**EXPERIMENT – 4**

**TITLE Inheritance**

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

**i. with private class it shows error**

class parent{

private int a =10;

private void A(){

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

}

}

class derived extends parent{

public static void main(String args[]){

int c;

parent obj= new parent();

c=a.parent;

System.out.println(c);

}

}

**ii. access given using a method.**

class parent{

private int a =10;

private void A(){

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

}

int asc(){

return a;

}

}

class derived extends parent{

public static void main(String args[]){

int c;

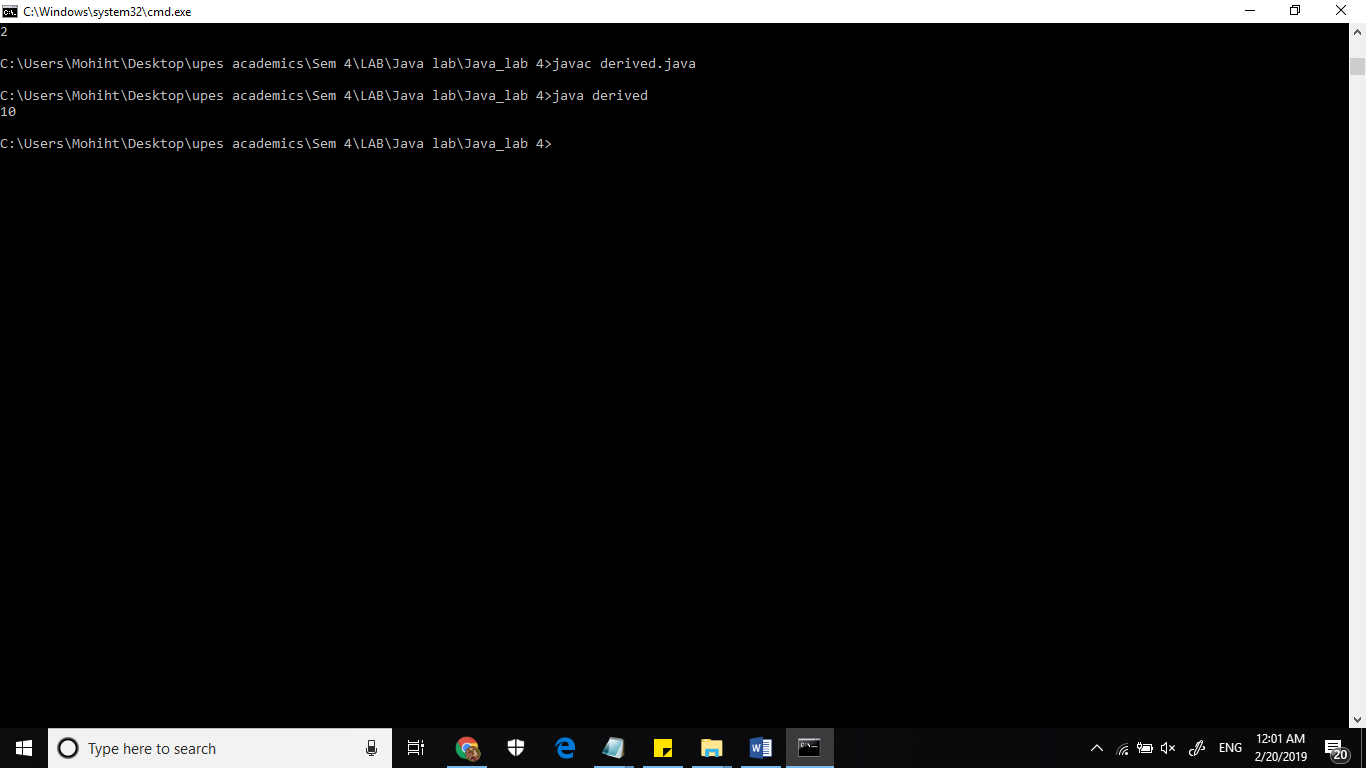
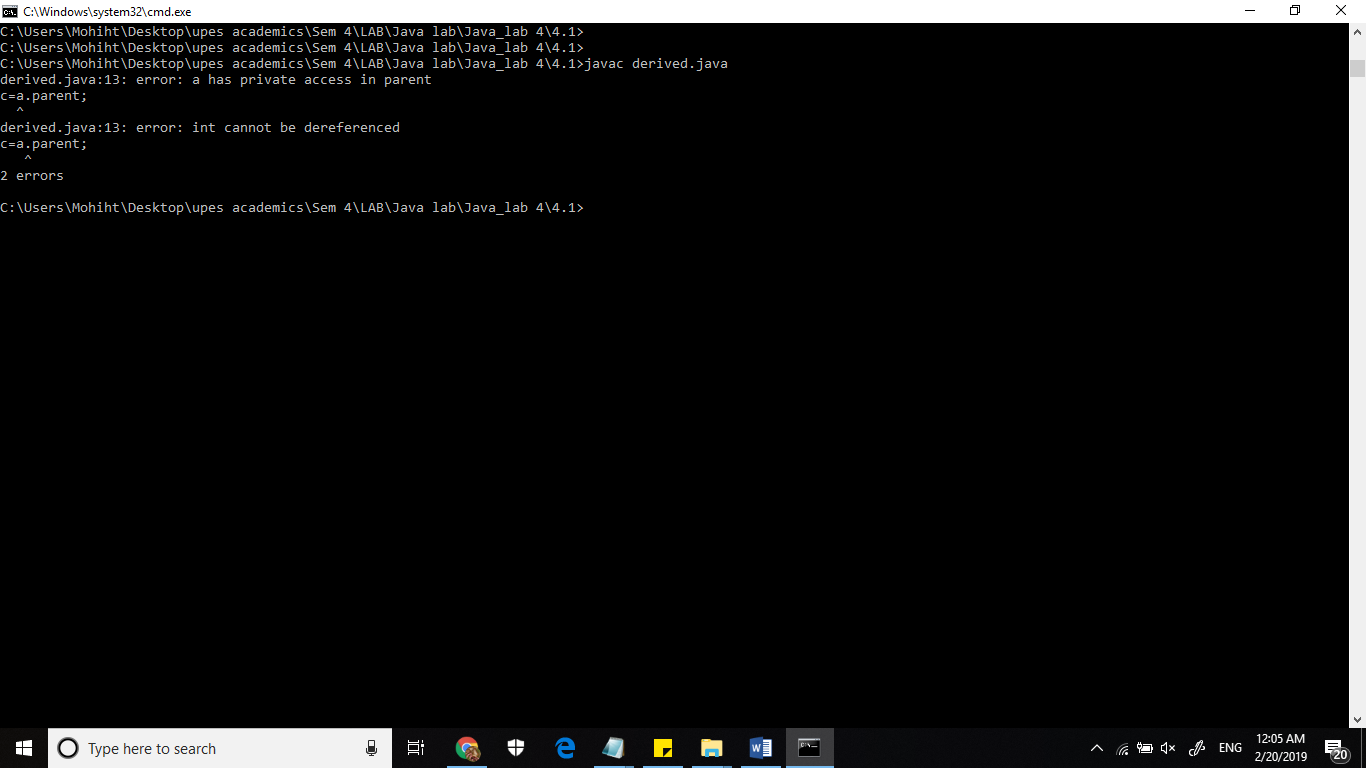
parent obj= new parent();

