ASSIGNMENT – 1

Subject: CSW2 (CSE 2141)

Name: Arpit Kumar Mohanty

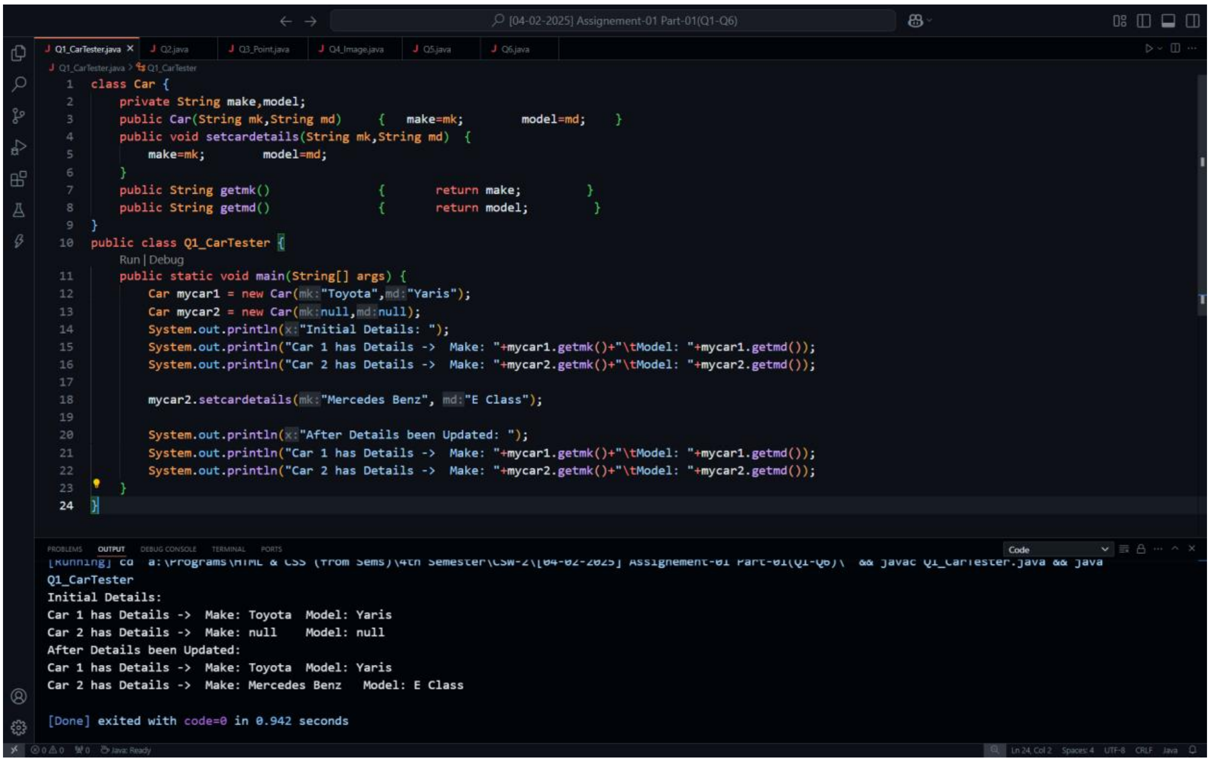
Registration Number: 2341013237

Section: 23412G1

Branch: CSE

Q1. Write a Java program with a Car class having private fields (make, model), a parameterized constructor, getter, and setter methods. In the CarTester class, instantiate myCar1 with values and myCar2 with null. Print their initial details, update myCar2 using setters, and print the updated details.

Solution along with Output:



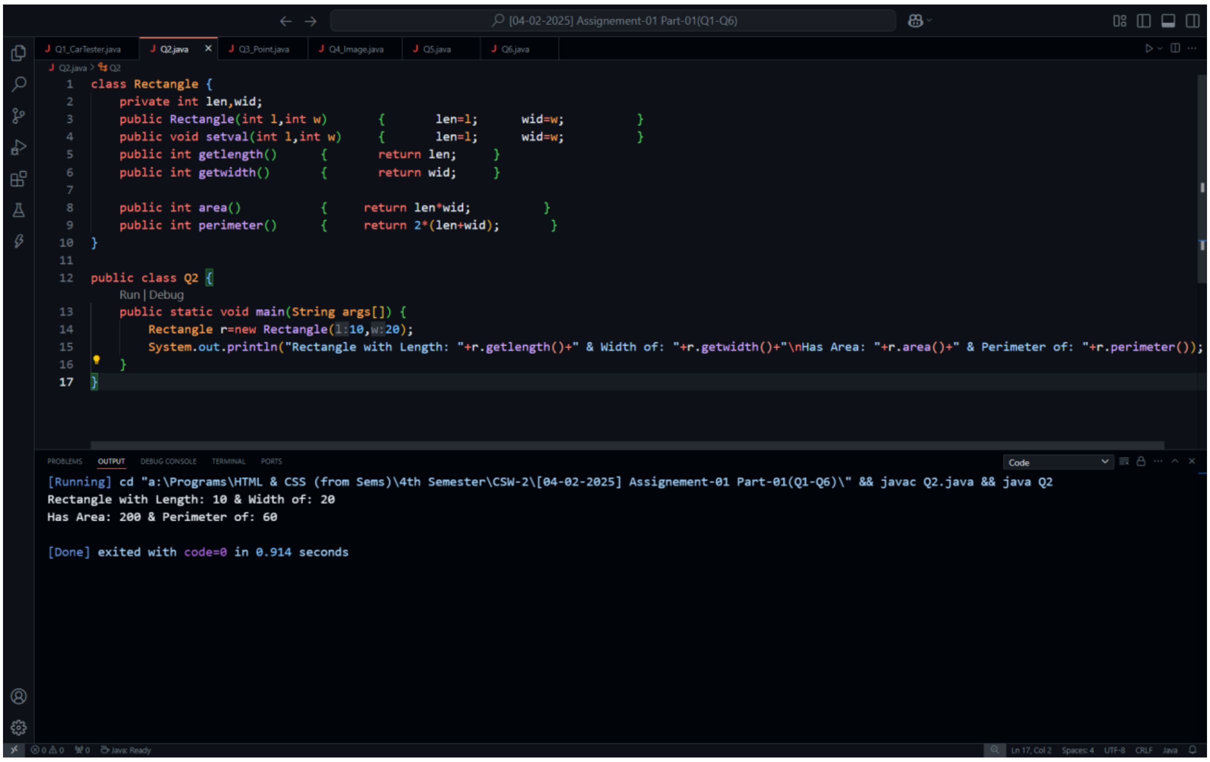
```java
class Car {
    private String make,model;
    public Car(String mk,String md)     {   make=mk;        model=md;    }
    public void setcardetails(String mk,String md)  {
        make=mk;        model=md;
    }
    public String getmk()               {       return make;        }
    public String getmd()               {       return model;        }
}
public class Q1_CarTester {
    public static void main(String[] args) {
        Car mycar1 = new Car(mk:"Toyota",md:"Yaris");
        Car mycar2 = new Car(mk:null,md:null);
        System.out.println(x:"Initial Details: ");
        System.out.println("Car 1 has Details ->  Make: "+mycar1.getmk()+"\tModel: "+mycar1.getmd());
        System.out.println("Car 2 has Details ->  Make: "+mycar2.getmk()+"\tModel: "+mycar2.getmd());

        mycar2.setcardetails(mk:"Mercedes Benz", md:"E Class");

        System.out.println(x:"After Details been Updated: ");
        System.out.println("Car 1 has Details ->  Make: "+mycar1.getmk()+"\tModel: "+mycar1.getmd());
        System.out.println("Car 2 has Details ->  Make: "+mycar2.getmk()+"\tModel: "+mycar2.getmd());
    }
}
```

```
[Running] cd  a:\Programs\HTML & CSS (from Sems)\4th Semester\CSW-2\[04-02-2025] Assignement-01 Part-01(Q1-Q6)\  && javac Q1_CarTester.java && java Q1_CarTester
Initial Details:
Car 1 has Details ->  Make: Toyota  Model: Yaris
Car 2 has Details ->  Make: null    Model: null
After Details been Updated:
Car 1 has Details ->  Make: Toyota  Model: Yaris
Car 2 has Details ->  Make: Mercedes Benz   Model: E Class

[Done] exited with code=0 in 0.942 seconds
```

Q2. Design a Java class called Rectangle with private attributes length and width. Include a constructor to initialize these attributes, as well as setter and getter methods for each attribute. Additionally, implement

methods to calculate the area and perimeter of the rectangle. Write a main method to create an object of the Rectangle class, set values for its attributes, and display the area and perimeter.

Solution along with Output:

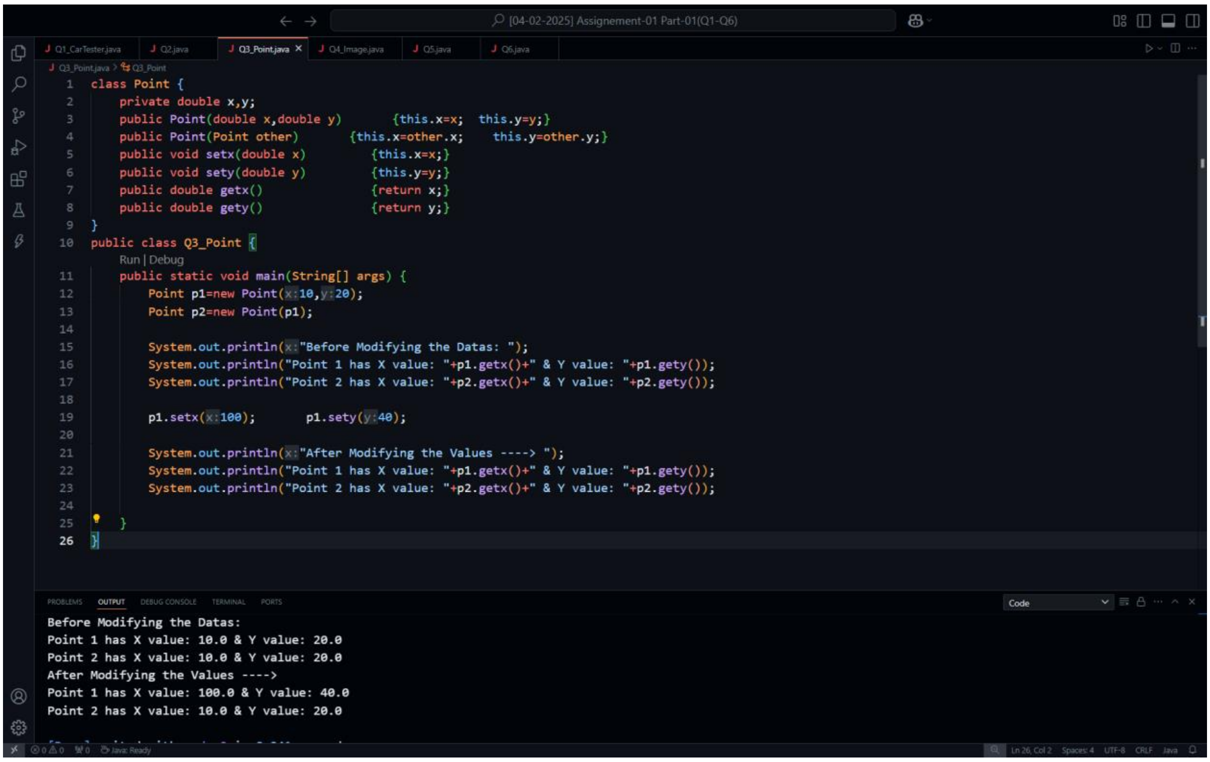


```java
class Rectangle {
    private int len,wid;
    public Rectangle(int l,int w)       {       len=l;      wid=w;          }
    public void setval(int l,int w)     {       len=l;      wid=w;          }
    public int getlength()      {       return len;     }
    public int getwidth()       {       return wid;     }

    public int area()           {     return len*wid;           }
    public int perimeter()      {     return 2*(len+wid);        }
}

public class Q2 {
    public static void main(String args[]) {
        Rectangle r=new Rectangle(10,20);
        System.out.println("Rectangle with Length: "+r.getlength()+" & Width of: "+r.getwidth()+"\nHas Area: "+r.area()+" & Perimeter of: "+r.perimeter());
    }
}
```

```
[Running] cd "a:\Programs\HTML & CSS (from Sems)\4th Semester\CSW-2\[04-02-2025] Assignement-01 Part-01(Q1-Q6)\" && javac Q2.java && java Q2
Rectangle with Length: 10 & Width of: 20
Has Area: 200 & Perimeter of: 60

[Done] exited with code=0 in 0.914 seconds
```

Q3. Write a Java program that defines a Point class with attributes X and Y, and includes a parameterized constructor to initialize these attributes. Implement a copy constructor to create a new point object with the same attribute values. Ensure that modifications made to one object do not affect the other. Utilize getter and setter methods to retrieve and update the attribute values.

