1901301071 15 - Pusum.

CEB2

OOPS - Assignment

Q-1) Write a program to demonstrate the use of array of objects and static variables.

```
class Students {
    String sname;
    int rollno;
    Static string HOD;

    Static { HOD = "Prof. Mentu"; }

    Public Students (String sname, int rollno)
    {
        this.sname = sname;
        this.rollno = rollno;

        public string to string()
        {
            return "Student name:" + sname +
                "| Roll number" + roll no + "| HOD"
                + HOD;
        }
}

Public class Array of objects {

    public Static void main (String[] args) {
        Students S1 = new students ("Pusum", 1);
        students S2 = new students ("B", 2);
```

190130104118

Date
Page

```
Students s3 = new Students("C", 3);
Students s4 = new Students("D", 4);

Students[] Stud = new Students[4];

Stud[0] = S1;
Stud[1] = S2;
Stud[2] = S3;
Stud[3] = S4;

For (students s : stud)
{ System.out.println(s); }

    }
  }
}
```

(2) Create a Class with two overlouded Constructors The first Constructor is used for initializing, the name of the account holder, the account holder, the account number, and the initial amount in the account. The second Constructor is used for initializing the name of the account and the current balance. The account class is having the method Deposit(), withdraw and getbalance().

190130107118

Date
Page

```
Class Account {
  private string Account_HOL;
  Private int Account_num;
  Private int Initial_Amt;
  Private int Current_Bal;

  Public Account ()
  {
    Account_HOL = "Anony";
    Account_num = 31400l;
    Initial_Amt = 100;
  }

  Public Account (String name of Account)
  {
    Account_HOL = name of Account;
  }

  Public void Deposit (int Amount)
  {
    Current_Bal = Current_bal + Amount;
  }

  Public void withdraw (int amount)
  {
    if (Current_Bal < amount)
      System.out.println ("Your balance is low");
```

19013010718

Date
Page

```
else
    Current_Bal += amount;

Public int Get_Balance()
{
    return Current_Bal;
}

Public Class CC:-Bank {
    public Static Void main (String[] args) {

Scanner SC = new Scanner (System.in);

System.out.print ("Enter Name :");
String AccountHolder = new String (input.nextLine
                                   ());

Account A1 = new Account (AccountHolder);

System.out.println ("Name :" + A1.Account_HOL);
System.out.println ("Current Balance :" + A1.Get_Bal);
System.out.println ("Enter almost to deposit");

int deposit = input.nextInt();
A1.Deposit (deposit);

System.out.Println ("Current Balance :" + A1.Get_Bal
```

```
System.out.println("Enter amount to withdraw");
int WD = input.nextInt();

A1.withdraw(WD);
System.out.println("Current Balance:" + A.
                                getBalance());
  }

}
```

(3) The Airplane class has three subclasses named B747, B757 and B767. Each plane type can transport diff. no. of passengers. Each airplane object has a unique serial no. write an application that declares this class hierarchy. Initiate several types of airplanes and display them. Override toString() method of the object to return a string with the type, serial number and capacity.

```
Abstract class Airplane
{
   Public String serial-NO;
   Public int passenger-capacity;
   Public String model;

   @override
   Public abstract String toString();
}
```

Date
Page

```
Class B747 extends Airplane {
    public B747() {

        serialNo = "B747";
        passenger_capacity = 500;
        Model = "Boeing - B747";

    }
    @Override

    Public String toString() {
        return "serial no:" + serialNo + " Passenger
                capacity:" + passenger_capacity +
                "Model:" + model;

    }
}

Class B757 extends Airplane {
    public B757() {
        this.serialNo = "B757";
        this.passengers_capacity = 100;
        this.model = "Boeing B757";

    }
```

190130107118

Date
Page

```
@override
public String toString() {
    return "serial NO:" + serialNo + "Passenger
    capacity:" + Passenger_capacity + "Model:" +
    model;

    }
}

Class B767 extends Airplane {
    public B767() {
        this.serialNo = "B767";
        this.Passengers_capacity = 50;
        this.Model = "Boeing B767";
    }

}

Public class Q3_Airplane {
    public static void main (String[] args)

    Airplane plane_1 = new B747();
    Airplane plane_2 = new B767();
    Airplane plane_3 = new B757();

System.out.println ("Plane 1:" + plan_1.toString());
System.out.println ("plane 2:" + plane_2.toString());
System.Out.println ("plane 3:" + plant_3.toString());
}
```

190130107118

Date
Page

(4) Write a program that illustrates interfaces. Interface P is extended by P1 and P2. Interface P12 inherits from both P1 and P2. Each interface declares constant and one method class C implements P12. Instantiate C and invoke each of it's methods. Each method displays one of the constants.

