.println("The details of Two Wheeler");
          System.out.println("Brand:"+Brand+"\nCost:"+Cost+"\nEngineType:"+EngineTy
pe+"\nColor:"+Color+"\nYear of Manufacture:"+yom+"\nHas helmet:"+hashelmet);
    }
}

final class FourWheeler extends Vehicle
{
    FourWheeler(int n)
    {
       super(false,n);
```

```
    }

    void getData()
    {

    }

    void putData()
    {
        System.out.println("\n");
            System.out.println("The details of Four Wheeler");
        System.out.println("Year of Manufacture:"+yom+"\nHas helmet:"+hashelmet);
    }
}

/*class A extends FourWheeler
{
    A()
    {
        super(5);
    }
}
*/

public class Program1
{
        public static void main(String[] args)
        {
                //Vehicle v=new Vehicle();
                //Cannot Create instance of an abstract class

                TwoWheeler t1=new TwoWheeler(1995);
      FourWheeler f1=new FourWheeler(2006);

                t1.getData();
      t1.putData();
           f1.putData();
        }
}
```

3.	**Write a Java Program that does the following:** **a. Create an abstract class called 'Shape' which contains Two instance variables color (String) and filled (boolean).** • **Two constructors: a no-arg (no-argument) constructor that initializes**

the color to "green" and filled to true, and a constructor that initializes the color and filled to the given values.

- **Getter and setter for all the instance variables. By convention, the getter for a boolean variable xxx is called isXXX() (instead of getXxx() for all the other types).**
- **A toString() method that returns "A Shape with color of xxx and filled/Not filled".**
- **An abstract method getArea()**
- **Demonstrate the error when attempt is made to create objects of 'Shape'.**

b. Write two subclasses of Shape called Circle and Rectangle. Rectangle is a final class. Demonstrate the error when attempt is made to inherit from 'Rectangle'.

c. Write a class called Square, as a subclass of Rectangle. Convince yourself that Square can be modeled as a subclass of Rectangle. Square has no instance variable, but inherits the instance variables width and length from its superclass Rectangle.

Solution:

```
abstract class Shape
{
   String color;
   boolean filled;
   abstract double getArea();

   Shape()
   {
      color="green";
      filled=true;
   }

   Shape(String c,boolean f)
   {
      color=c;
      filled=f;
   }

   boolean isFILLED()
   {
      return filled;
   }

   String getColor()
   {
      return color;
   }

   void setFILLED(boolean b)
   {
      this.filled=b;
   }

   void setColor(String c)
   {
      this.color=c;
   }
```

```
    public String toString()
    {
       if(this.filled==false)
          return "A Shape with color " +this.color+" and not filled";
       else
          return "A Shape with color " +this.color+" and filled";
    }
}

class Circle extends Shape
{
    int r;

    Circle(int r1)
    {   super();
       r=r1;
    }

    Circle(String c,boolean f,int r1)
    {   super(c,f);
       r=r1;
    }

    double getArea()
    {
       return 3.14*r*r;
    }

    void display()
    {
       System.out.println("\nDisplaying the details of Circle:");
       System.out.println(isFILLED());
       System.out.println(getColor());
       System.out.println("\n");
    }

    void change(String c,boolean b)
    {
       setColor(c);
       setFILLED(b);
    }
}

final class Rectangle extends Shape
{
    int a,b;

    Rectangle(int a1,int b1)
    {   super();
       a=a1;
       b=b1;
    }

    Rectangle(String c,boolean f,int a1,int b1)
    {   super(c,f);
```

```
        a=a1;
        b=b1;
    }

    double getArea()
    {
        return a*b;
    }
}

/*class Square extends Rectangle
{   Square()
    {
        super(5,6);
    }
    void display()
    {
        System.out.println(a + " " + b);
    }
}
Cannot inherit a final class
*/

public class Program2
{
        public static void main(String[] args)
        {
                /*Shape s=new Shape();
                Cannot create instance of an abstact class*/

        Circle c=new Circle("blue",false,5);
                Rectangle r=new Rectangle("red",true,2,4);

        System.out.println(c);
                System.out.println(r);

