JAVA PROGRAMMING

QUESTION

Submitted by

Tamilvanan B.

2018503566.

MO Batch.

09-25-2020.

1. Create an interface for library application add atleast two interface methods for renewal of books and renewal day calculation and amount calculation. Create two classes as student and faculty. Try to implement the interface methods in the classes.

Program:

import java.util.\*;

public class Library {

public static void main(String arg[])

{

System.out.println("Faculty or Student ?(F/S)");

Scanner sc = new Scanner(System.in);

char x = sc.next().charAt(0);

if(x == 'F')

{

Faculty f = new Faculty();

int day;

System.out.println("Enter the current day(time lapsed)");

day = sc.nextInt();

int n = f.calculateRenewalTime(day);

if(n > 0)

{

System.out.println("There are " +n+ " days for the renewal");

}

else

{

System.out.println("You're " +(n \* -1)+ " days late for the renewal");

}

n = f.amountCalculation(day);

System.out.println("The Total cost to be paid is " +n);

}

else

{

Student f = new Student();

int day;

System.out.println("Enter the current day(time lapsed)");

day = sc.nextInt();

int n = f.calculateRenewalTime(day);

if(n > 0)

{

System.out.println("There are " +n +" days for the renewal");

}

else

{

System.out.println("You're " +(n \* -1)+ " days late for the renewal");

}

n = f.amountCalculation(day);

System.out.println("The total cost to be paid is " +n);

}

}

}

interface LibraryMangement {

public int calculateRenewalTime(int currentDay);

public int amountCalculation(int currentDay);

}

class Student implements LibraryMangement{

public int calculateRenewalTime(int currentDay)

{

return 14 - currentDay;

}

public int amountCalculation(int currentDay)

{

if(currentDay > 14)

{

return (currentDay - 14) \* 15;

}

return 0;

}

}

class Faculty implements LibraryMangement {

public int calculateRenewalTime(int currentDay)

{

return 21 - currentDay;

}

public int amountCalculation(int currentDay)

{

if(currentDay > 21)

{

return (currentDay - 21) \* 5;

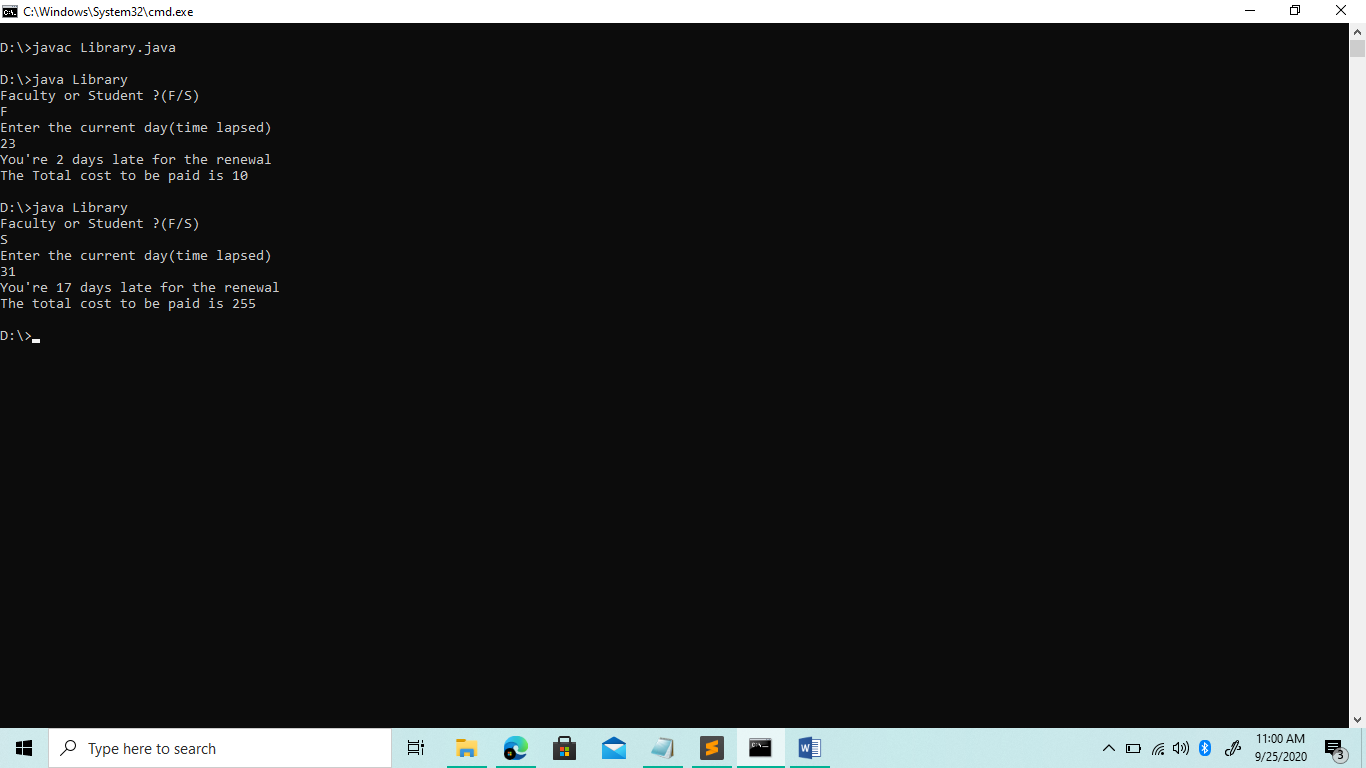
}

return 0;