c=obj.asc();

System.out.println(c);

}

}



1. **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{

public void pp(){

int p=10;

System.out.println(p);

}

}

class hockey\_player extends player{

void show1(){

System.out.println("1");

}

}

class cricket\_player extends player{

void show2(){

System.out.println("2");

}

}

class football\_player extends player{

void show3(){

System.out.println("3");

}

}

class test{

public static void main(String args[]){

football\_player obj1 = new football\_player();

obj1.pp();

obj1.show3();

hockey\_player obj2= new hockey\_player();

obj2.pp();

obj2.show1();

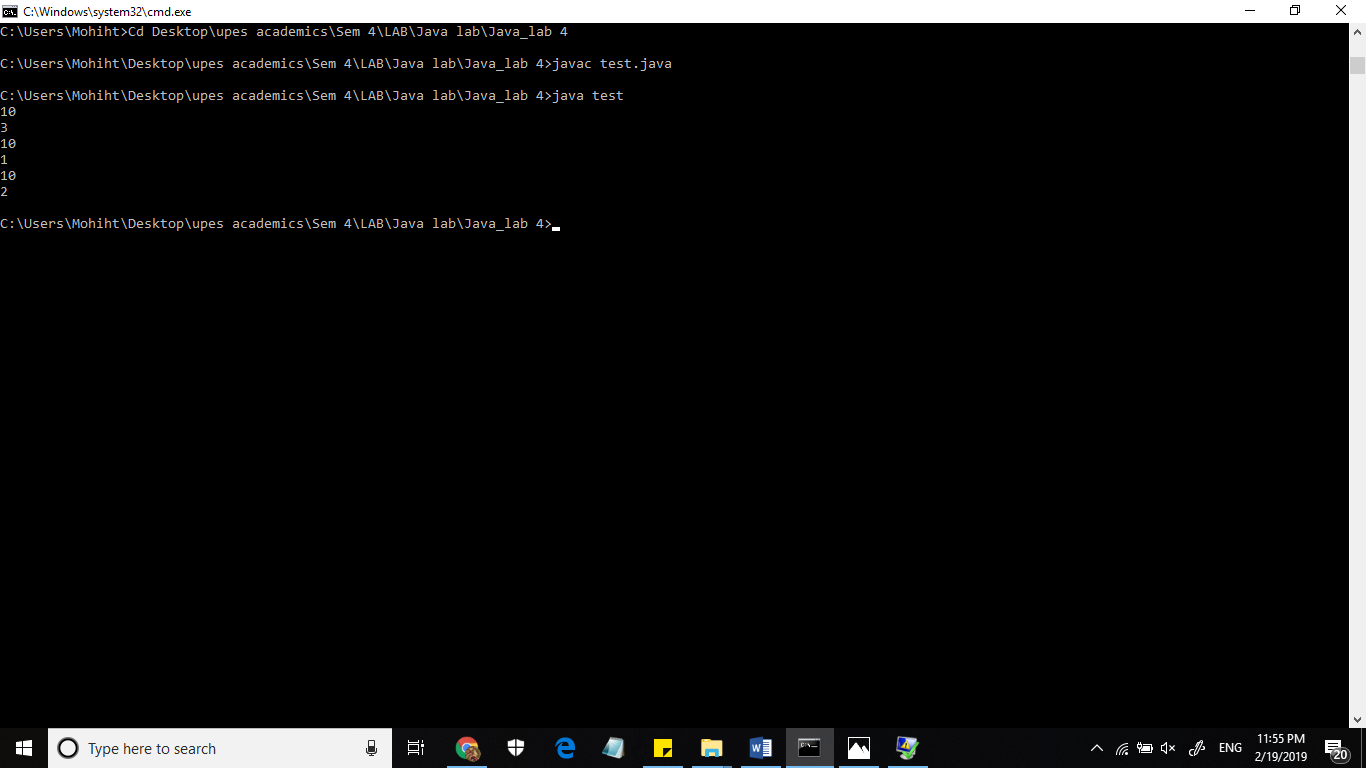
cricket\_player obj3= new cricket\_player();

obj3.pp();

obj3.show2();

}

}



1. **Write a class Worker and derive classes DailyWorker and SalariedWorker from it. Every worker has a name and a salary rate. Write method ComPay (int hours) to compute the week 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.**

import java.util.\*;

class worker{

int wage2;

double wage1;

Scanner sc= new Scanner(System.in);

int i;

worker(){

String name;

System.out.println("enter the name of the employee");

name=sc.nextLine();

System.out.println("name of employee is" +name);

}

void ComPay(double week){

double wage1= 40\*100\*week;

System.out.println(wage1);

}

void ComPay(int day){

wage2=day\*800;

System.out.println(wage2);

}

}

class SalariedWorker extends worker{

void show1(){

System.out.println("wage of salaried worker is"+ wage1);

}

}

class DailyWorker extends worker{

void show2(){

System.out.println("wage of daily worker is"+ wage2);

}

}

class workerdr{

public static void main(String args[]){

SalariedWorker sw1 =new SalariedWorker();