        System.out.println("\n");
        System.out.println("Area of Circle:"+c.getArea());
                System.out.println("Area of Rectangle:"+r.getArea());

        c.display();

        c.change("brown",true);

        c.display();
        }
}
```

4. **Write a Java Program that does the following**

a. Create a superclass, Student, and two subclasses, Undergrad and Grad.

b. The superclass Student should have the following data members: name, ID, grade, age

c. The superclass, Student should have at least one method: Boolean isPassed (double grade)

d. The purpose of the isPassed method is to take one parameter, grade (value between 0 and 100) and check whether the grade has passed the requirement for passing a course.

In the Student class this method should be empty as an abstract method.

e. **The two subclasses, Grad and Undergrad, will inherit all data members of the Student class and override the method isPassed. For the UnderGrad class, if the grade is above 70.0, then isPassedreturns true, otherwise it returns false. For the Grad class, if the grade is above 80.0, then isPassed returns true, otherwise returns false.**

f. **Demonstrate "final" keyword in the above class.**

g. **Create a test class for your three classes. In the test class, create one Grad object and one Undergrad object. For each object, provide a grade and display the results of the isPassed method.**

Solution:

```
import java.util.*;
abstract class Student
{
   private String Name;
   protected int ID;
   double grade;
   public int age;
   abstract boolean isPassed(double Grade);

   final void setter(String name)
   {
      Name=name;
   }

   String getter()
   {
      return Name;
   }
}

class Undergrad extends Student
{
   void getData()
   {
      Scanner sc=new Scanner(System.in);
      System.out.println("\nEnter Name,ID,age");
      setter(sc.next());
      ID=sc.nextInt();
      age=sc.nextInt();
   }

   boolean isPassed(double Grade)
   {
      grade=Grade;
      if(grade<=70)
```

```
            return false;
        else
            return true;
    }

    void display()
    {
        System.out.println("Name:"+getter()+"\nAge:"+age+"\nID:"+ID);
    }

    /*void setter(String name)
    {
    }
    Cannot override the final setter method*/
}

class Grad extends Student
{
    void getData()
    {
        Scanner sc=new Scanner(System.in);
        System.out.println("\nEnter Name,ID,age");
        setter(sc.next());
        ID=sc.nextInt();
        age=sc.nextInt();
    }

    boolean isPassed(double Grade)
    {
        grade=Grade;
        if(grade<=80)
            return false;
        else
            return true;
    }

    void display()
    {
        System.out.println("Name:"+getter()+"\nAge:"+age+"\nID:"+ID);
    }
}

public class Program3
{
```

```
        public static void main(String[] args)
        {
                Undergrad u=new Undergrad();
                u.getData();
        if(u.isPassed(65))
                System.out.println("Student has passed\n");
                else
                System.out.println("Student has failed\n");

        u.display();

        Grad g=new Grad();
        g.getData();

                if(g.isPassed(90))
                System.out.println("Student has passed\n");
                else
                System.out.println("Student has failed\n");
                g.display();
        }
}
```

5. **Write a Java Program that does the following**

a. Create a super class called Car. The Car class has the following fields and methods.
- **int speed; double regularPrice; String color; double getSalePrice();**

b. Create a sub class of Car class and name it as Truck. The Truck class has the following fields and methods.
- **int weight; double getSalePrice();**
- **//Ifweight>2000,10% discount. Otherwise,20%discount.**

c. Create a subclass of Car class and name it as Ford. The Ford class has the following fields and methods
- **int year; intmanufacturerDiscount; double getSalePrice();**
- **//From the sale price computed from Carclass, subtract the manufacturer Discount.**

d. Create a subclass of Car class and name it as Sedan. The Sedan class has the following fields and methods.
- **int length; double getSalePrice();**
- **//If length>20feet, 5% discount, Otherwise, 10% discount.**

e. Create MyOwnAutoShop class which contains the main() method. Perform the following within the main() method.
- **Create an instance of Sedan class and initialize all the fields with appropriate values.**
- **Use super(...) method in the constructor for initializing the fields of the superclass.**
- **Create an instance of the Ford class and initialize all the fields with appropriate values**
- **Use super(...) method in the constructor for initializing the fields of the super class.**
- **Create an instance of Car class and initialize all the fields with appropriate**

values. Display the sale prices of all instances.

Solution:

```
class Car
{
        int speed;
        double regularPrice;
        String color;
        Car(int s,double price,String c)
        {
                speed=s;
        regularPrice=price;
        color=c;
        }

        double getSalePrice()
        {
        return regularPrice;
        }
}

class Truck extends Car
{
        int weight;
        Truck(int s,double price,String c,int w)
        {
        super(s,price,c);
        weight=w;
        }

        double getSalePrice()
        {
        if(weight>2000)
        {
                        regularPrice=regularPrice*0.9;
                        return regularPrice;
        }
        else
        {
                        regularPrice=regularPrice*0.8;
                        return regularPrice;
        }
        }
}
```

```
class Ford extends Car
{
	int manufacturerDiscount,year;

	Ford(int s,double price,String c,int m)
	{
	super(s,price,c);
	manufacturerDiscount=m;
	}

	double getSalePrice()
	{
	regularPrice-=manufacturerDiscount;
	return regularPrice;
	}
}

class Sedan extends Car
{
	int length;

	Sedan(int s,double price,String c,int l)
	{
	super(s,price,c);
	length=l;
	}

	double getSalePrice()
	{
		if(length>20)
	{
			regularPrice=regularPrice*0.95;
			return regularPrice;
	}
		else
	{
			regularPrice=regularPrice*0.9;
			return regularPrice;
	}
	}
}

public class Program4
```

```
{
        public static void main(String[] args)
        {
                Truck t=new Truck(65,2500000,"Red",3000);
        System.out.println("Price of truck is "+t.getSalePrice());
        Car c = new Car(100,800000,"Black");
        System.out.println("Price of Car is "+c.getSalePrice());
        Ford f=new Ford(120,2200000,"Yellow",120000);
        System.out.println("Price of ford is "+f.getSalePrice());
        Sedan s= new Sedan(100,3500000,"Blue",22);
        System.out.println("Price of Sedan is "+s.getSalePrice());
        }
}
```

6. **Write a Java Program that implements the following**

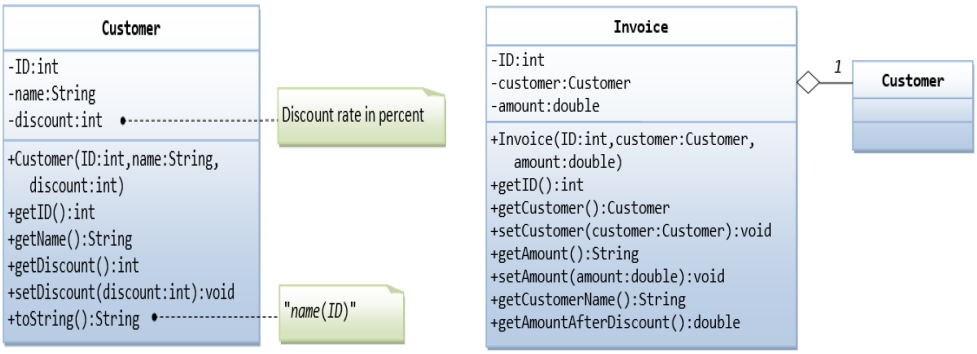
- **Define a class SavingsAccount with following characteristics.**
- **Use a static variable annualInterestRate to store the annual interest rate for all account holders.**
- **Private data member savingsBalance indicating the amount the saver currently has on deposit.**
- **Method calculateMonthlyInterest to calculate the monthly interest as (savingsBalance * annualInterestRate / 12). After calculation, the interest should be added to savingsBalance.**
- **Static method modifyInterestRate to set annualInterestRate.**
- **Parameterized constructor with savingsBalance as an argument to set the value of that instance.**
- **Test the class SavingsAccount to instantiate two savingsAccount objects, saver1 and saver2, with balances of Rs.2000.00 and Rs3000.00, respectively. Set annualInterestRate to 4%, then calculate the monthly interest and print the new balances for both savers. Then set the annualInterestRate to 5%, calculate the next month's interest and print the new balances for both savers.**

Solution:

```
class SavingsAccount
{
        static int annualInterestRate;
        private double savingsBalance;

