**PROGRAM 1**

Write a Java program to show that a private member of a super class cannot be accessed from derived classes.

class room

{

private int l,b;

room(int x,int y)

{

l=x; b=y;

}

int area()

{

return(l\*b);

}

}

class class\_room extends room

{

int h;

class\_room(int x,int y,int z)

{

super(x,y);

h=z;

}

int volume()

{

return(area()\*h);

}

}

class Super

{

public static void main(String args[])

{

class\_room cr=new class\_room(18,45,55);

int a1=cr.area();

int v1=cr.volume();

System.out.println("Area of Room : "+a1);

System.out.println("Volume of Room : "+v1);

System.out.println ("Akshat Negi");

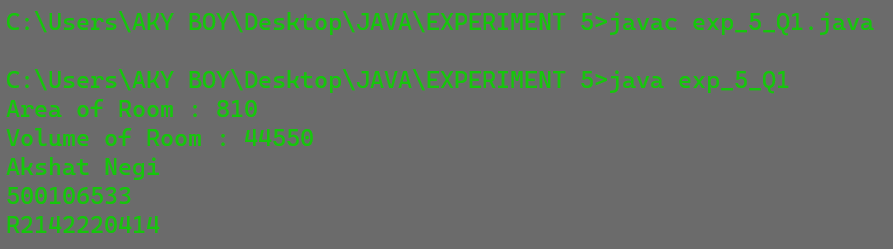
System.out.println ("500106533");

System.out.println ("R2142220414");

}

}

**OUTPUT**



**PROGRAM 2**

Write a program in Java to create a Player class. Inherit the classes Cricket Player, Football \_Player, and Hockey Player from Player class.

class player

{

String name;

int age;

player(String n,int a)

{

name=n; age=a;

}

void show()

{

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

System.out.println("Player name : "+name);

System.out.println("Age : "+age);

}

}

class cricket\_player extends player

{

String type;

cricket\_player(String n,String t,int a)

{

super(n,a);

type=t;

}

public void show()

{

super.show();

System.out.println("Player type : "+type);

}

}

class football\_player extends player

{

String type;

football\_player(String n,String t,int a)

{

super(n,a);

type=t;

}

public void show()

{

super.show();

System.out.println("Player type : "+type);

}

}

class hockey\_player extends player

{

String type;

hockey\_player(String n,String t,int a)

{

super(n,a);

type=t;

}

public void show()

{

super.show();

System.out.println("Player type : "+type);

}

}

class HCPlayer

{

public static void main(String args[])

{

cricket\_player c=new cricket\_player("Akshu","Cricket",25);

football\_player f=new football\_player("Yogi","Football",15);

hockey\_player h=new hockey\_player("Mahi","Hockey",29);

c.show();

f.show();

h.show();

System.out.println ("Akshat Negi");

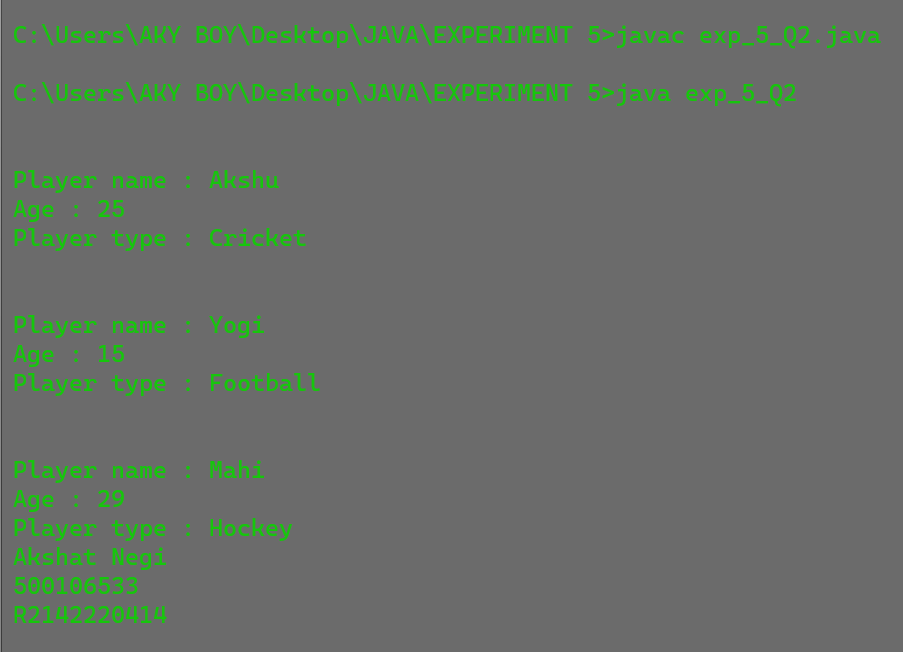
System.out.println ("500106533");

System.out.println ("R2142220414");

}

}

**OUTPUT**



**PROGRAM 3**

Write a class Worker and derive classes DailyWorker and SalariedWorker from it. Every worker has a name and a salary rate. Write the method ComPay (int hours) to compute the weekly pay of every worker. A Daily Worker is paid on the basis of the number of days he/she works. The Salaried Worker gets paid the wage for 40 hours a week no matter what the actual hours are. Test this program to calculate the pay of workers. You are expected to use the concept of polymorphism to write this program.