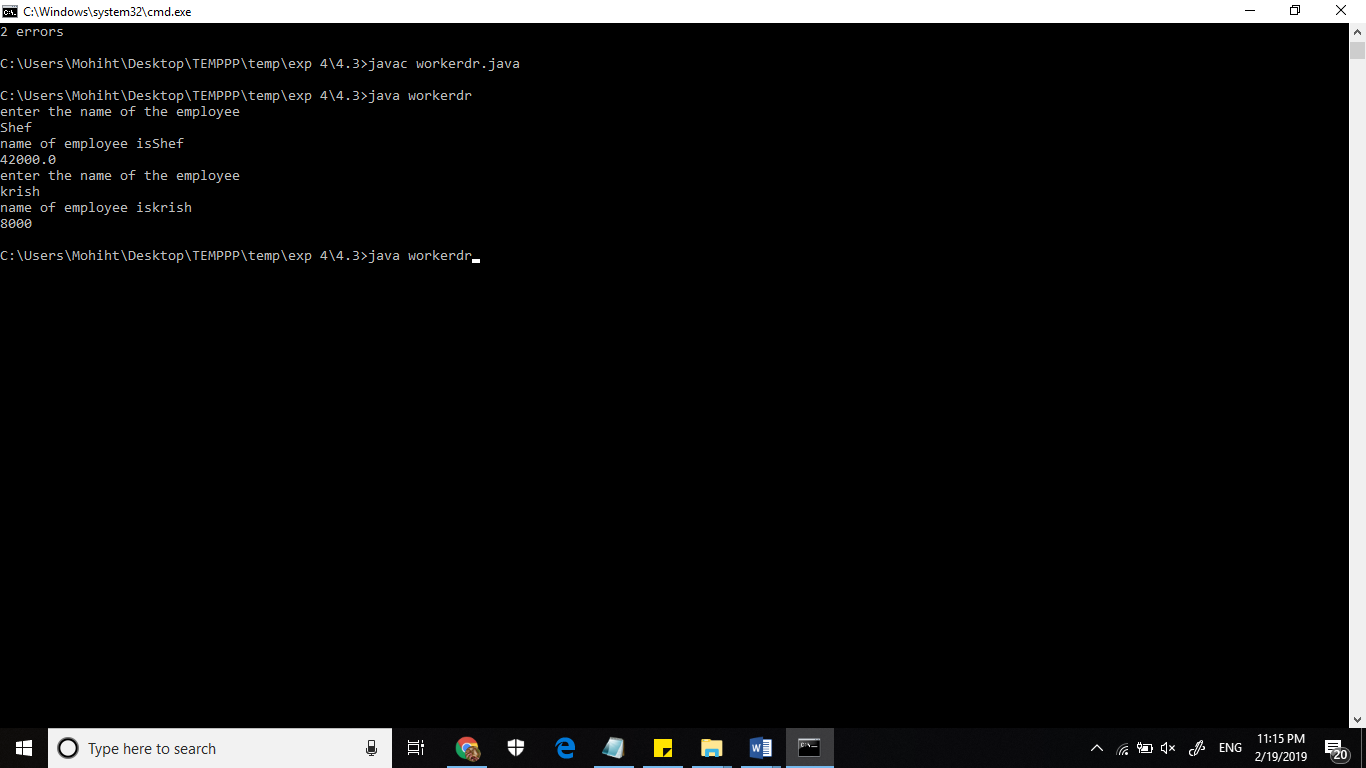
sw1.ComPay(10.5);

DailyWorker dw1= new DailyWorker();

dw1.ComPay(10);

}

}



1. **Consider the trunk calls of a telephone exchange. A trunk call can be ordinary, urgent or lightning. The charges depend on the duration and the type of the call. Write a program using the concept of polymorphism in Java to calculate the charges.**

import java.util.Scanner;

class trunk{

public static void main(String args[]){

int dur;

System.out.println("what kind of call do you want ot make \t 1. ordianry 2.urgent 3.lightning");

Scanner sc= new Scanner(System.in);

int choice=sc.nextInt();

switch(choice){

case 1:

charge c1=new charge();

break;

case 2:

System.out.println("enter duration");

dur=sc.nextInt();

charge c2=new charge(dur);

break;

case 3:

System.out.println("enter duration in decimal");

dur=sc.nextInt();

charge c3=new charge(dur);

break;

default:

System.out.println("error try again");

}

}

}

class charge{

int rate;

charge(){

rate=100;

System.out.println("your fixed charges are" + rate);

}

charge(int dur){

rate=20\*dur;

System.out.println("your charges are" + rate);