        SavingsAccount(double s)
        {
        savingsBalance=s;
        }

        static void modifyInterestRate(int x)
        {
        annualInterestRate=x;
        }
```

```
        void calculateMonthlyInterest()
        {
        double d=(savingsBalance*annualInterestRate)/12;
        savingsBalance=savingsBalance+d;
        }

        void display()
        {
        System.out.println(savingsBalance);
        }
}

public class Program5
{
        public static void main(String[] args)
        {
                SavingsAccount saver1=new SavingsAccount(2000);
                SavingsAccount saver2=new SavingsAccount(3000);

                SavingsAccount.modifyInterestRate(4);
                saver1.calculateMonthlyInterest();
                saver2.calculateMonthlyInterest();

                saver1.display();
                saver2.display();

                SavingsAccount.modifyInterestRate(5);
                saver1.calculateMonthlyInterest();
                saver2.calculateMonthlyInterest();

                saver1.display();
                saver2.display();
        }
}
```

7. **Write a Java Program that does the following**
   - **The Customer class models a customer is design as shown in the class diagram. Write the codes for the Customer class and a test driver to test all the public methods.**



- **The Invoice class, design as shown in the class diagram, composes a Customer instance (written earlier) as its member. Write the codes for the Invoice class and a test driver to test all the public methods.**

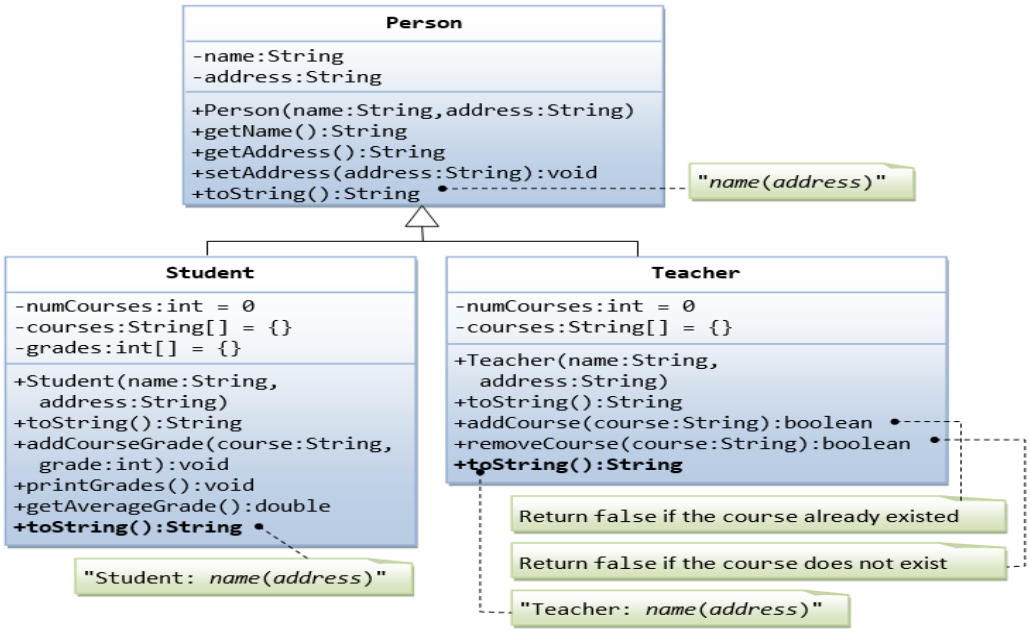
Solution:

```
class Customer
{
   private int ID;
   private String Name;
   private int discount;

   Customer(int ID,String Name,int discount)
   {
      this.Name=Name;
      this.ID=ID;
      this.discount=discount;
   }
   int getID()
   {
      return ID;
   }

   String getName()
   {
      return Name;
   }

   int getDiscount()
   {
      return discount;
   }

   void setDiscount(int discount)
   {
      this.discount=discount;
   }
```

```
    public String toString()
    {
        return Name+"("+ID+")";
    }
}
class Invoice
{
    private int ID;
    private Customer customer;
    private double amount;

