**1st OBJECTIVE:**

Create a class Card, and its children classes are Valentine, Holiday, and Birthday. A card class will have a greeting () method that writes out a greeting. Each type of card contains an appropriate greeting. The Holiday card says "Season's Greetings." The Birthday card says "Happy Birthday." The Valentine card says "Happy Valentine Day". Create objects of the three child classes: Valentine, Holiday, and Birthday and show the polymorphic behavior (call the greeting methods from each object).

**PROGRAM # 1:**

SOURCE CODE:

class card{

void greeting()

{

System.out.println("Greatings");

}

}

class valentine extends card

{

void holiday()

{

System.out.println("Season Greatings"); }}

class holiday extends valentine

{

void valentine()

{

System.out.println("Happy valentine day");

}

void display()

{

greeting();

holiday();

valentine();

}}

public class carddemo

{

public static void main(String args[])

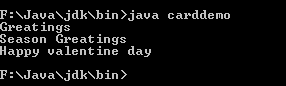
{

holiday obj = new holiday();

obj.display();

}}

OUTPUT:

  
CONCLUSION:

In this program we are learning how to apply method overloading.

**2nd OBJECTIVE:**

Create a class called Employee whose objects are records for an employee. This class will be a derived class of the class [Person](https://www.cs.bham.ac.uk/~mdr/teaching/RedHotChilli/java/Person.java) which you will have to copy into a file of your own and compile. An employee record has an employee's name (inherited from the class Person), an annual salary represented as a single value of type double, a year the employee started work as a single value of type int and a national insurance number, which is a value of type String.  
Your class should have a reasonable number of constructors and accessor methods, as well as an equals method. Write another class containing a main method to fully test your class definition.

**PROGRAM # 2:**

SOURCE CODE:

import java.util.\*;

class Person

{

String name1,name2,name3,name4;

double salary1,salary2;

int year1,year2;

public Person(String s1,String s2,String s3,String s4,double a1,double a2,int y1,int y2)

{

name1=s1;

name2=s2;

name3=s3;

name3=s4;

salary1=a1;

salary2=a2;

year1=y1;

year2=y2;

}

}

class Employ extends Person

{

public Employ(String s1,String s2,String s3,String s4,double a1,double a2,int y1,int y2)

{

super(s1, s2, s3, s4, a1, a2, y1, y2);

}

void Record1()

{

System.out.print("\n\n 1st Employ Record ");

System.out.print("\n Employ Full Name:- "+name1+name2);

System.out.print("\nEmploy Anual Salary:- "+salary1);

System.out.print("\n Emply Joining Year:- "+year1);

}

void Record2()

{

System.out.print("\n\n 2nd Employ Record ");

System.out.print("\n Employ Full Name:- "+name3+name4);

System.out.print("\nEmploy Anual Salary:- "+salary2);

System.out.print("\n Emply Joining Year:- "+year2);

}

}

public class perdemo

{

public static void main(String args[])

{

Scanner in=new Scanner(System.in);

System.out.print("\nEnter Your Data Here\n");

System.out.print("\n Employ First Name:- ");

String a=in.nextLine();

System.out.print(" Employ Last Name:- ");

String b=in.nextLine();

System.out.print("Employ Anual Salary:- ");

double c=in.nextDouble();

System.out.print("Employ Joining Year:- ");

int d=in.nextInt();

System.out.print("\n");

String i=in.nextLine();

System.out.print("\n Employ First Name:- ");

String e=in.nextLine();

System.out.print(" Employ Last Name:- ");

String f=in.nextLine();

System.out.print("Employ Anual Salary:- ");

double g=in.nextDouble();

System.out.print("Employ Joining Year:- ");

int h=in.nextInt();

Person obj1=new Person(a,b,e,f,c,g,d,h);

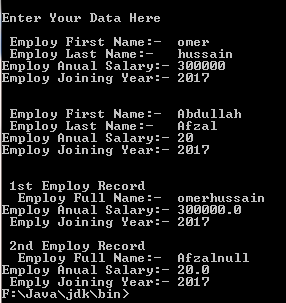
Employ obj2=new Employ(a,b,e,f,c,g,d,h);

obj2.Record1();

obj2.Record2();

}}

OUTPUT:



CONCLUSION:

In this program we are learning how to extend classes.