Solution along with Output:

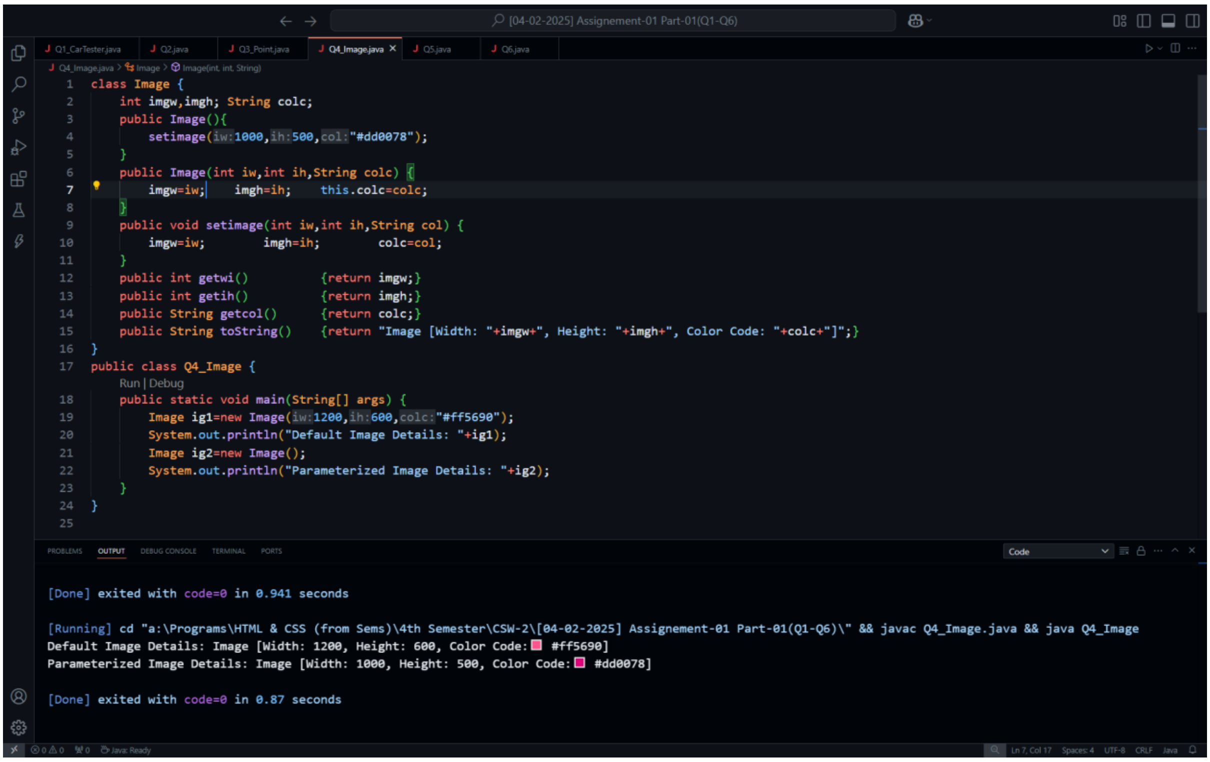


```java
class Point {
    private double x,y;
    public Point(double x,double y)       {this.x=x;  this.y=y;}
    public Point(Point other)       {this.x=other.x;    this.y=other.y;}
    public void setx(double x)          {this.x=x;}
    public void sety(double y)          {this.y=y;}
    public double getx()                {return x;}
    public double gety()                {return y;}
}
public class Q3_Point {
    public static void main(String[] args) {
        Point p1=new Point(10,20);
        Point p2=new Point(p1);

        System.out.println("Before Modifying the Datas: ");
        System.out.println("Point 1 has X value: "+p1.getx()+" & Y value: "+p1.gety());
        System.out.println("Point 2 has X value: "+p2.getx()+" & Y value: "+p2.gety());

        p1.setx(100);       p1.sety(40);

        System.out.println("After Modifying the Values ----> ");
        System.out.println("Point 1 has X value: "+p1.getx()+" & Y value: "+p1.gety());
        System.out.println("Point 2 has X value: "+p2.getx()+" & Y value: "+p2.gety());

    }
}
```

```
Before Modifying the Datas:
Point 1 has X value: 10.0 & Y value: 20.0
Point 2 has X value: 10.0 & Y value: 20.0
After Modifying the Values ---->
Point 1 has X value: 100.0 & Y value: 40.0
Point 2 has X value: 10.0 & Y value: 20.0
```

Q4. Write a program to create an Image class with attributes imageWidth, imageHeight, and colorCode. Add the required constructor, set methods, get methods, and toString method. Create the object of the image class using the default and parameterized constructor and print the details of the object.

Solution along with Output:

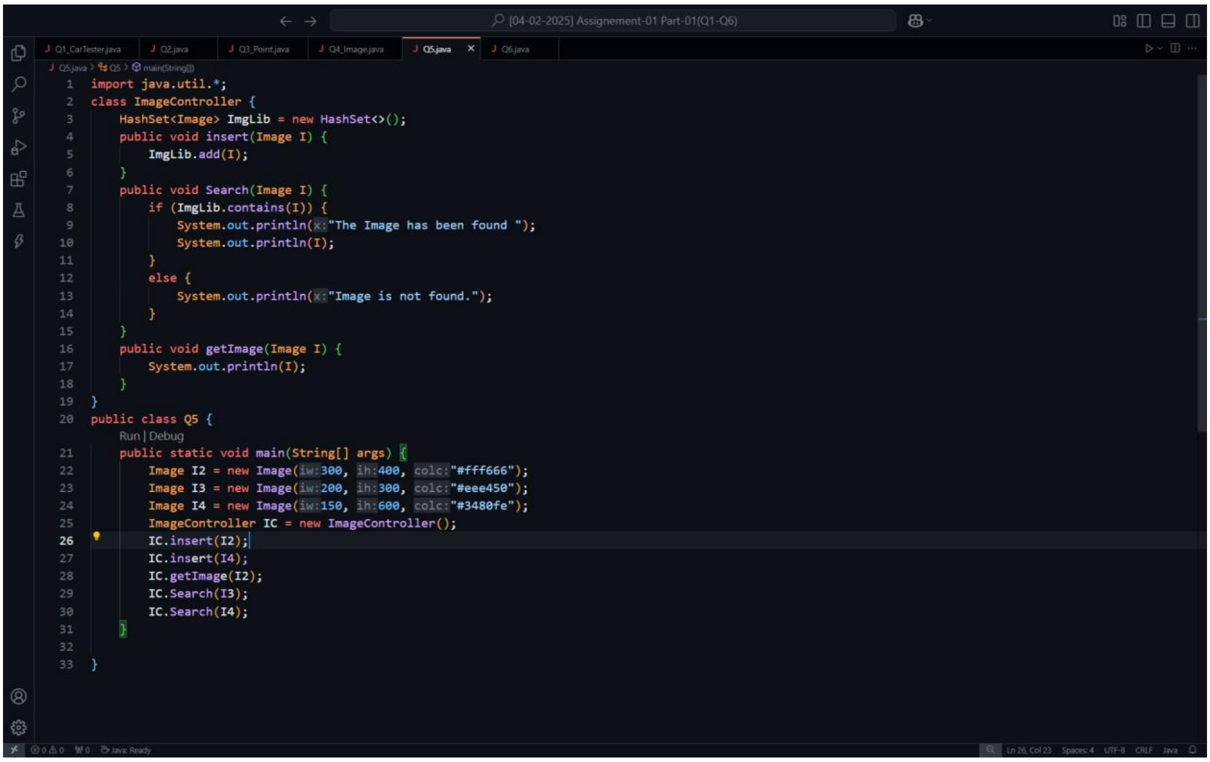


```java
class Image {
    int imgw,imgh; String colc;
    public Image(){
        setimage(iw:1000,ih:500,col:"#dd0078");
    }
    public Image(int iw,int ih,String colc) {
        imgw=iw;    imgh=ih;    this.colc=colc;
    }
    public void setimage(int iw,int ih,String col) {
        imgw=iw;        imgh=ih;        colc=col;
    }
    public int getwi()          {return imgw;}
    public int getih()          {return imgh;}
    public String getcol()      {return colc;}
    public String toString()    {return "Image [Width: "+imgw+", Height: "+imgh+", Color Code: "+colc+"]";}
}
public class Q4_Image {
    Run | Debug
    public static void main(String[] args) {
        Image ig1=new Image(iw:1200,ih:600,colc:"#ff5690");
        System.out.println("Default Image Details: "+ig1);
        Image ig2=new Image();
        System.out.println("Parameterized Image Details: "+ig2);
    }
}
```

```
[Done] exited with code=0 in 0.941 seconds

[Running] cd "a:\Programs\HTML & CSS (from Sems)\4th Semester\CSW-2\[04-02-2025] Assignement-01 Part-01(Q1-Q6)\" && javac Q4_Image.java && java Q4_Image
Default Image Details: Image [Width: 1200, Height: 600, Color Code:■ #ff5690]
Parameterized Image Details: Image [Width: 1000, Height: 500, Color Code:■ #dd0078]

[Done] exited with code=0 in 0.87 seconds
```

Q5. Create an ImageLibrary, which contains a set of image objects (from Q4) and operations such as searching an image, inserting an image, and getting an image. Create an ImageController class to manage the program execution and call the methods to create and manipulate images.