    Invoice(int ID,Customer customer,double amount)
    {
        this.ID=ID;
        this.customer=customer;
        this.amount=amount;
    }

    int getID()
    {
        return ID;
    }

    Customer getCustomer()
    {
        return customer;
    }

    void setCustomer(Customer customer)
    {
        this.customer=customer;
    }

    String getAmount()
    {
        return Double.toString(amount);
    }

    void setAmount(double amount)
    {
        this.amount=amount;
    }
```

```
   String getCustomerName()
   {
      return customer.getName();
   }

   double getAmountAfterDiscount()
   {
      return (amount*customer.getDiscount())/100;
   }
}

public class Program6
{
        public static void main(String[] args)
   {
            Customer c=new Customer(25,"Raju",5);
            System.out.println("The details of first customer:");
            System.out.println(c.getID());
            System.out.println(c.getDiscount());
            System.out.println(c.getName());
            c.setDiscount(7);
      System.out.println("After discount:");
            System.out.println(c.getDiscount());
            System.out.println(c);

            Customer c1=new Customer(26,"Ram",9);
            Invoice i=new Invoice(28,c1,60000);
      System.out.println("\nInvoice details:");
            System.out.println(i.getID());
            System.out.println(i.getCustomer());
            System.out.println(i.getAmount());
            i.setAmount(70000);
            System.out.println(i.getAmount());
            System.out.println(i.getCustomerName());
            System.out.println(i.getAmountAfterDiscount());
         }
}
```

8. **We are required to model students and teachers in our application. We can define a superclass called `Person` to store common properties such as `name` and `address`, and subclasses `Student` and `Teacher` for their specific properties. For students, we need to maintain the courses taken and their respective grades; add a course with grade, print all courses taken and the average grade. Assume that a student takes no more than 30 courses for the entire program. For teachers, we need to maintain the courses taught currently, and able to add or remove a course taught. Assume that a teacher teaches not more than 5 courses concurrently.**



Solution:

```
class Person
{
  private String name;
  private String address;
    Person (String name, String address)
  {
    this.name = name;
    this.address = address;
  }
  String getName ()
  {
    return name;
  }
  String getAddress ()
  {
    return address;
  }
  void setAddress (String address)
  {
    this.address = address;
  }
  public String toString ()
  {
    return name + "(" + address + ")";
  }
}
```

```
class Student1 extends Person
{
 int numCourses = 0;
 String courses[] = new String[30];
 int grades[] = new int[30];
  Student1 (String name, String address)
 {
  super (name, address);
 }
 void addCourseGrade (String course, int grade)
 {
  if (numCourses <= 29)
    {
        courses[numCourses] = course;
        grades[numCourses] = grade;
        numCourses++;
    }
  else
    {
        System.out.println ("Maximum number of courses");
    }
 }
 void printGrades ()
 {
  for (int i = 0; i < numCourses; i++)
    System.out.println ("Course:" + courses[i] + " Grade:" + grades[i]);
 }
 double getAverageGrades ()
 {
  double d = 0;
  for (int i = 0; i < numCourses; i++)
    d = d + grades[i];
  d = d / numCourses;
  return d;
 }
 public String toString ()
 {
  return getName () + "(" + getAddress () + ")";
 }
}

class Teacher extends Person
{
 int numCourses = 0;
```

```
String courses[] = new String[5];
  Teacher(String name, String address)
{
  super (name, address);
}
boolean addCourse (String course)
{
  if (numCourses <= 4)
    {
        for (int i = 0; i < numCourses; i++)
          if (courses[i].equals (course))
            return false;
        courses[numCourses] = course;
        numCourses++;
        return true;
    }
  else
    return false;
}
boolean removeCourse (String course)
{
  if (numCourses != 0)
     {
            for (int i = 0; i < numCourses; i++)
               if (courses[i].equals (course))
                {
                  courses[i] = " ";
                  return true;
                }
            return false;
     }
  return false;
}
public String toString ()
{
            return getName () + "(" + getAddress () + ")";}
}
public class Program7
{
          public static void main (String[]args)
              {
                Student1 s = new Student1("Raju", "Bangalore");
                System.out.println (s);
                s.addCourseGrade ("Maths", 85);
```

