Lab Report

Course Tittle: Object Oriented Programming Lab

Course Code: CSE 215

Experiment No: 01 - 10

Experiment Name: Object Oriented Programming in Java

Submitted To:

Name: Mst. Umme Ayman

Designation: Lecturer

Department of CSE

Daffodil International University

Submitted By

Name: Md Raduan Ahamed

ID: 0242220005101839

Section: 63_O

Department of CSE

Daffodil International University

Submission Date: 14-12-2023

INDEX

Lab Report

Course Tittle: Object Oriented Programming Lab

Course Code: CSE 215

Experiment No: 01

Experiment Name: Implementation of Java class, object, construction, method overriding, method overloading, this keyword, super keyword, Inheritance

Submitted To:

Name: Mst. Umme Ayman

Designation: Lecturer

Department of CSE

Daffodil International University

Submitted By

Name: Md Raduan Ahamed

ID: 0242220005101839

Section: 63_O

Department of CSE

Daffodil International University

Submission Date: 29-08-2023

```java
import java.util.Scanner;
public class Shape {
    double a;
    double b;
    double c;
    String color = "Default";
    void displayVariables(){
        System.out.print("Three sides of the triangle are: ");
        System.out.println(a + " " + b + " " + c);
    }
}
class Circle extends Shape{
    double pi = 3.1416;
    String color = "Red";
    Circle(double a){
        this.a = a;
    }
    //Constructor Overloading
    Circle(double r, String color) {
        a = r;
        super.color = color; // super key-word
    }
    void display(){
        System.out.println("Radius of the circle is: " + a);
        System.out.println("Circumference is: " + (2.0*pi*a));
        System.out.println("Area is: " + (pi*a*a));
        System.out.println("Original color is " + color); // Red
        System.out.println("Re-painted color is " + super.color); // input
color
    }
}
class Triangle extends Shape{
    double height;
    @Override
    void displayVariables(){
        System.out.print("Three sides of the triangle are: ");
        System.out.println(a + " " + b + " " + c);
        System.out.println("Height: " + height);
    }
}

import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        System.out.println("Hello! Welcome to our Lab Project. " +
                "What shape do you want to work with?\n 1. Triangle" +
                "\n 2. Circle");
        Scanner sc = new Scanner(System.in);
        Scanner sc2 = new Scanner(System.in);
        int choice = sc.nextInt();
        if (choice == 2) {
            System.out.print("Enter the radius you want: ");
            double radius = sc.nextDouble();
            System.out.println("Color of your circle? ");
            String color = sc2.nextLine();
            // overloading
            Circle obj_1 = new Circle(radius);
            Circle obj_2 = new Circle(radius, color);
            obj_2.display();
        } else {
            System.out.print("Enter three sides of the triangle: ");
            Shape obj_1 = new Shape();
            obj_1.a = sc.nextDouble();
            obj_1.b = sc.nextDouble();
            obj_1.c = sc.nextDouble();
            System.out.print("Enter three sides and also the height: ");
            Triangle obj_2 = new Triangle();
            obj_2.a = sc.nextDouble();
            obj_2.b = sc.nextDouble();
            obj_2.c = sc.nextDouble();
            obj_2.height = sc.nextDouble();
            //override
            obj_1.displayVariables();
            obj_2.displayVariables();
        }
    }
}
```

Output:

```
Hello! Welcome to our Lab Project. What shape do you want to work with?

 1. Triangle

 2. Circle

2

Enter the radius you want: 5

Color of your circle?

red

Radius of the circle is: 5.0

Circumference is: 31.416

Area is: 78.54

Original color is Red

Re-painted color is red
```

Lab Report

Course Tittle: Object Oriented Programming

Course Code: CSE 215

Experiment No: 02

Experiment Name: Usages of Super Keyword in terms of Inheritance

Submitted To:

Name: Mst. Umme Ayman

Designation: Lecturer

Department of CSE

Daffodil International University

Submitted By

Name: Md Raduan Ahamed

ID: 0242220005101839

Section: 63_O

Department of CSE

Daffodil International University

Submission Date: 05-09-2023

Problem: 01

```java
package Student_Employee;

public class Person {

    String name;
    int id;

    Person(String name, int id) {

        this.name = name;
        this.id = id;

    }

    void displayInfo() {

        System.out.println("Name: " + name);
        System.out.println("ID: " + id);

    }

}

public class Student extends Person {

    String qualification;

    Student(String name, int id, String qualification) {

        super(name, id);
        this.qualification = qualification;

    }

    @Override
    void displayInfo() {

        super.displayInfo();
        System.out.println("Qualification: " + qualification);

    }

}

public class Employee extends Person {

    String email;
    String phoneNumber;

    Employee(String name, int id, String email, String phoneNumber) {

        super(name, id);
        this.email = email;
        this.phoneNumber = phoneNumber;

    }
```

```
  @Override
  void displayInfo() {
    super.displayInfo();
    System.out.println("Email: " + email);
    System.out.println("Phone Number: " + phoneNumber);
  }
}

public class Main {
  public static void main(String[] args) {

    Student student = new Student("Md Raduan Ahamed", 0242220005101839, "Bachelor's in
Computer Science");

    Employee employee = new Employee("Mr Abdur Rahman Rohan", 2001250001,
"abdrohan0001@gamil.com", "01600000254");

    System.out.println("Student Information:");
    student.displayInfo();

    System.out.println("\nEmployee Information:");
    employee.displayInfo();
  }
}
```

Output:

```
Student Information:
Name: Md Raduan Ahamed
ID: 0242220005101839
Qualification: Bachelor's in Computer Science

Employee Information:
Name: Mr Abdur Rahman Rohan
ID: 2001250001
Email: abdrohan0001@gamil.com
Phone Number: 01600000254
```

Problem: 02

```
package Animal_Sound;

public class Animal {

    String sound;

    Animal(String sound) {

        this.sound = sound;

    }

    void makeSound() {

        System.out.println("Animal sound: " + sound);

    }

}

public class Dog extends Animal {

    Dog(String sound) {

        super(sound);

    }

    void makeSound() {

        super.makeSound();
        System.out.println("Dog sound: " + sound);

    }

}

public class Main {

    public static void main(String[] args) {

        Dog dog = new Dog("Woof Woof!!!");

        dog.makeSound();

    }

}
```

Output:

```
Animal sound: Woof Woof!!!

Dog sound: Woof Woof!!!
```

Problem: 03

```
Package Vehicle_Car;

Public class Vehicle {

    String brand;

    Vehicle(String brand) {

        this.brand = brand;

    }

    void display() {

        System.out.println("This is a " + brand + " vehicle.");

    }

}

public class Car extends Vehicle {

    int year;

    Car(String brand, int year) {

        super(brand);
        this.year = year;

    }

    void display() {

        super.display();
        System.out.println("It was manufactured in " + year);

    }

}

public class Main {

    public static void main(String[] args) {

        Car myCar = new Car("Range Rover",1970);
        myCar.display();

    }

}
```

Output:

```
This is a Range Rover vehicle.

It was manufactured in 1970
```

Problem: 04

```
Package Mobile_Company;

public class Mobile {

    String brand;
    String color;

    Mobile(String brand, String color) {

        this.brand = brand;
        this.color = color;

    }

    void displaySpecs() {

        System.out.println("Brand: " + brand);
        System.out.println("Color: " + color);

    }

}

public class Smartphone extends Mobile {

    int camera;
    double price;

    Smartphone(String brand, String color, int camera, double price) {

        super(brand, color);
        this.camera = camera;
        this.price = price;

    }

    @Override

    void displaySpecs() {

        super.displaySpecs();
        System.out.println("Camera (MP): " + camera);
        System.out.println("Price: BDT" + price);

    }

}

public class FeaturePhone extends Mobile {

    int battery;

    FeaturePhone(String brand, String color, int battery) {

        super(brand, color);
        this.battery = battery;

    }
```

```
 @Override

   void displaySpecs() {

      super.displaySpecs();
      System.out.println("Battery (mAh): " + battery);

   }

}


public class Main {

   public static void main(String[] args) {


      Smartphone smartphone = new Smartphone("Samsung", "Black", 48, 59900.00);

      FeaturePhone featurePhone = new FeaturePhone("Pixel", "Silver", 4500);


      System.out.println("Smartphone Specs:");
      smartphone.displaySpecs();

      System.out.println("\nFeature Phone Specs:");
      featurePhone.displaySpecs();

   }

}
```

Output:

```
Smartphone Specs:

Brand: Samsung

Color: Black

Camera (MP): 48

Price: BDT 59900.00


Feature Phone Specs:

Brand: Pixel

Color: Silver

Battery (mAh): 4500
```