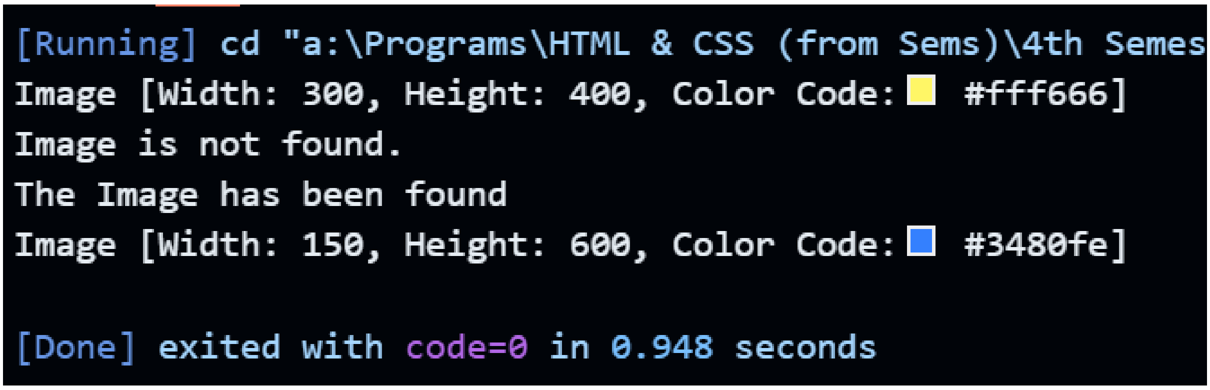
Solution:



[04-02-2025] Assignement-01 Part-01(Q1-Q6)
Q1_CarTester.java
Q2.java
Q3_Point.java
Q4_Image.java
Q5.java
Q6.java
Q5.java > Q5 > main(String[])
Run | Debug
0 0 0 Java: Ready
Ln 26, Col 23 Spaces: 4 UTF-8 CRLF Java

```java
import java.util.*;
class ImageController {
    HashSet<Image> ImgLib = new HashSet<>();
    public void insert(Image I) {
        ImgLib.add(I);
    }
    public void Search(Image I) {
        if (ImgLib.contains(I)) {
            System.out.println(x:"The Image has been found ");
            System.out.println(I);
        }
        else {
            System.out.println(x:"Image is not found.");
        }
    }
    public void getImage(Image I) {
        System.out.println(I);
    }
}
public class Q5 {
    public static void main(String[] args) {
        Image I2 = new Image(iw:300, ih:400, colc:"#fff666");
        Image I3 = new Image(iw:200, ih:300, colc:"#eee450");
        Image I4 = new Image(iw:150, ih:600, colc:"#3480fe");
        ImageController IC = new ImageController();
        IC.insert(I2);
        IC.insert(I4);
        IC.getImage(I2);
        IC.Search(I3);
        IC.Search(I4);
    }

}
```

Output:

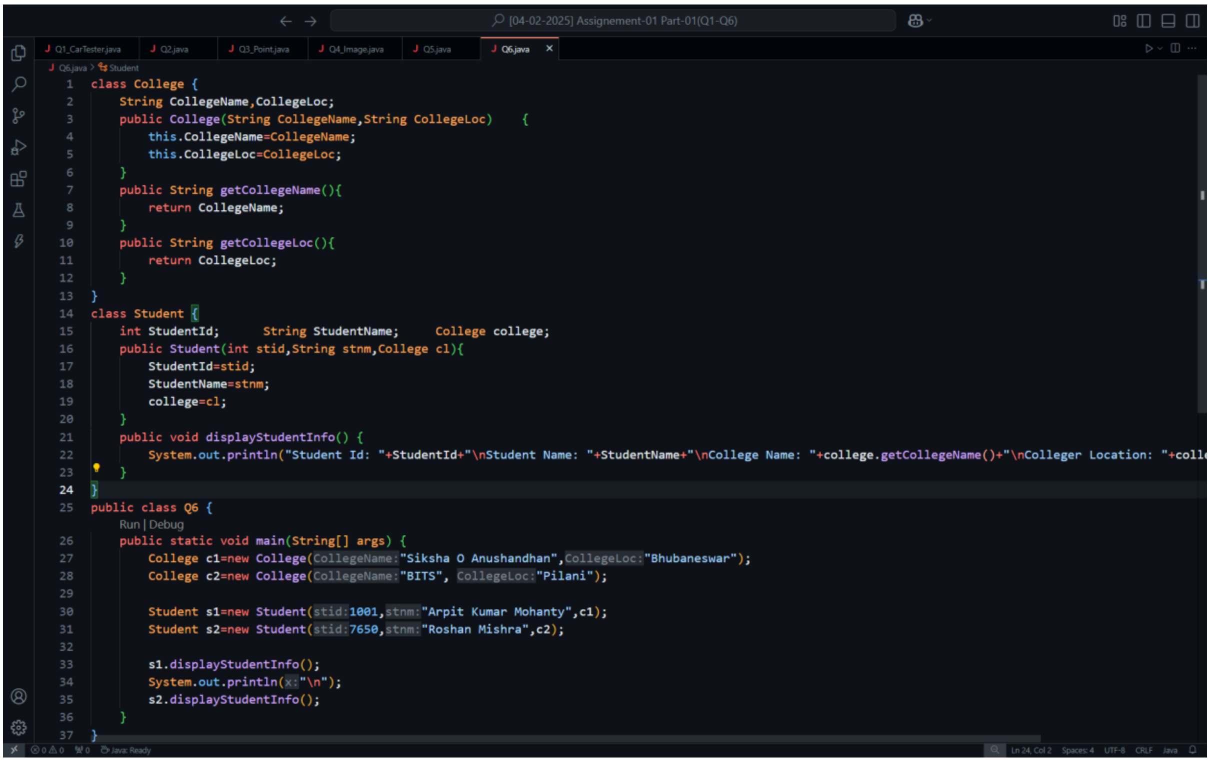


```
[Running] cd "a:\Programs\HTML & CSS (from Sems)\4th Semes
Image [Width: 300, Height: 400, Color Code: 🟨 #fff666]
Image is not found.
The Image has been found
Image [Width: 150, Height: 600, Color Code: 🟦 #3480fe]

[Done] exited with code=0 in 0.948 seconds
```

Q6. Develop a Java-based College Management System to model the relationship between colleges and students. Create a College class with attributes collegeName and collegeLoc, and a Student class with studentId, studentName, and a reference to a College object. Implement a constructor in Student to initialize these attributes and a displayStudentInfo() method to print student and college details. In the MainApp class, instantiate at least two College and Student objects

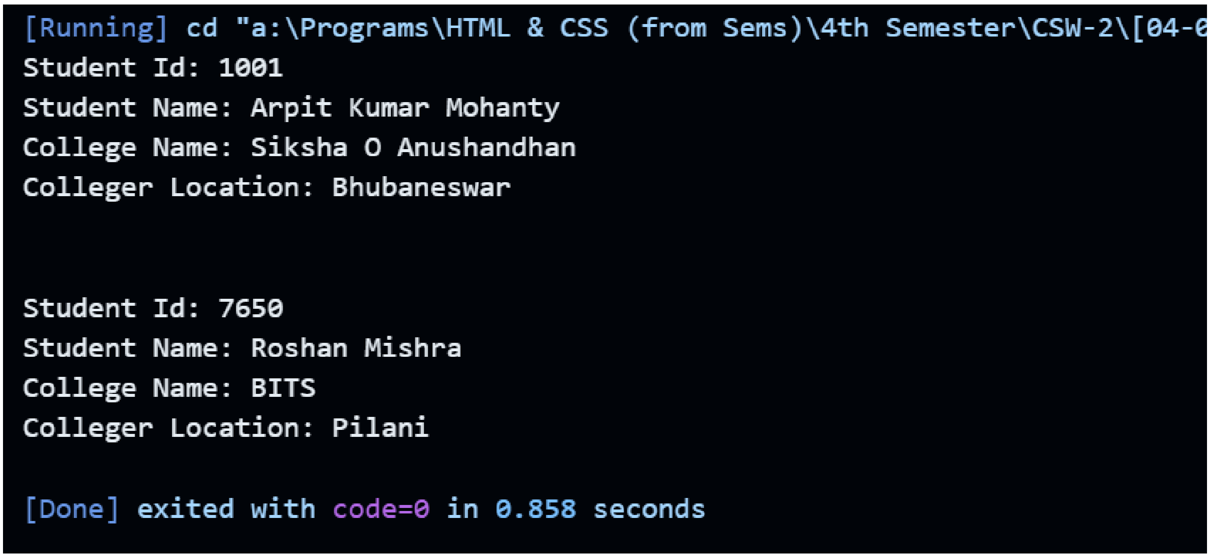
enroll each student in one of the colleges, and display all details using appropriate methods.

Solution:



```java
class College {
    String CollegeName,CollegeLoc;
    public College(String CollegeName,String CollegeLoc)    {
        this.CollegeName=CollegeName;
        this.CollegeLoc=CollegeLoc;
    }
    public String getCollegeName(){
        return CollegeName;
    }
    public String getCollegeLoc(){
        return CollegeLoc;
    }
}
class Student {
    int StudentId;      String StudentName;     College college;
    public Student(int stid,String stnm,College cl){
        StudentId=stid;
        StudentName=stnm;
        college=cl;
    }
    public void displayStudentInfo() {
        System.out.println("Student Id: "+StudentId+"\nStudent Name: "+StudentName+"\nCollege Name: "+college.getCollegeName()+"\nColleger Location: "+colle
    }
}
public class Q6 {
    public static void main(String[] args) {
        College c1=new College("Siksha O Anushandhan","Bhubaneswar");
        College c2=new College("BITS", "Pilani");

        Student s1=new Student(1001,"Arpit Kumar Mohanty",c1);
        Student s2=new Student(7650,"Roshan Mishra",c2);

        s1.displayStudentInfo();
        System.out.println("\n");
        s2.displayStudentInfo();
    }
}
```

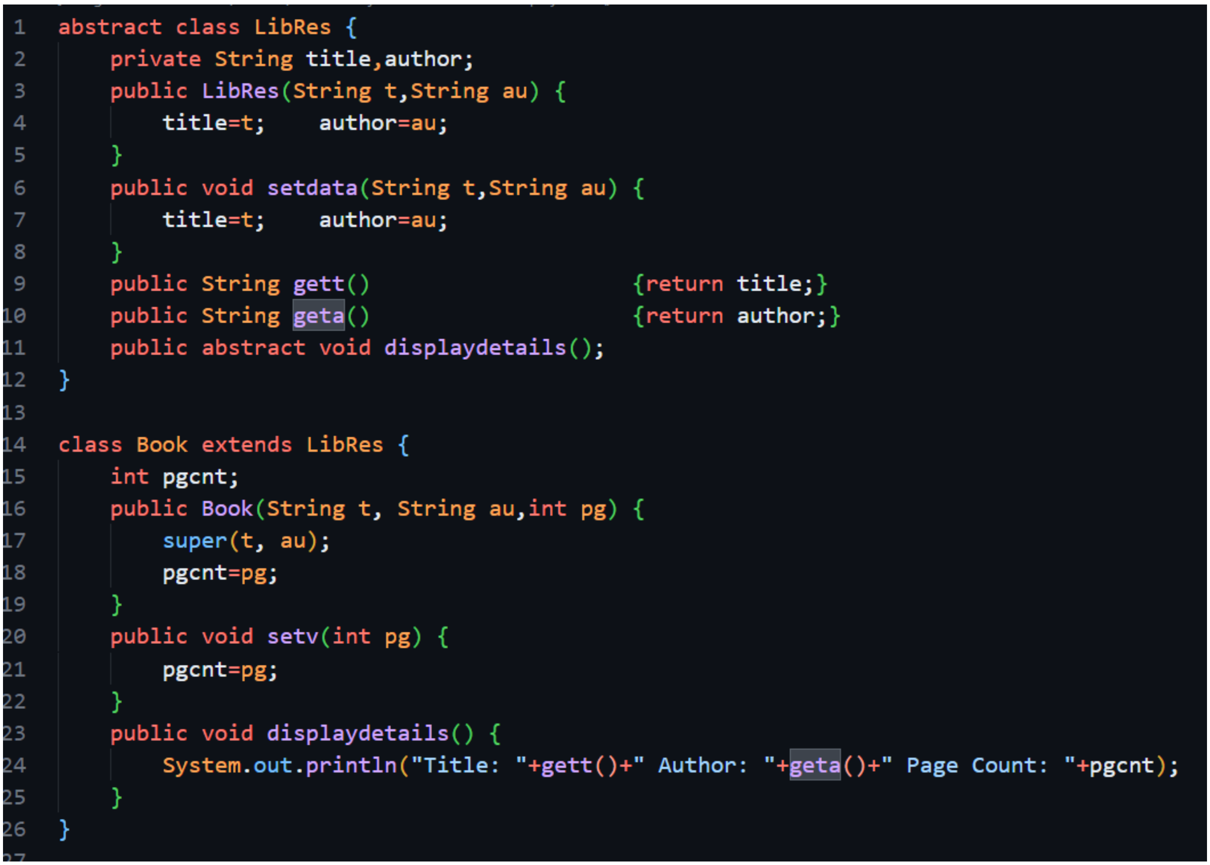
Output:



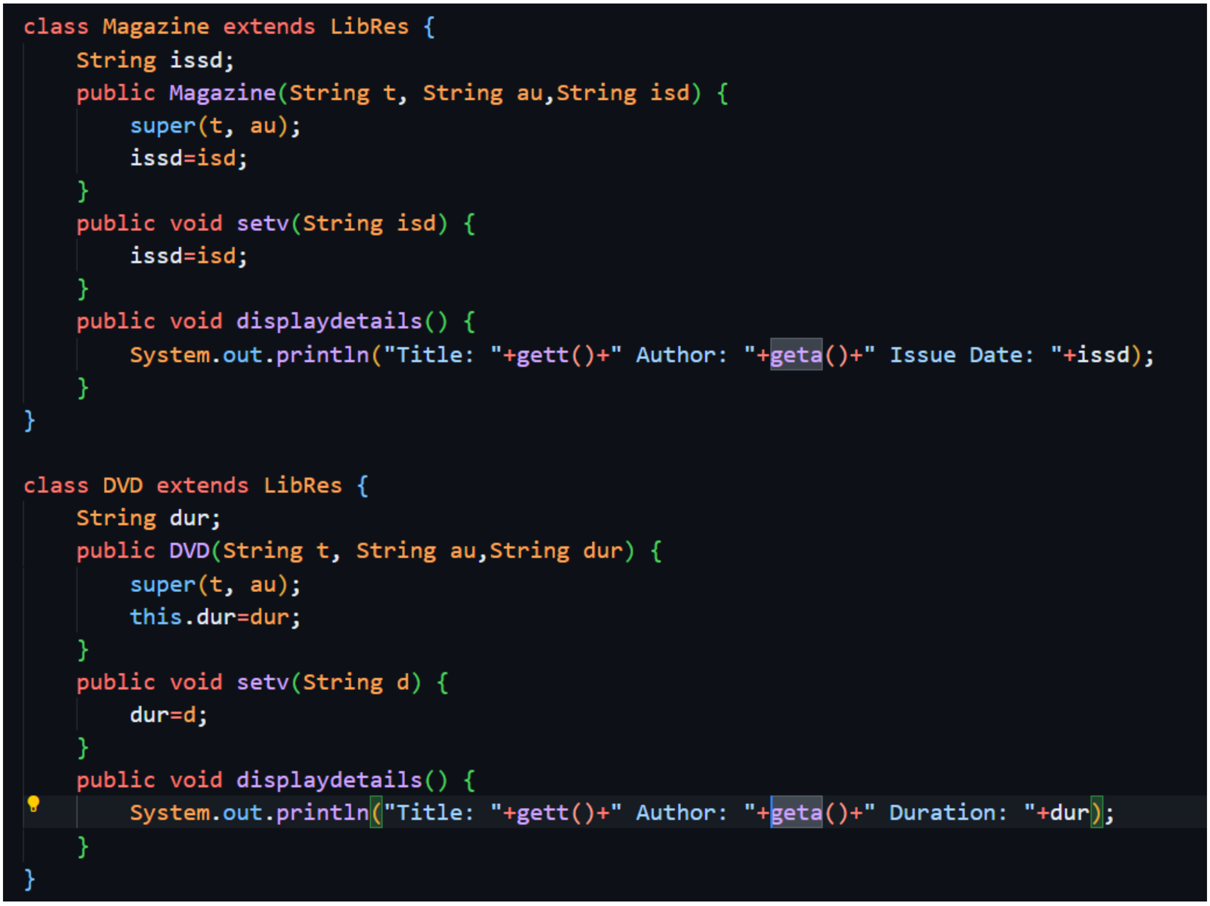
```
[Running] cd "a:\Programs\HTML & CSS (from Sems)\4th Semester\CSW-2\[04-0
Student Id: 1001
Student Name: Arpit Kumar Mohanty
College Name: Siksha O Anushandhan
Colleger Location: Bhubaneswar


Student Id: 7650
Student Name: Roshan Mishra
College Name: BITS
Colleger Location: Pilani

[Done] exited with code=0 in 0.858 seconds
```

Q7. Develop a Java program for a library system using encapsulation, abstraction, and inheritance. Create an abstract LibraryResource class with private attributes (title, author), a constructor, getters, setters, and an abstract displayDetails() method. Extend it into Book, Magazine, and DVD classes, each adding a specific attribute (pageCount, issueDate, duration), along with constructors, getters, setters, and overridden displayDetails() methods. In the LibrarySystem class, instantiate various resources and call displayDetails() to display their information.

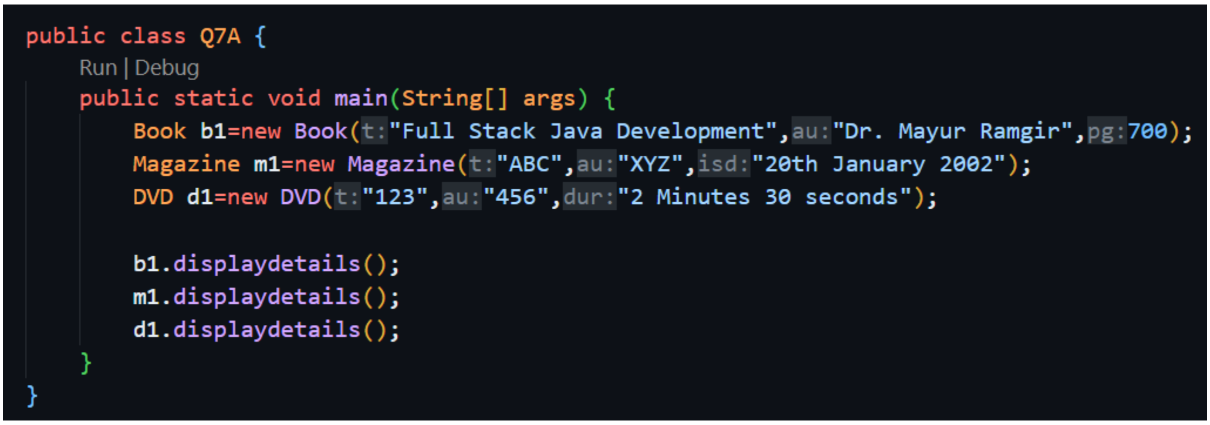
Soution:



```java
abstract class LibRes {
    private String title,author;
    public LibRes(String t,String au) {
        title=t;    author=au;
    }
    public void setdata(String t,String au) {
        title=t;    author=au;
    }
    public String gett()                    {return title;}
    public String geta()                    {return author;}
    public abstract void displaydetails();
}

class Book extends LibRes {
    int pgcnt;
    public Book(String t, String au,int pg) {
        super(t, au);
        pgcnt=pg;
    }
    public void setv(int pg) {
        pgcnt=pg;
    }
    public void displaydetails() {
        System.out.println("Title: "+gett()+" Author: "+geta()+" Page Count: "+pgcnt);
    }
}
```

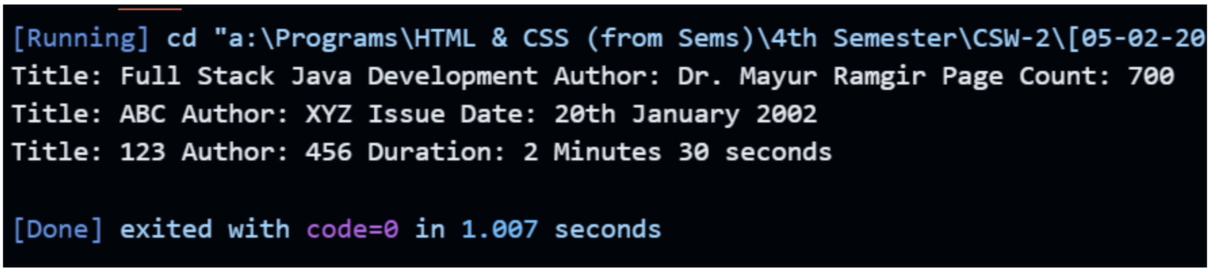


```java
class Magazine extends LibRes {
    String issd;
    public Magazine(String t, String au,String isd) {
        super(t, au);
        issd=isd;
    }
    public void setv(String isd) {
        issd=isd;
    }
    public void displaydetails() {
        System.out.println("Title: "+gett()+" Author: "+geta()+" Issue Date: "+issd);
    }
}

class DVD extends LibRes {
    String dur;
    public DVD(String t, String au,String dur) {
        super(t, au);
        this.dur=dur;
    }
    public void setv(String d) {
        dur=d;
    }
    public void displaydetails() {
        System.out.println("Title: "+gett()+" Author: "+geta()+" Duration: "+dur);
    }
}
```



```
public class Q7A {
    Run | Debug
    public static void main(String[] args) {
        Book b1=new Book(t:"Full Stack Java Development",au:"Dr. Mayur Ramgir",pg:700);
        Magazine m1=new Magazine(t:"ABC",au:"XYZ",isd:"20th January 2002");
        DVD d1=new DVD(t:"123",au:"456",dur:"2 Minutes 30 seconds");

        b1.displaydetails();
        m1.displaydetails();
        d1.displaydetails();
    }
}
```