```
            s.addCourseGrade ("OOPS", 80);
        s.addCourseGrade ("DS",75);
        s.addCourseGrade ("DMS", 70);
        s.printGrades();
        System.out.println(s.getAverageGrades());
        Teacher t=new Teacher("CC","DD");
        if(t.addCourse("Maths"))
          System.out.println("Course added");
        else
          System.out.println("Max limit reached/course already exists");
        if(t.addCourse("Maths"))
          System.out.println("Course added");
        else
          System.out.println("Max limit reached/course already exists");
        if(t.addCourse("OOPS"))
          System.out.println("Course added");
        else
          System.out.println("Max limit reached/course already exists");
        if(t.addCourse("DS"))
          System.out.println("Course added");
        else
          System.out.println("Max limit reached/course already exists");
        if(t.removeCourse("Maths"))
          System.out.println("Course removed");
        else
          System.out.println("Zero courses/course does not exist");
            if(t.removeCourse("TOC"))
          System.out.println("Course removed");
        else
          System.out.println("Zero courses/course does not exist");
           }
}
```

PART - B

1. **Write a JAVA program which does the following operations:**

   a. **An Interface class for Stack Operations**
   b. **A Class that implements the Stack Interface and creates a fixed length Stack.**
   c. **A Class that implements the Stack Interface and creates a Dynamic Length Stack.**
   d. **A Class that uses both the above Stacks through Interface reference and does the Stack operations that demonstrates the runtime binding.**

```
    interface stackop
{
    void push(int item);
    int pop();
}
class FixedStack implements stackop
{
    private int stk[ ];
    private int tos;
    FixedStack(int size)
    {
        stk=new int[size];
        tos=-1;
    }
    public void push(int item)
    {
        if(tos==stk.length-1)
        {
            System.out.println("Stack Overflows");
            int t[ ]=new int[stk.length * 2];
            for(int i=0;i<stk.length;i++)
                t[i]=stk[i];
            stk=t;
            stk[++tos]=item;
        }
        else
            stk[++tos]=item;
    }
    public int pop()
    {
        if(tos<0)
        {
            System.out.println("Stack Underflows");
            return 0;
```

```
            }
            else
                return stk[tos--];
        }
}
class DynStack implements stackop
{
        private int stk[ ];
        private int tos;
        DynStack(int size)
        {
            stk=new int[size];
            tos=-1;
        }
        public void push(int item)
        {
            if(tos==stk.length-1)
            {
                System.out.println("Stack Overflows.");
                int t[ ]=new int[stk.length * 2];
                for(int i=0;i<stk.length;i++)
                    t[i]=stk[i];
                stk=t;
                stk[++tos]=item;
            }
            else
                stk[++tos]=item;
        }
        public int pop()
        {
            if(tos<0)
            {
                System.out.println("Stack Underflows.");
                return 0;
```

```
            }
            else
                return stk[tos--];
        }
}
class StackTest
{
    public static void main(String args[ ])
    {
        FixedStack fs=new FixedStack(3);
        DynStack ds=new DynStack(5);
        stackop mystk;
        for(int i=0;i<3;i++)
            fs.push(i);
        System.out.println("Fixed length Stack Contents are.");
        for(int i=0;i<3;i++)
            System.out.println(fs.pop());
        for(int i=0;i<7;i++)
            ds.push(i);
        System.out.println("Dynamic length Stack Contents are");
        for(int i=0;i<7;i++)
            System.out.println(ds.pop());
        mystk=fs;
        for(int i=0;i<3;i++)
            mystk.push(i);
        System.out.println("Fixed length Stack Contents using interface reference");
        for(int i=0;i<3;i++)
            System.out.println(mystk.pop());
        mystk=ds;
        for(int i=0;i<7;i++)
            mystk.push(i);
        System.out.println("Dynamic length Stack Contents using interface reference");
        for(int i=0;i<7;i++)
            System.out.println(mystk.pop());
```

	} }

2.	**Write a Java Program to implement Packages by performing following operations:**
	a. Create a class TwoDim which contains private members as x and y coordinates in package P1. Define the default constructor, a parameterized constructor and override toString() method to display the co-ordinates.
	b. Reuse the class TwoDim and in package P2 create another class ThreeDim, adding a new dimension as z as its private member. Define the constructors for the subclass and override toString() method in the subclass also.
	c. Write a driver code that imports both packages and usage of classes TwoDim and ThreeDim by creating objects.
	
	Create a sub directory named p1 under current working directory and create a java program containing class TwoDim and save as TwoDim.java
	package p1;
	public class TwoDim
	{
	private int x;
	private int y;
	public TwoDim()
	{
	this.x = 0;
	this.y = 0;
	}
	public TwoDim(int x, int y)
	{
	this.x = x;
	this.y = y;
	}
	@Override
	public String toString()
	{
	return "Coordinate: x = " + x + " y = " + y;
	}
	}
	
	Create a sub directory named p2 under current working directory and create a java program containing class ThreeDim and save as ThreeDim.java
	package p2;