class worker

{

String name;

int empno;

worker(int no,String n)

{ empno=no; name=n; }

void show()

{

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

System.out.println("Employee number : "+empno);

System.out.println("Employee name : "+name);

}

}

class dailyworker extends worker

{

int rate;

dailyworker(int no,String n,int r)

{

super(no,n);

rate=r;

}

void compay(int h)

{

show();

System.out.println("Salary : "+rate\*h);

}

}

class salariedworker extends worker

{

int rate;

salariedworker(int no,String n,int r)

{

super(no,n);

rate=r;

}

int hour=40;

void compay()

{

show();

System.out.println("Salary : "+rate\*hour);

}

}

class Pay

{

public static void main(String args[])

{

dailyworker d=new dailyworker(2546,"Yogesh",70);

salariedworker s=new salariedworker(9000,"Aadi",800);

d.compay(45);

s.compay();

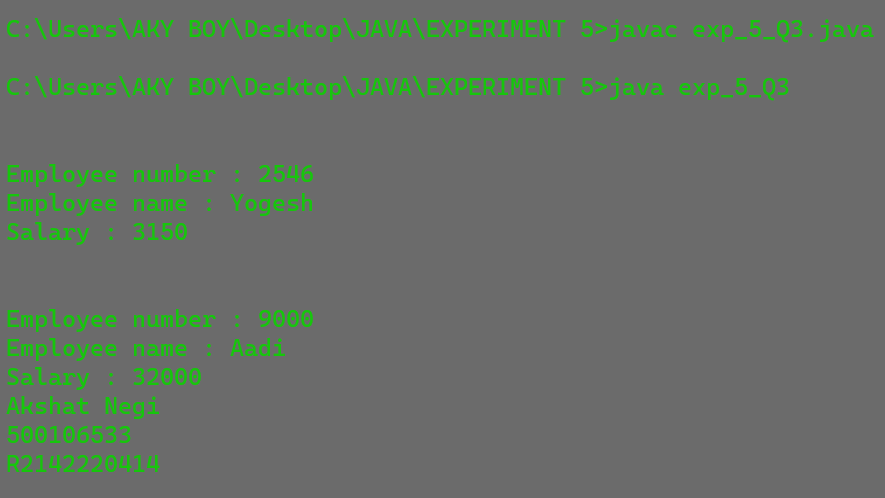
System.out.println ("Akshat Negi");

System.out.println ("500106533");

System.out.println ("R2142220414");

}

**OUTPUT**



**PROGRAM 4**

Design a class employee of an organization. An employee has a name, empid, and salary. Write the default constructor, a constructor with parameters (name, empid, and salary) and methods to return name and salary. Also, write a method increaseSalary that raises the employee’s salary by a certain user-specified percentage. Derive a subclass Manager from the employee. Supply a test program that uses these classes and methods.

class employee

{

int empid;

String name;

double salary;

employee()

{

empid=500106533;

name="Akshu";

salary=90000;

}

employee(String name,int empid,double salary)

{

this.empid=empid;

this.name=name;

this.salary=salary;

}

String getName()

{

return name;

}

double getSalary()

{

return salary;

}

double increaseSalary(double x)

{

salary=salary+(salary\*x);

return salary;

}

}

class manager extends employee

{

double r=0.5;

manager()

{

super();

}

manager(String name, int empid, double salary)

{

super(name,empid,salary);

salary= increaseSalary(r);

}

}

class Empl

{

public static void main(String[] args)

{

manager m=new manager("Akshu" , 500106533 , 150000);

System.out.println("NAME:" +m.name);

System.out.println("ID:" +m.empid);

System.out.println("Salary:" +m.salary);

System.out.println ("Akshat Negi");

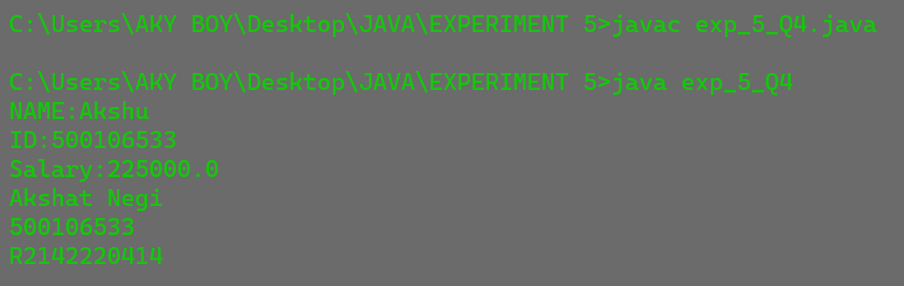
System.out.println ("500106533");

System.out.println ("R2142220414");

}

}

**OUTPUT**



**PROGRAM 5**

Write a JAVA program to implement constructor overloading.

class Rectangle

{

private int length;

private int breadth;

public Rectangle(int side)

{

length = side;

breadth = side;

}

public Rectangle(int l, int b)

{

length = l;

breadth = b;

}

public int getArea()

{

return length \* breadth;

}

}

class Test

{

public static void main(String[] args)

{

Rectangle rect = new Rectangle(6, 3);

Rectangle sq = new Rectangle(8);

System.out.println(rect.getArea());

System.out.println(sq.getArea());

System.out.println ("Akshat Negi");

System.out.println ("500106533");

System.out.println ("R2142220414");

}

}

**OUTPUT**