→
```
interface P
{
  int p=0;
  void display_p();
}
interface P1 extends P{
      int p1=1;
      void display_p1();
  }

interface P2 extends P{
      int p2=2;
      void display_p2();
}

Interface P12 extends P1,P2{
      int p12=12;
      void display_p12();
}
```

190130107118

Date
Page

```
class A implements P12 {

    @override
    public void display_P() {
        System.out.println("Value of Interface P:" +P); }

    @override
    Public void display_P1() {
        System.out.println("Value of Interface P1:" +P1);
    }

    @override
    Public void display_P2() {
        System.out.println("Value of Interface P2:" +P2);}

    @override
    Public void display_P12() {
        System.out.println("Value of Interface P12:" +P12);
    }

}

Public class Cls_P12 {
    public static void main(String[] args) {
        A obj = new A();
        obj.display_P();
        obj.display_P1();
        obj.display_P2();
        obj.display_P12();
    }
}
```

1901301071118

Date
Page

(5) Write a program to implement an Abstract class Shape which contains Abstract method Area(), Create two other classes circle and square which overrides the method Area() and find the area of rectangle and square in respective class. Write demo class.

```
Abstract class shape{
    public class void calculate Area();
}

class circle extends shape{
    public double a;
    public circle() { this.a = a}
    public circle (double a)
      { this.a = a }

    Public void calculate Area() {

      System.out.println("Area of circle:" +
                  Math.PI * a * a);

    }
}

Class square extends shape {
    public double a;
    public square() { this.a = a;}
```

190130107118

Date
Page

```
Public square (double a)
    { this.a = a ;}

@ override
Public Void CulCulate Area() {
  System.out.println ("Area of square :" + a*a);
}

Class negativeValueExpection extends Exception {

@ override
public string getMessage () {
  return "Can not be negative :"); }

@override
public string toString () {
    return super.toString (); }

}

public class OS.Shape {
  public static Void main (String [] args) throws
    negative_value_Exception {

Scanner input = new scanner (System.in);
System.out.print ("Enter radius of circle");
double radius = input.nextInt ();
```

Date
Page

```
System.out.print ("Enter Side for square:");
double side = input.nextInt();

System.out.println (" ");


try {
    if (radius < 0)
      throw new negativeValueException();
    Shape S1 = new circle (radius);
    S1.calculateArea();

  }

catch (negativeValueException e)
  {
   System.out.println ("Radius" + getmessage());
  }


try {
    if (side < 0)
      throw new negativeValueException();
    Shape S2 = new square (side);
    S2.calculateArea();

}
```

```
catch (NegativeValueException e)
 {
    System.out.println ("sides" + e.Messages());
 }

 }

}
```

(6) Write an application that illustrates how a method can invoke a superclass method. Class I1 is extended by I2. Class I2 is extended by us. Each of these classes defines a getDiscription() method that returns a string. That string includes a description of the class plus description of each super class. Initiate each object of these classes and invoke the getDescription() method.

```
Class k2 {
      String sc;
      Public String getDescription () {
      return "I'm from k2 Class in super class" +
                  sc; }
}
```

190130107118

Date
Page

```
Class I2 extends K2 {
    public I2() { this.s( = "K2"; }

}

Class I2 extends I2 {
    public I2() { this.s( = "I2"; }
    public String getDescription() {
        return "I'm From I2 class \n super class
                + s(; }

Public class Q6_Super {
    public static void main (String [] args)

    K2 obj1 = new K2();
    I2 obj2 = new I2();
    I2 obj3 = new I2();

System.out.println (obj1. getDescription() + );
System.out.println (obj2.getDescription());
System.out.println (obj3.getDescription());
    }

}
```

19013010718

Date
Page

(7) Write a program to demonstrate the use of multithreading

```
Public class multithreading {
    public static void (String [] args)
    {
        Thread t3 = new Thread () -
        {
            for (int i=0; i<=5; i++) {
            System.out.println ("H: : ");

            try {
                Thread.sleep (500);
                }
            catch (InterruptedException e)
                {
                e.printStackTrace ();
                }
            }
        });

    Thread th = new Thread () →
    {
        for (int i=0; i<=5; i++)
        {
        System.out.println ("Hello. ");
```

1901301071B

Date
Page

```
try {
    Thread.sleep(500);
  }

  catch (InterruptedException e)
  {
      e.printStackTrace();
  }
 }
});

t3.start();

try Thread.sleep(10);

catch(exception e) {};

th.start();

}

}
```

190130107118

Date
Page

(6) Write a Program to demonstrate the use of exceptional handling

```
Public class Exceptional-Handling-1 {

    Public static void main (String [] args)

    Arithmetic Exceptional
     try {
         int num = 100/0;
     }

     Catch Arithmetic Exception (ex)
     {
       System.out.println (exc)
     }

     Catch (Exception e)
      {
         System.out.println(e);
      }

    System.out.println ("Runtime Exception :");

  }
}
```