```
import p1.*;
public class ThreeDim extends TwoDim
{
        private int z;
         public ThreeDim()
        {
                super(0, 0);
                this.z = 0;
        }
        public ThreeDim(int x, int y, int z)
        {
        super(x, y);
        this.z = z;
        }
        @Override
        public String toString()
        {
                return super.toString() + " z = " + z;
        }
}
```

In the current working directory create a file named as PkgDemo.java

```
import p1.*;
import p2.*;
public class PkgDemo
 {
        public static void main(String[] args)
        {
                 TwoDim td = new TwoDim(1, 2);
                System.out.println(td);
                ThreeDim thd = new ThreeDim(3, 4, 5);
                 System.out.println(thd);
        }
}
```

To Execute follow the below steps:

(i)Compile the file of package p1 and p2 using the below command in current working directory to create class files of TwoDim.java and ThreeDim.java

```
javac -classpath . p1/TwoDim.java
```

javac -classpath . p2/ThreeDim.java

(ii) Compile the source file PkgDemo.java and run the same:

javac PkgDemo.java

java PkgDemo

Coordinate: x = 1 y = 2

Coordinate: x = 3 y = 4 z = 5

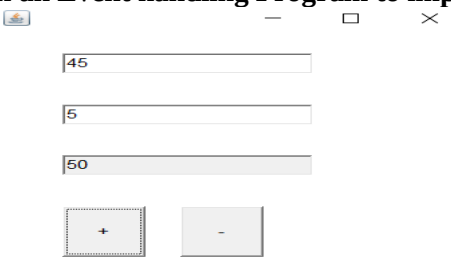
3. **Write a program to create two threads t1, t2 which should prints odd numbers, and reverse of a number respectively and stops thread after creating 3 odd numbers.**

Solution:

```
class A extends Thread
{       public void run()
        {       boolean flag=false;
                int c=0,i=0;
                while(c<3)
                {
                        if(i%2!=0)
                        {       System.out.println(i);
                                c++;
                        }
                        i++;
                        if(c==3)
                        {       flag=true;
                                break;
                        }
                }
                if(flag)
                {
                        try {
                                wait();
                        }
                        catch(Exception e)
                        {}
                }
        }
}
class B extends Thread
{       int n;
        B(int n)
        {
                this.n=n;
        }
```

```
        public void run()
        {       int reversed = 0;
        while(n != 0) {
                int digit = n % 10;
                reversed = reversed * 10 + digit;
                n /= 10;
        }
        System.out.println("Reversed Number: " + reversed);
        }
}
public class Program1 {
        public static void main(String[] args) {
                A a=new A();
                a.start();
                B b=new B(1234);
                b.start();
        }
}
```

4. **Design an Event handling Program to implement the following**



Solution:

```
// importing necessary libraries
import java.awt.*;
import java.awt.event.*;
// Our class extends Frame class and implements ActionListener interface
public class Program2 extends Frame implements ActionListener {
   // creating instances of TextField and Button class
   TextField tf1, tf2, tf3;
   Button b1, b2;
   // instantiating using constructor
   Program2() {
      // instantiating objects of text field and button
      // setting position of components in frame
      tf1 = new TextField();
      tf1.setBounds(50, 50, 150, 20);
      tf2 = new TextField();
      tf2.setBounds(50, 100, 150, 20);
```