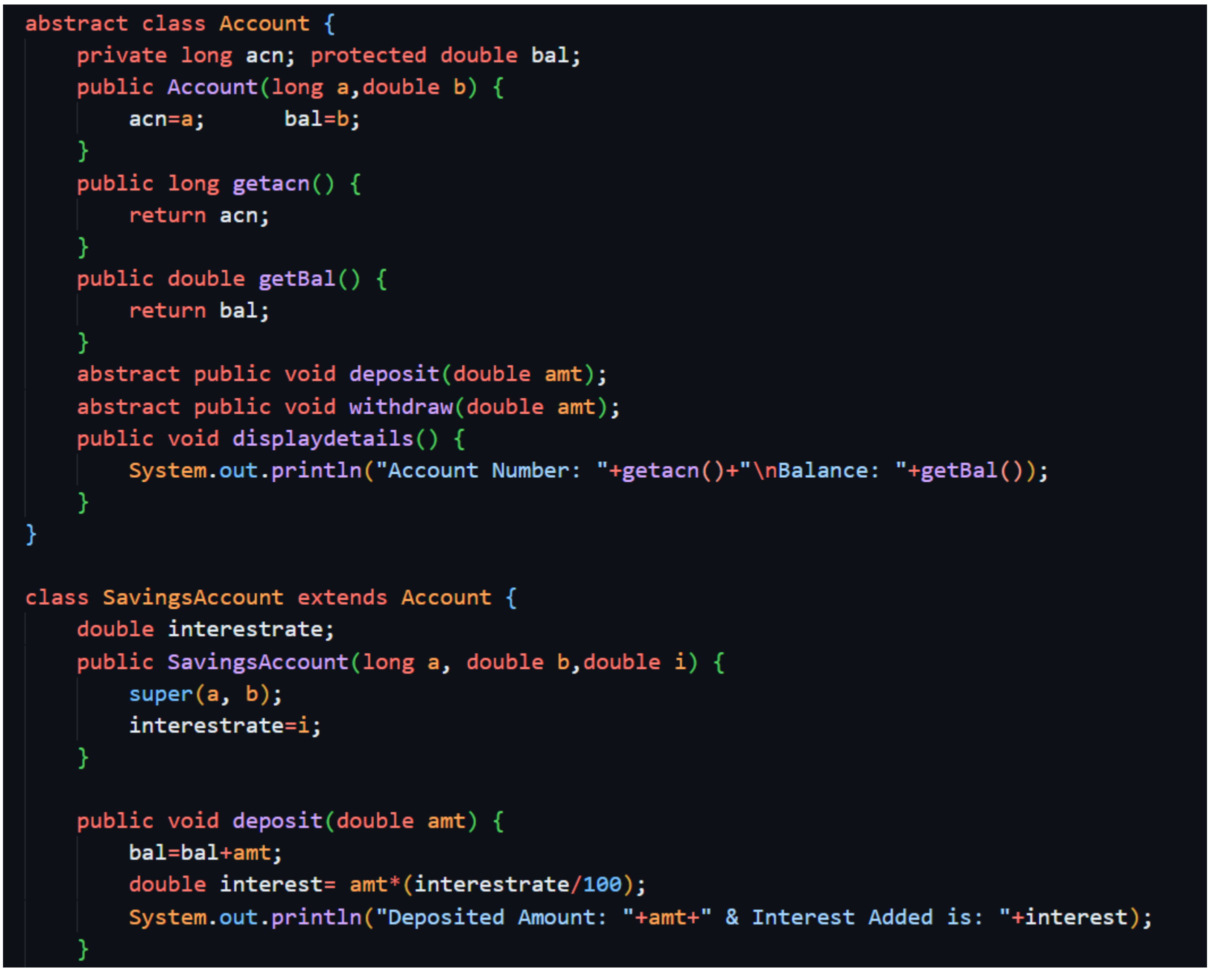
Output:



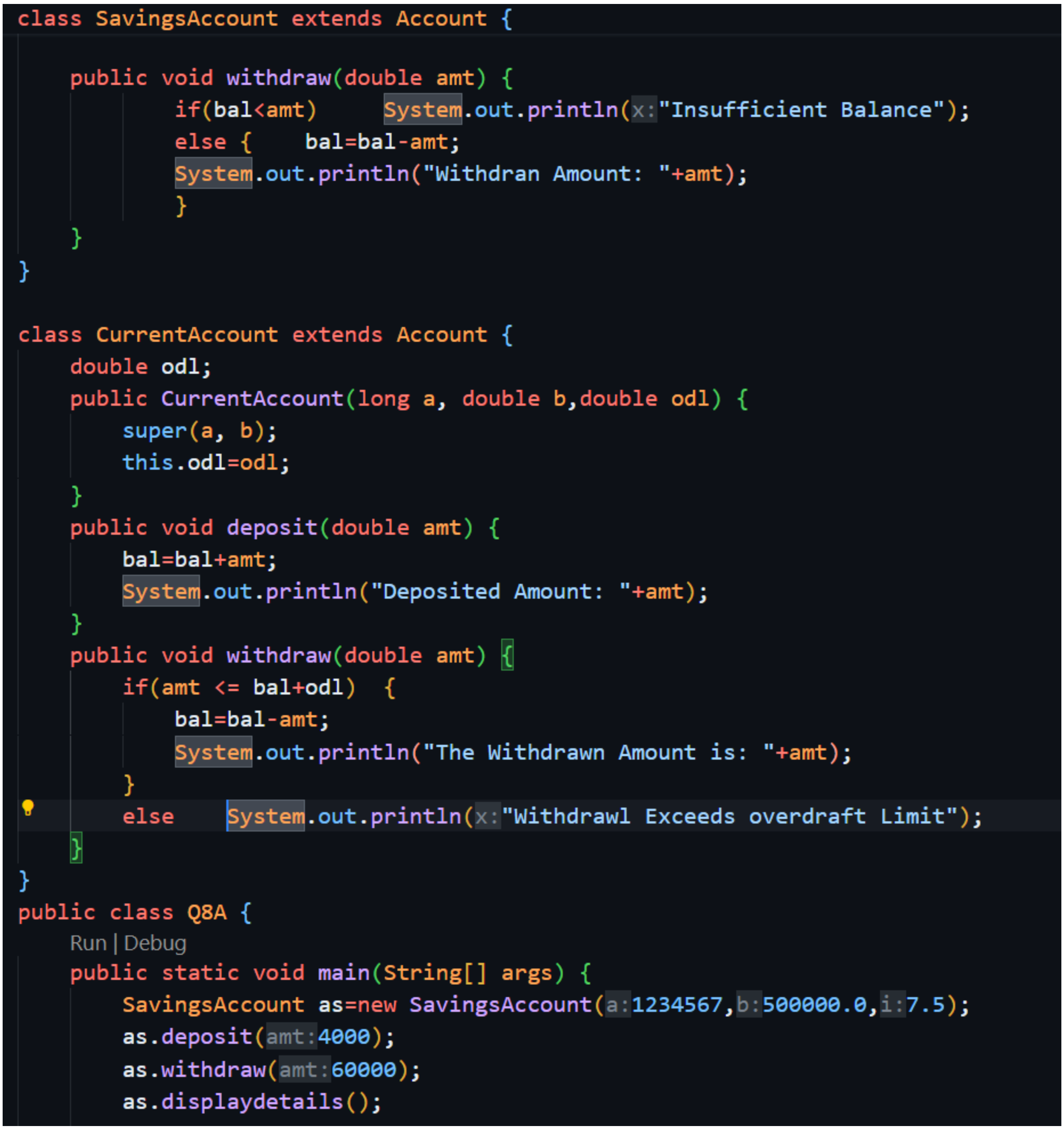
```
[Running] cd "a:\Programs\HTML & CSS (from Sems)\4th Semester\CSW-2\[05-02-20
Title: Full Stack Java Development Author: Dr. Mayur Ramgir Page Count: 700
Title: ABC Author: XYZ Issue Date: 20th January 2002
Title: 123 Author: 456 Duration: 2 Minutes 30 seconds

[Done] exited with code=0 in 1.007 seconds
```

Q8. Develop a Java banking application using polymorphism with three classes: Account, SavingsAccount, and CurrentAccount. The abstract Account class has private attributes (accountNumber, balance) and abstract methods for deposit and withdrawal. SavingsAccount adds an interestRate attribute, overrides deposit to calculate interest, and ensures sufficient balance in withdrawal. CurrentAccount introduces an overdraftLimit and overrides withdrawal to check this limit. In the BankingApp class, instantiate both account types, perform transactions, and display account details to demonstrate polymorphism.

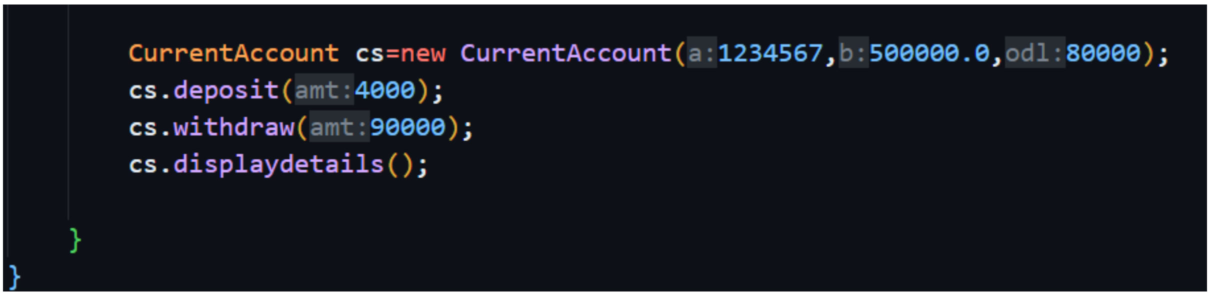
Solution:



```java
abstract class Account {
    private long acn; protected double bal;
    public Account(long a,double b) {
        acn=a;      bal=b;
    }
    public long getacn() {
        return acn;
    }
    public double getBal() {
        return bal;
    }
    abstract public void deposit(double amt);
    abstract public void withdraw(double amt);
    public void displaydetails() {
        System.out.println("Account Number: "+getacn()+"\nBalance: "+getBal());
    }
}

class SavingsAccount extends Account {
    double interestrate;
    public SavingsAccount(long a, double b,double i) {
        super(a, b);
        interestrate=i;
    }

    public void deposit(double amt) {
        bal=bal+amt;
        double interest= amt*(interestrate/100);
        System.out.println("Deposited Amount: "+amt+" & Interest Added is: "+interest);
    }
```

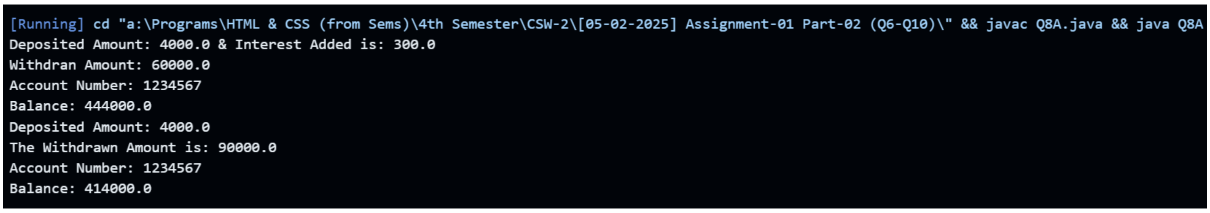


```
class SavingsAccount extends Account {

    public void withdraw(double amt) {
            if(bal<amt)      System.out.println(x:"Insufficient Balance");
            else {     bal=bal-amt;
            System.out.println("Withdran Amount: "+amt);
            }
    }
}

class CurrentAccount extends Account {
    double odl;
    public CurrentAccount(long a, double b,double odl) {
        super(a, b);
        this.odl=odl;
    }
    public void deposit(double amt) {
        bal=bal+amt;
        System.out.println("Deposited Amount: "+amt);
    }
    public void withdraw(double amt) {
        if(amt <= bal+odl)  {
            bal=bal-amt;
            System.out.println("The Withdrawn Amount is: "+amt);
        }
        else    System.out.println(x:"Withdrawl Exceeds overdraft Limit");
    }
}
public class Q8A {
    Run | Debug
    public static void main(String[] args) {
        SavingsAccount as=new SavingsAccount(a:1234567,b:500000.0,i:7.5);
        as.deposit(amt:4000);
        as.withdraw(amt:60000);
        as.displaydetails();
```



```
        CurrentAccount cs=new CurrentAccount(a:1234567,b:500000.0,odl:80000);
        cs.deposit(amt:4000);
        cs.withdraw(amt:90000);
        cs.displaydetails();

    }
}
```