}

}

Output:



2. create an abstruct class Student with concreate method and abstruct method. Try to create a class Fulltime and define the abstuct class inside fulltime. Create an object referecne to refer the fulltime object and access the abstruct method.

Program:  
abstract class Student{

Student(){}

abstract void display();

void dispalymarks(){

System.out.println("Abstract method executed");

}

}

class FullTime extends Student{

FullTime(){}

public void display(){

System.out.println("Concrete Method executed");

}

}

public class Abstructdemo{

public static void main(String args[]){

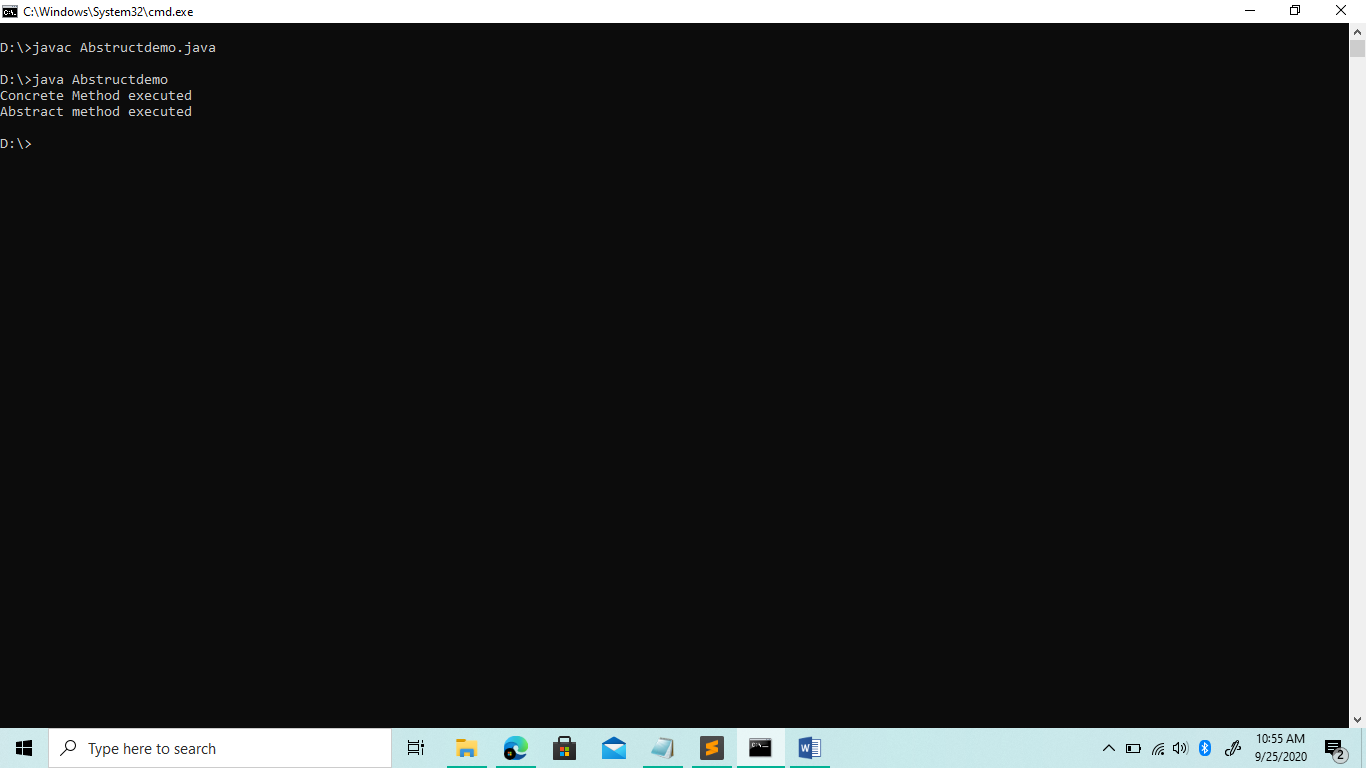
Student f = new FullTime();//refer the fulltime object and access the abstruct method

f.display();

f.dispalymarks();

}

}

Output