**3rd OBJECTIVE:**

 Derive a subclass called Cylinder from the superclass Circle, which we have created in the previous lab. It is important to note that we reuse the class Circle. Reusability is one of the most important properties of OOP. (Why reinvent the wheels?) The class Cylinder inherits all the member variables (radius and color) and methods (getRadius(), getArea(), among others) from its superclass Circle. It further defines a variable called height, two public methods - getHeight() and getVolume() and its own constructors.

**PROGRAM # 3:**

SOURCE CODE:

class Circle

{

double rad;

String color="Red";

static double PI = 3.142;

Circle(double x)

{

rad=x;

this.color=color;

}

double getRadius()

{

System.out.print("\n Colour Of Circle = "+color);

return rad;

}

double getArea()

{

double e=2\*PI\*rad;

return e;

}

}

class Cylinder extends Circle

{

private double height;

static double PI = 3.142;

Cylinder(double x)

{

super(x);

}

double getHeight(double y)

{

height=y;

return y;

}

double getVolume()

{

double f=PI\*(rad\*rad)\*height;

return f;

}

}

public class cylindemo

{

public static void main(String args[])

{

System.out.print("\n\t CIRCLE ");

Circle obj1=new Circle(10.2);

double a=obj1.getRadius();

System.out.print("\n Radius Of Circle = "+a);

double b=obj1.getArea();

System.out.print("\n Area Of Circle = "+b);

System.out.print("\n\n\n\t CYLINDER ");

Cylinder obj2=new Cylinder(10.2);

double c=obj2.getHeight(2);

System.out.print("\nHeight of Cylinder = "+c);

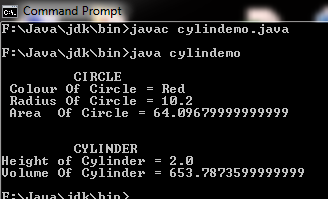
double d=obj2.getVolume();

System.out.print("\nVolume Of Cylinder = "+d);

System.out.print("\n");

}}

OUTPUT:



CONCLUSION:

In this program we are learning how to reuse class.

**4th OBJECTIVE:**

Develop a registration system for a University.  It should consist of three classes namely Student, Teacher, and Course.  For our example, a student needs to have a name, roll number, address and GPA to be eligible for registration. Therefore choose appropriate data types for encapsulating these properties in a Student object/s.  Similarly a teacher needs to have name, address, telephone number, and a degree (or a list of degrees) he has received.  Finally courses must have a title and course ID, students (5 students) registered for the course, and a teacher assigned to conduct the course. Create an object of Course with 5 Students and a Teacher. A call to a method, say printDetails(), of the selected course reference should print name of the course, details of the teacher assigned to that course, and names and roll numbers of the students enrolled with the course.

**PROGRAM # 4:**

SOURCE CODE:

class Student

{

String name;

int rollNo;

String add;

double gpa;

Student ( String name, int rollNo, String add, double gpa)

{

this.name = name;

this.rollNo = rollNo;

this.add = add;

this.gpa = gpa;

}

}

class Teacher

{

String name;

String add;

int tel;

String deg;

Teacher ( String name, String add, int tel, String deg )

{

this.name = name;

this.add = add;

this.tel = tel;

this.deg = deg;

}

}

class Course

{

String name;

Student std[] = new Student[5];

Teacher teach;

Course ( String name, Teacher teach, Student std[] )

{

this.name = name;

this.std = std;

this.teach = teach;

}

void printDetails() {

System.out.println("Course Name: "+this.name);

System.out.println("Course Teacher: "+this.teach.name);

for ( int i = 0 ; i < 5 ; i++ ) {

System.out.println((i+1)+". "+this.std[i].name+" ("+this.std[i].rollNo+")"); } }}

public class pg2

{

public static void main(String[] args)

{

Student std[] = new Student[5];

std[0] = new Student("Omer",258,"FB Area",3.9);

std[1] = new Student("Ali",66,"Gulshan",2.8);

std[2] = new Student("Abdullah",176,"Gujar Nala",2.86);

std[3] = new Student("Faisal",172,"Anna Mor",2.6);

std[4] = new Student("Mujtaba",177,"Scheme33",2.8);

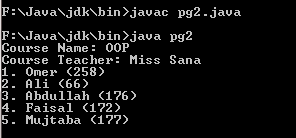
Teacher teach = new Teacher( "Miss Sana", "R-431, Gulistna-e-Jogar", 34567892, "M.S in Software Eng" );

Course c = new Course( "OOP", teach, std );

c.printDetails();

}}

OUTPUT:



CONCLUSION:

In this program we are learning how to develop a registration system.