}

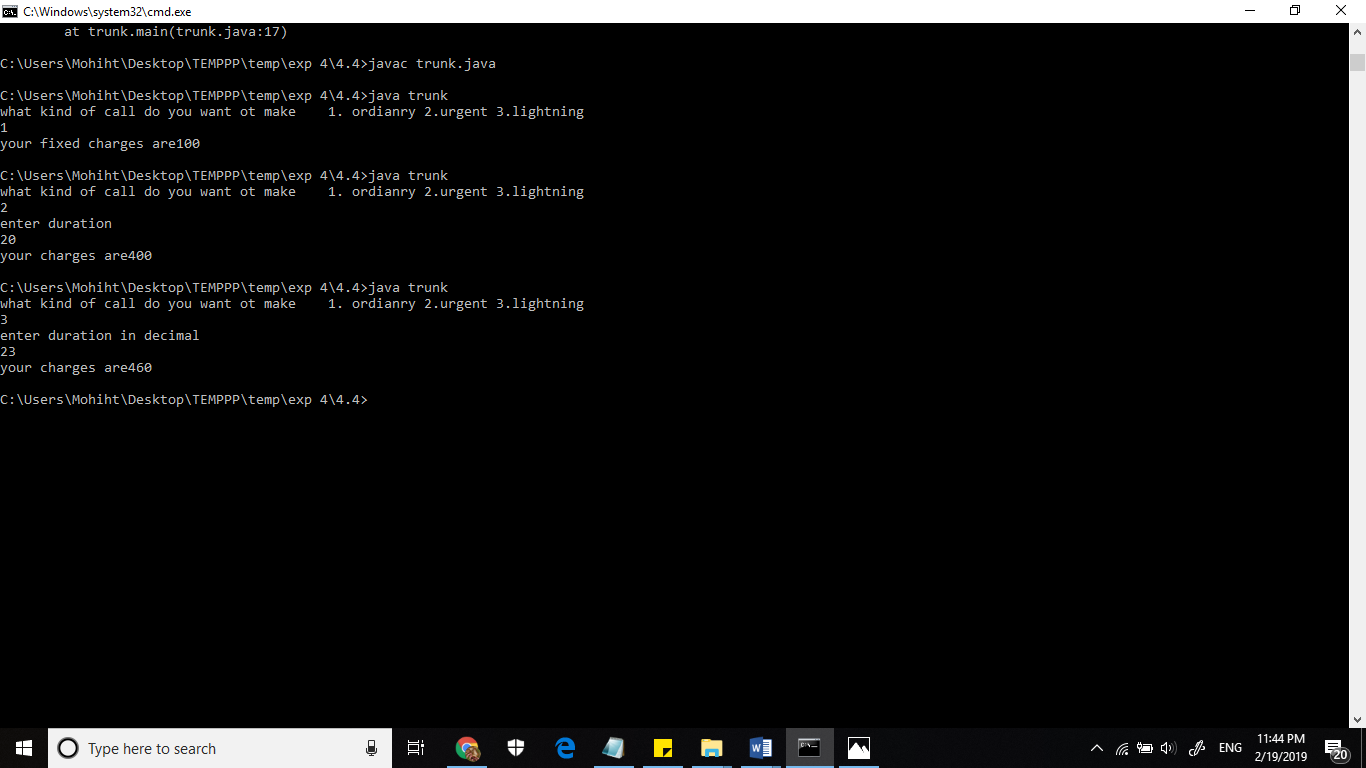
charge(double dur){

rate=(int)(30\*dur);

System.out.println("your charges are" + rate);

}

}



1. **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 employee. Add an instance variable named department to the manager class. Supply a test program that uses theses classes and methods.**

import java.util.\*;

class Employee{

Scanner sc= new Scanner(System.in);

String name;

int empid;

int salary;

Employee(){

System.out.println("enter name");

name =sc.nextLine();

System.out.println("Enter Empid");

empid =sc.nextInt();

System.out.println("Enter salary");

salary=sc.nextInt();

}

Employee(String name,int empid,int salary){

System.out.println("name is\t"+name);

System.out.println("Employee id is\t"+ empid);

System.out.println(" old salary is\t"+ salary);

}

int increaseSalary(){

salary = (int) (salary+salary\*0.2);

return salary;

}

String method1(){

this.name=name;

return name;

}

int method2(){

this.salary=salary;

return salary;

}

}

class Manager extends Employee{

Scanner sc= new Scanner(System.in);

String department;

Manager(){

System.out.println("Enter department");

department=sc.nextLine();

System.out.println("department is\t"+department);

}

}

class testclass{

public static void main(String args[]){

Scanner sc= new Scanner(System.in);

Manager m1= new Manager();

Employee e1= new Employee();

e1.increaseSalary();

System.out.println("Name of Employee is\t" +e1.method1());

System.out.println("New updated salary is\t" +e1.method2());

System.out.println("Name of Employee is\t" +m1.method1());

System.out.println("New updated salary is\t" +m1.method2());

Employee e2=new Employee("shef",12,13);

}

}