```
        tf3 = new TextField();
        tf3.setBounds(50, 150, 150, 20);
        tf3.setEditable(false);
        b1 = new Button("+");
        b1.setBounds(50, 200, 50, 50);
        b2 = new Button("-");
        b2.setBounds(120,200,50,50);
        // adding action listener
        b1.addActionListener(this);
        b2.addActionListener(this);
        // adding components to frame
        add(tf1);
        add(tf2);
        add(tf3);
        add(b1);
        add(b2);
        // setting size, layout and visibility of frame
        setSize(300,300);
        setLayout(null);
        setVisible(true);
    }
    // defining the actionPerformed method to generate an event on buttons
    public void actionPerformed(ActionEvent e) {
        String s1 = tf1.getText();
        String s2 = tf2.getText();
        int a = Integer.parseInt(s1);
        int b = Integer.parseInt(s2);
        int c = 0;
        if (e.getSource() == b1){
            c = a + b;
        }
        else if (e.getSource() == b2){
            c = a - b;
        }
        String result = String.valueOf(c);
        tf3.setText(result);
    }
// main method
public static void main(String[] args) {
    new Program2();
}
}
```

5. **Write a Java program to find area of a triangle with three sides a, b, c. A triangle can be formed only if a+b>c, b+c>a, c+a>b. First verify whether the above three conditions are satisfied. If any one of them is not satisfied then throw an exception called ValidateTriangle Exception**

Enter the 3 sides of triangle:
7 4 10
Valid Triangle
Enter the 3 sides of triangle:
2 6 8
Not a valid triangle

Solution:

```
import java.util.*;
class Triangle extends Exception
{
	public String toString()
	{
		return "Not an acceptable triangle";
	}
}
public class Program4
{
	public static void main(String args[])throws Triangle
	{	Scanner sc=new Scanner(System.in);
		int a,b,c;
		System.out.println("Enter the 3 sides of a triangle:");
		a=sc.nextInt();
		b=sc.nextInt();
		c=sc.nextInt();
		try
		{
			if((a<b+c)&&(b<a+c)&&(c<a+b))
				System.out.println("Valid Triangle");
			else
				throw new Triangle();
		}
		catch(Triangle e)
		{
			System.out.println(e);
		}
	}
}
```

6. **Write a Java program to display multiplication table of 8 & 9 using shared resources "synchronized displayTable(intnum)". The table should be displayed with 1 sec delay**

between every number. First print multiplication table of 8 and then 9.

Solution:

```
class A
{
	synchronized void displayTable(int n)
	{	try
		{
			for(int i=1;i<=10;i++)
			{
				System.out.println(i*n);
				Thread.sleep(1000);
			}
		}
		catch(Exception e)
		{

		}
	}
}
class Mul extends Thread
{
	A a;int n;
	Mul(A a,int n)
	{	this.n=n;
		this.a=a;
	}
	public void run()
	{
		a.displayTable(n);
	}
}
public class Program5
{
	public static void main(String args[])
	{
		A a=new A();
		Mul m=new Mul(a,8);
		Mul m1=new Mul(a,9);
		m.start();
		m1.start();
	}
}
```

7. **Write a Java program to implement "ADDTION" and "MULTIPLICATION" of two**

numbers using Lambda Expressions

Solution:

```
interface Calc{
	public int res(int x,int y);
}

public class Program6{
	public static void main(String args[])
	{

		Calc a=(x,y)->(x+y);
			System.out.println("\nSUM IS:"+a.res(2,3));


		Calc  m=(x,y)->(x*y);
			System.out.println("\nPRODUCT IS:"+m.res(3,4));
	}

}
```

8. **Write a java program to accept a string. Convert the string to uppercase. Count and output the number of double letter sequences that exist in the string.**
Sample Input: "SHE WAS FEEDING THE LITTLE RABBIT WITH AN APPLE"
Sample Output: 4

Solution:

```
import java.util.*;
public class Program7
{
	public static void main(String args[])
	{
		Scanner sc=new Scanner(System.in);
		String s;
		System.out.println("Enter string");
		s=sc.nextLine();
		int c=0;
		char ch=s.charAt(0);
		for(int i=1;i<s.length();i++)
		{
			if(s.charAt(i)==ch)
				c++;
			ch=s.charAt(i);
		}
		System.out.println(c);
	}
}
```