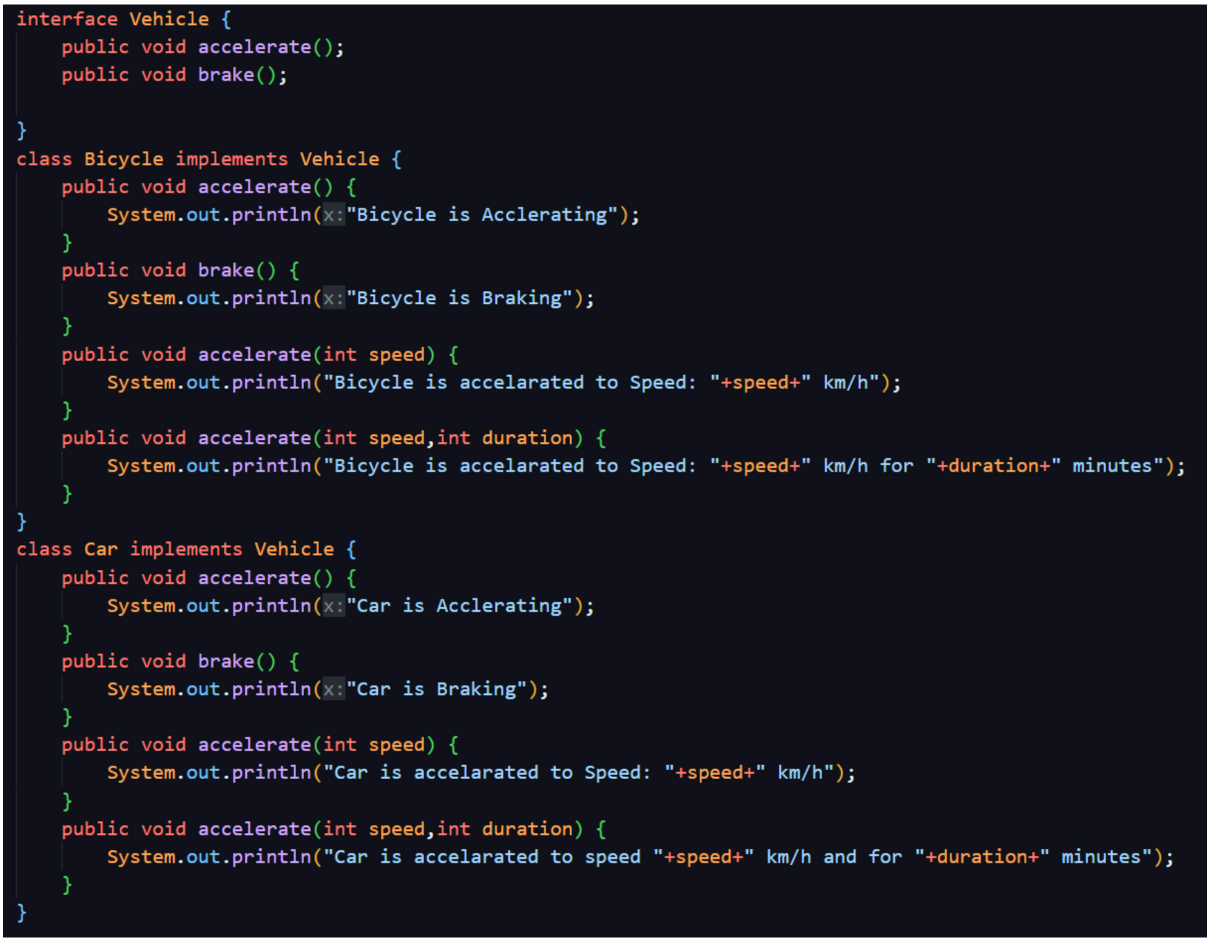
Output:



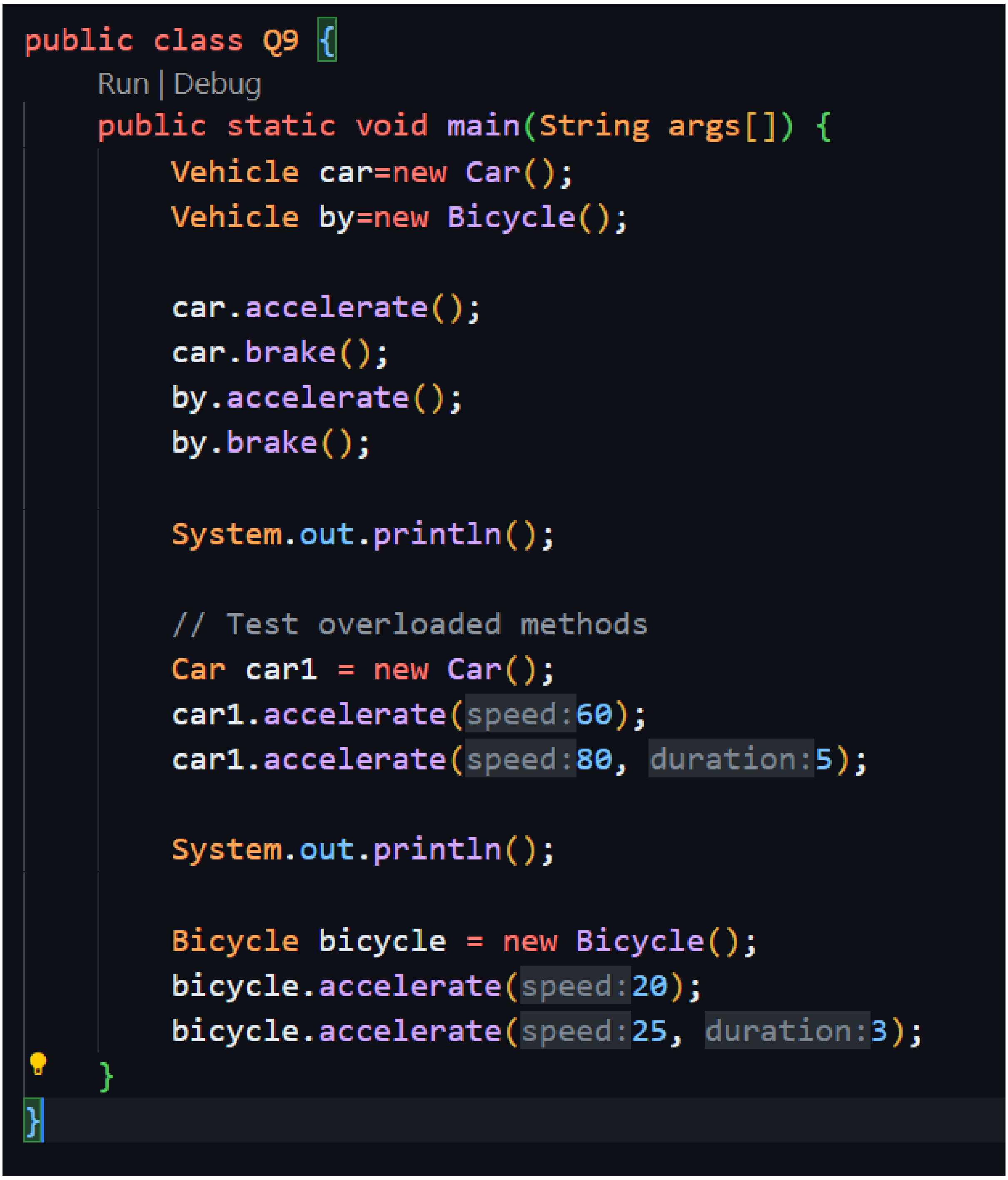
```
[Running] cd "a:\Programs\HTML & CSS (from Sems)\4th Semester\CSW-2\[05-02-2025] Assignment-01 Part-02 (Q6-Q10)\" && javac Q8A.java && java Q8A
Deposited Amount: 4000.0 & Interest Added is: 300.0
Withdran Amount: 60000.0
Account Number: 1234567
Balance: 444000.0
Deposited Amount: 4000.0
The Withdrawn Amount is: 90000.0
Account Number: 1234567
Balance: 414000.0
```

Q9. Write a Java program demonstrating interfaces, method overriding, and method overloading. Define a Vehicle interface with abstract methods accelerate() and brake(). Implement Car and Bicycle classes that override these methods with specific messages for acceleration and braking. Introduce method overloading in both classes by defining multiple accelerate() methods with different parameters (e.g., speed, duration). In the VehicleApp class, instantiate Car and Bicycle objects, test overridden methods, and invoke overloaded accelerate() methods to showcase polymorphism.

Solution:

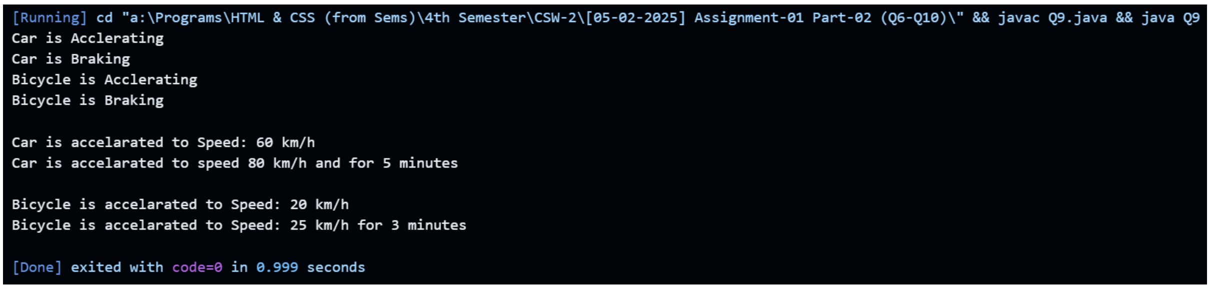


```java
interface Vehicle {
    public void accelerate();
    public void brake();

}
class Bicycle implements Vehicle {
    public void accelerate() {
        System.out.println(x:"Bicycle is Acclerating");
    }
    public void brake() {
        System.out.println(x:"Bicycle is Braking");
    }
    public void accelerate(int speed) {
        System.out.println("Bicycle is accelarated to Speed: "+speed+" km/h");
    }
    public void accelerate(int speed,int duration) {
        System.out.println("Bicycle is accelarated to Speed: "+speed+" km/h for "+duration+" minutes");
    }
}
class Car implements Vehicle {
    public void accelerate() {
        System.out.println(x:"Car is Acclerating");
    }
    public void brake() {
        System.out.println(x:"Car is Braking");
    }
    public void accelerate(int speed) {
        System.out.println("Car is accelarated to Speed: "+speed+" km/h");
    }
    public void accelerate(int speed,int duration) {
        System.out.println("Car is accelarated to speed "+speed+" km/h and for "+duration+" minutes");
    }
}
```



```java
public class Q9 {
    Run | Debug
    public static void main(String args[]) {
        Vehicle car=new Car();
        Vehicle by=new Bicycle();

        car.accelerate();
        car.brake();
        by.accelerate();
        by.brake();

        System.out.println();

        // Test overloaded methods
        Car car1 = new Car();
        car1.accelerate(speed:60);
        car1.accelerate(speed:80, duration:5);

        System.out.println();

        Bicycle bicycle = new Bicycle();
        bicycle.accelerate(speed:20);
        bicycle.accelerate(speed:25, duration:3);
    }
}
```

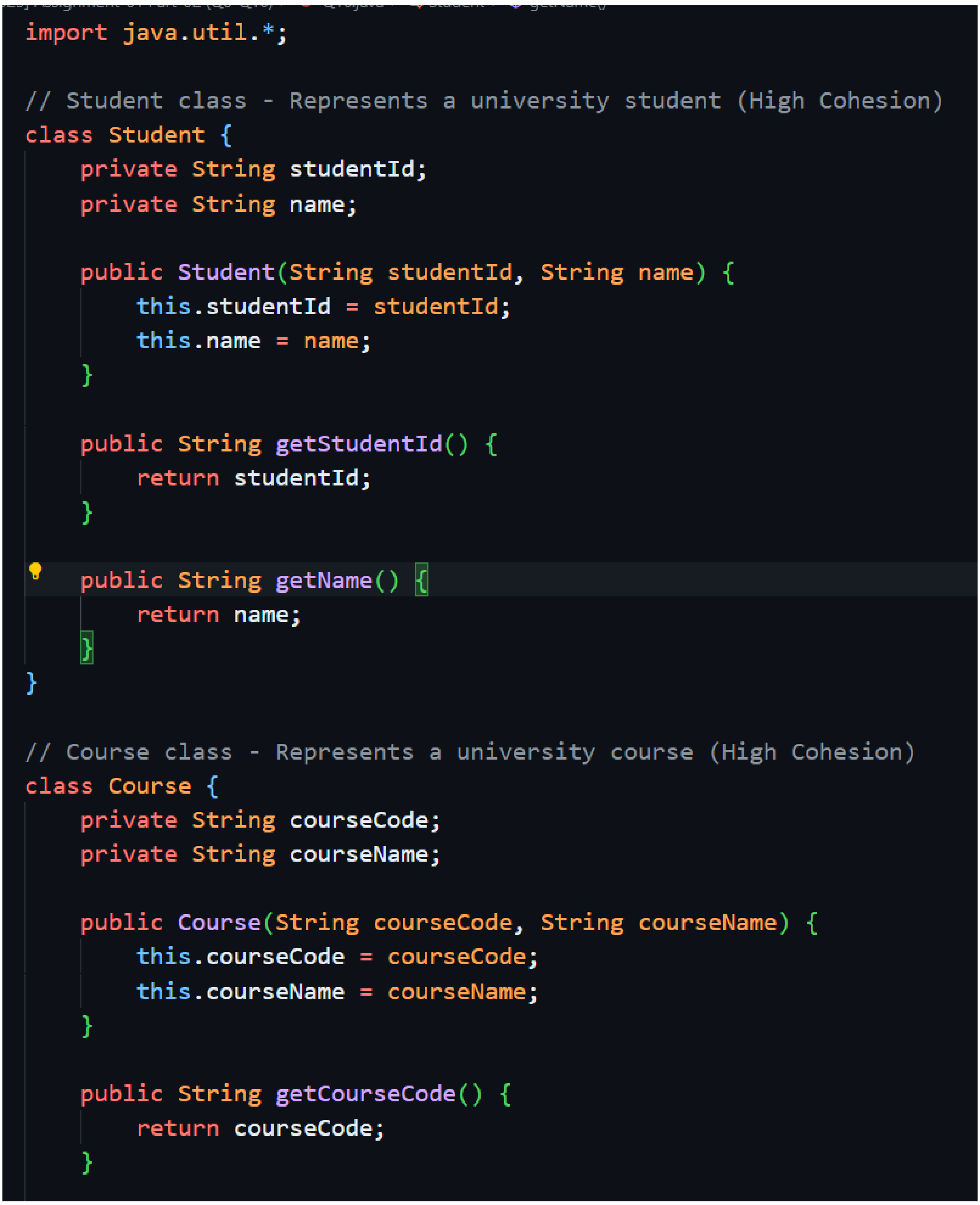
Output:



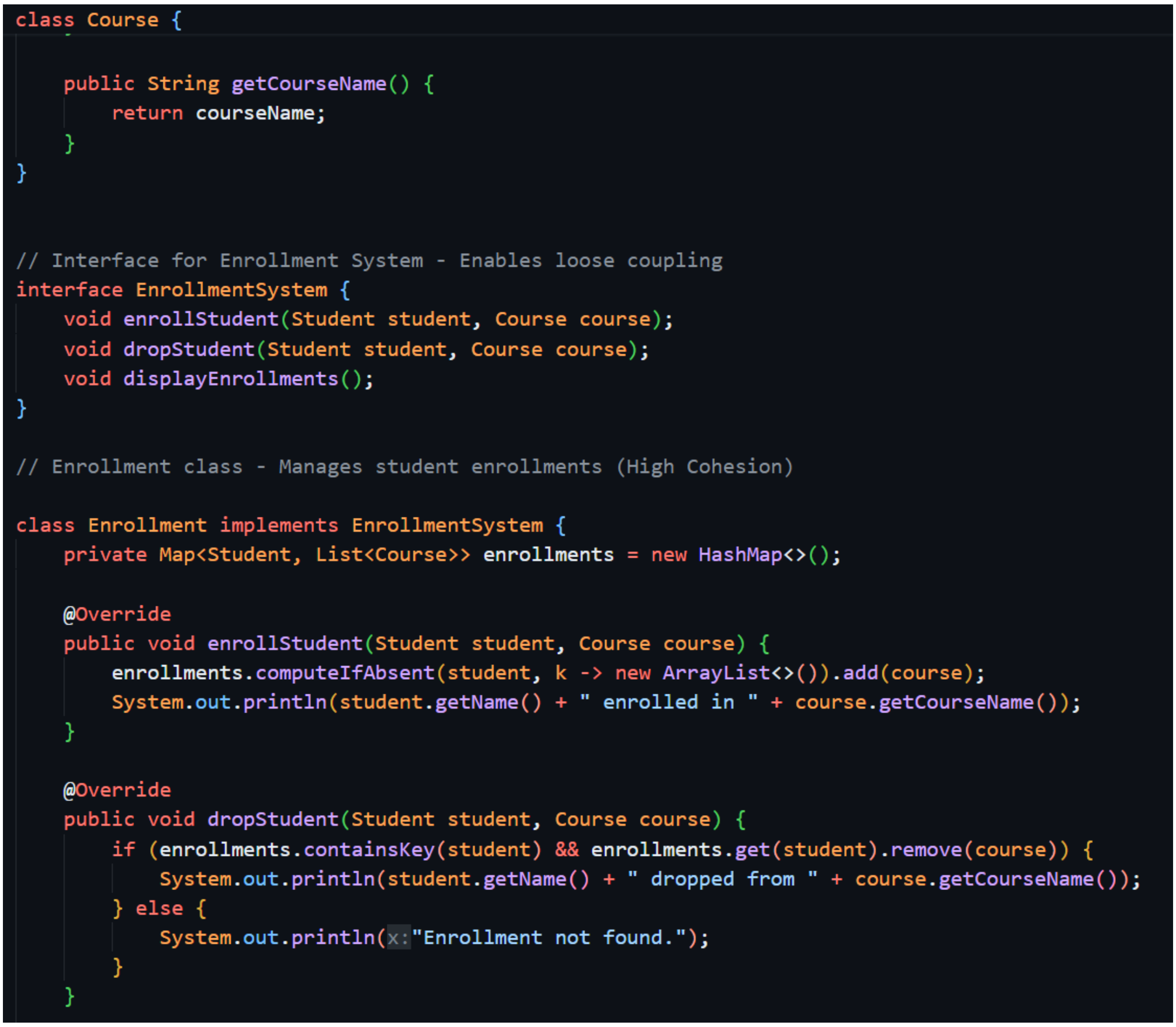
```
[Running] cd "a:\Programs\HTML & CSS (from Sems)\4th Semester\CSW-2\[05-02-2025] Assignment-01 Part-02 (Q6-Q10)\" && javac Q9.java && java Q9
Car is Acclerating
Car is Braking
Bicycle is Acclerating
Bicycle is Braking

Car is accelarated to Speed: 60 km/h
Car is accelarated to speed 80 km/h and for 5 minutes

Bicycle is accelarated to Speed: 20 km/h
Bicycle is accelarated to Speed: 25 km/h for 3 minutes

[Done] exited with code=0 in 0.999 seconds
```

Q10. Design a Java program for university student enrollment, ensuring loose coupling and high cohesion. Create Student and Course classes, and an Enrollment class that interacts with them through an EnrollmentSystem interface. Implement methods for enrolling and dropping students from courses, and displaying enrollment details. In the MainEnrollApp class, demonstrate functionality by managing student enrollments. Use comments to explain how the design maintains loose coupling (by relying on interfaces) and high cohesion (by keeping related functionalities within appropriate classes).

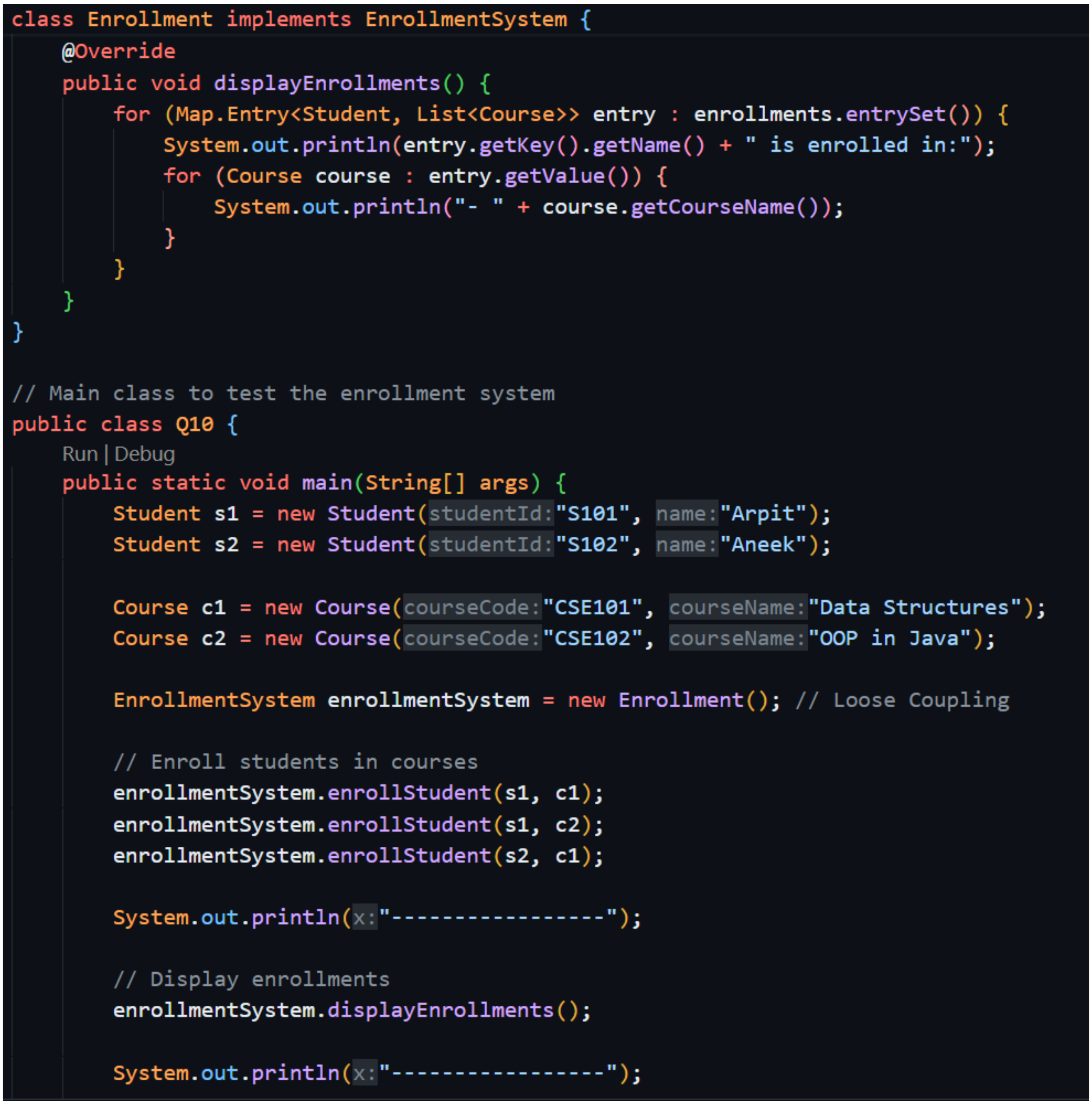
Solution:



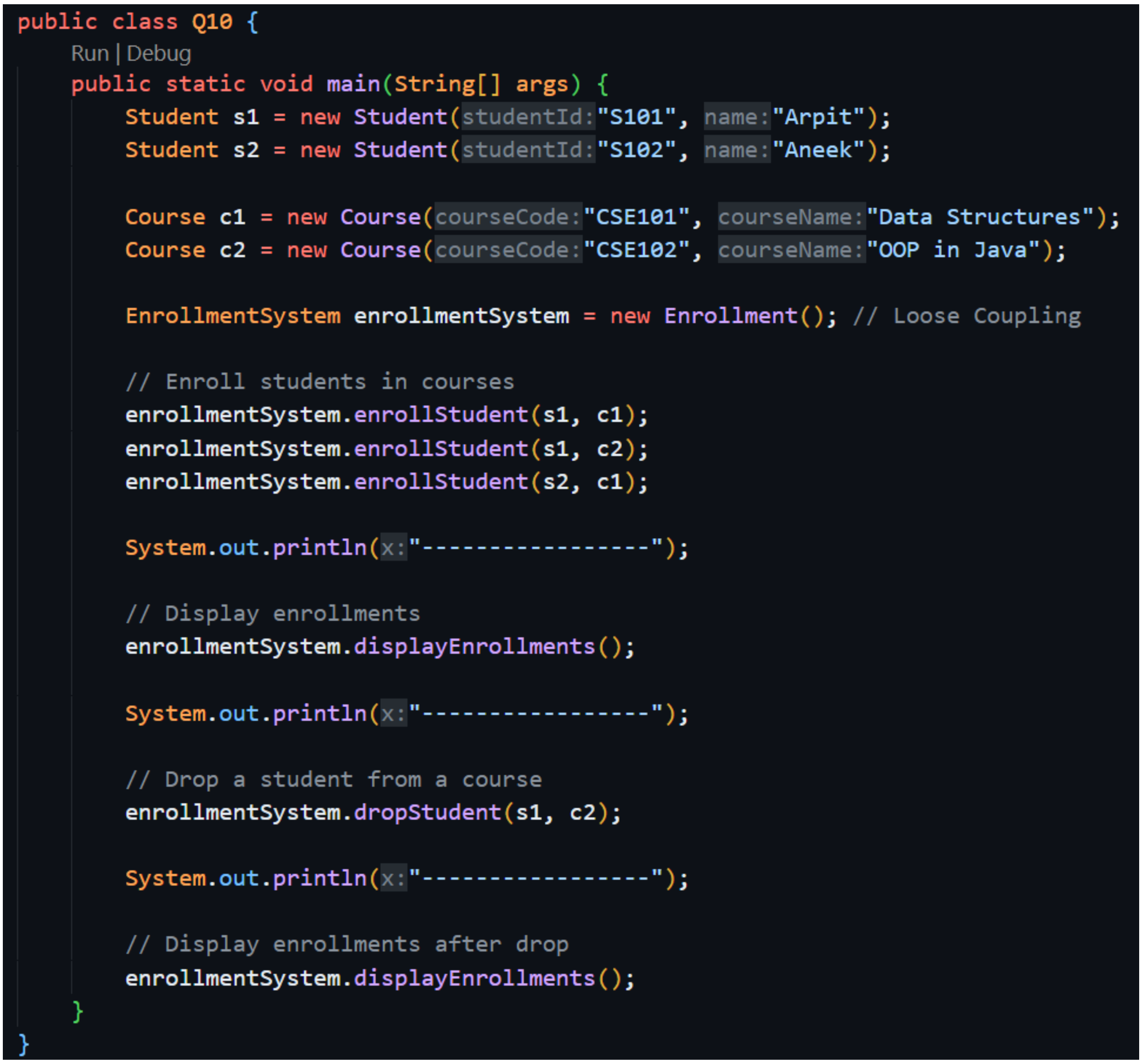
```java
import java.util.*;

// Student class - Represents a university student (High Cohesion)
class Student {
    private String studentId;
    private String name;

    public Student(String studentId, String name) {
        this.studentId = studentId;
        this.name = name;
    }

    public String getStudentId() {
        return studentId;
    }

    public String getName() {
        return name;
    }
}

// Course class - Represents a university course (High Cohesion)
class Course {
    private String courseCode;
    private String courseName;

    public Course(String courseCode, String courseName) {
        this.courseCode = courseCode;
        this.courseName = courseName;
    }

    public String getCourseCode() {
        return courseCode;
    }
```



```
class Course {

    public String getCourseName() {
        return courseName;
    }
}


// Interface for Enrollment System - Enables loose coupling
interface EnrollmentSystem {
    void enrollStudent(Student student, Course course);
    void dropStudent(Student student, Course course);
    void displayEnrollments();
}

// Enrollment class - Manages student enrollments (High Cohesion)

class Enrollment implements EnrollmentSystem {
    private Map<Student, List<Course>> enrollments = new HashMap<>();

    @Override
    public void enrollStudent(Student student, Course course) {
        enrollments.computeIfAbsent(student, k -> new ArrayList<>()).add(course);
        System.out.println(student.getName() + " enrolled in " + course.getCourseName());
    }

    @Override
    public void dropStudent(Student student, Course course) {
        if (enrollments.containsKey(student) && enrollments.get(student).remove(course)) {
            System.out.println(student.getName() + " dropped from " + course.getCourseName());
        } else {
            System.out.println(x:"Enrollment not found.");
        }
    }
```

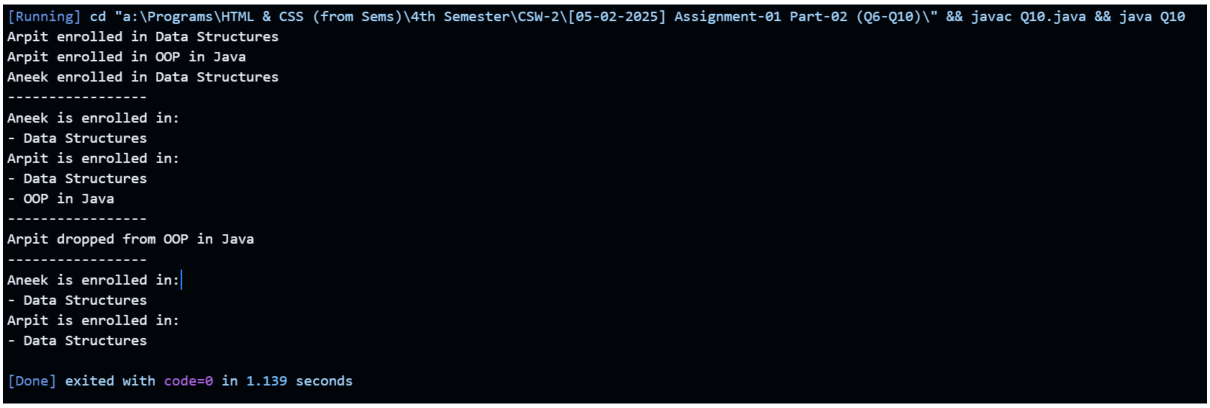


```
class Enrollment implements EnrollmentSystem {
    @Override
    public void displayEnrollments() {
        for (Map.Entry<Student, List<Course>> entry : enrollments.entrySet()) {
            System.out.println(entry.getKey().getName() + " is enrolled in:");
            for (Course course : entry.getValue()) {
                System.out.println("- " + course.getCourseName());
            }
        }
    }
}

// Main class to test the enrollment system
public class Q10 {
    Run | Debug
    public static void main(String[] args) {
        Student s1 = new Student(studentId:"S101", name:"Arpit");
        Student s2 = new Student(studentId:"S102", name:"Aneek");

        Course c1 = new Course(courseCode:"CSE101", courseName:"Data Structures");
        Course c2 = new Course(courseCode:"CSE102", courseName:"OOP in Java");

        EnrollmentSystem enrollmentSystem = new Enrollment(); // Loose Coupling

        // Enroll students in courses
        enrollmentSystem.enrollStudent(s1, c1);
        enrollmentSystem.enrollStudent(s1, c2);
        enrollmentSystem.enrollStudent(s2, c1);

        System.out.println(x:"-----------------");

        // Display enrollments
        enrollmentSystem.displayEnrollments();

        System.out.println(x:"-----------------");
```



```java
public class Q10 {
    Run | Debug
    public static void main(String[] args) {
        Student s1 = new Student(studentId:"S101", name:"Arpit");
        Student s2 = new Student(studentId:"S102", name:"Aneek");

        Course c1 = new Course(courseCode:"CSE101", courseName:"Data Structures");
        Course c2 = new Course(courseCode:"CSE102", courseName:"OOP in Java");

        EnrollmentSystem enrollmentSystem = new Enrollment(); // Loose Coupling

        // Enroll students in courses
        enrollmentSystem.enrollStudent(s1, c1);
        enrollmentSystem.enrollStudent(s1, c2);
        enrollmentSystem.enrollStudent(s2, c1);

        System.out.println(x:"-----------------");

        // Display enrollments
        enrollmentSystem.displayEnrollments();

        System.out.println(x:"-----------------");

        // Drop a student from a course
        enrollmentSystem.dropStudent(s1, c2);

        System.out.println(x:"-----------------");

        // Display enrollments after drop
        enrollmentSystem.displayEnrollments();
    }
}
```

Output:



```
[Running] cd "a:\Programs\HTML & CSS (from Sems)\4th Semester\CSW-2\[05-02-2025] Assignment-01 Part-02 (Q6-Q10)\" && javac Q10.java && java Q10
Arpit enrolled in Data Structures
Arpit enrolled in OOP in Java
Aneek enrolled in Data Structures
-----------------
Aneek is enrolled in:
- Data Structures
Arpit is enrolled in:
- Data Structures
- OOP in Java
-----------------
Arpit dropped from OOP in Java
-----------------
Aneek is enrolled in:
- Data Structures
Arpit is enrolled in:
- Data Structures

[Done] exited with code=0 in 1.139 seconds
```