Problem: 05

```
Package Apartment_Project;

public class Building {

   String flat = "Apartment";
   String floor = "Ground";

   Building() {

      System.out.println("Inside Building constructor");

   }
}

public class House extends Building {

   String door = "Front Door";
   String window = "Living Room Window";

   House() {

      super();
      System.out.println("Inside House constructor");

   }

   void display() {

      System.out.println("Type of flat: " + flat);
      System.out.println("Floor: " + floor);
      System.out.println("Door: " + door);
      System.out.println("Window: " + window);

   }
}

public class Main {

   public static void main(String[] args) {

      House myHouse = new House();
      myHouse.display();

   }

}
```

Output:

```
Inside Building constructor

Inside House constructor

Type of flat: Apartment     Floor: Ground     Door: Front Door

Window: Living Room Window
```

Lab Report

Course Tittle: Object Oriented Programming

Course Code: CSE 215

Experiment No: 03

Experiment Name: Implementation of inheritance based on some real life scenarios.

Submitted To:

Name: Mst. Umme Ayman

Designation: Lecturer

Department of CSE

Daffodil International University

Submitted By

Name: Md Raduan Ahamed

ID: 0242220005101839

Section: 63_O

Department of CSE

Daffodil International University

Submission Date: 12-09-2023

Problem: 01

```
// Define the Student interface
interface Student {

    String getName();
    int getId();
    String getUniversity();
    String getDepartment();
    double getCgpa();
}

// Create a class that implements the Student interface

class UniversityStudent implements Student {
    private String name;
    private int id;
    private String university;
    private String department;
    private double cgpa;

    // Constructor

    public UniversityStudent(String name, int id, String university, String
department, double cgpa) {

        this.name = name;
        this.id = id;
        this.university = university;
        this.department = department;
        this.cgpa = cgpa;
    }

    // Implement the interface methods

    public String getName() {
        return name;
    }

    public int getId() {
        return id;
    }

    public String getUniversity() {
        return university;
    }

    public String getDepartment() {
        return department;
    }

    public double getCgpa() {
        return cgpa;
    }

    @Override
    public String toString() {

        return "Name: " + getName() + "\n" +
               "ID: " + getId() + "\n" +
               "University: " + getUniversity() + "\n" +
```

```java
                "Department: " + getDepartment() + "\n" +
                "CGPA: " + getCgpa();
    }
}

public class Main {

    public static void main(String[] args) {

        // Create a UniversityStudent object

        UniversityStudent student = new UniversityStudent("Md Raduan
Ahamed",1839, "Daffodil International University", "Computer Science",
3.75);

        // Display the student's details
        System.out.println("Student Details:");
        System.out.println(student);
    }
}
```

Output:

```
Student Details:

Name: Md Raduan Ahamed

ID: 1839

University: Daffodil International University

Department: Computer Science

CGPA: 3.75
```

Problem: 02

```java
// Define an interface to represent a geographical entity
interface GeographicEntity {

    double getArea();          // Method to get the area of the entity
    String getName();          // Method to get the name of the entity
}

// Create a class to represent a City

class City implements GeographicEntity {

    private String name;
    private double area;

    public City(String name, double area) {

        this.name = name;
        this.area = area;
    }

    @Override
    public double getArea() {
        return area;
    }

    @Override
    public String getName() {

        return name;
    }
}

// Create a class to represent a Village

class Village implements GeographicEntity {

    private String name;
    private double area;

    public Village(String name, double area) {

        this.name = name;
        this.area = area;
    }


    @Override
    public double getArea() {

        return area;
    }


    @Override
    public String getName() {

        return name;
    }
}

// Create a class to represent a Country

class Country {

    private String name;
    private long population;
    private double totalArea;
    private List<GeographicEntity> entities;

    public Country(String name) {

        this.name = name;
        this.entities = new ArrayList<>();
    }

    public void addEntity(GeographicEntity entity) {

        entities.add(entity);
        totalArea += entity.getArea();
    }

    public void setPopulation(long population) {
        this.population = population;
    }

    public long getPopulation() {

        return population;
    }

    public double getTotalArea() {

        return totalArea;
    }

    public String getName() {

        return name;
    }

    public void printCountryInfo() {

        System.out.println("Country: " + name);
        System.out.println("Population: " + population);
        System.out.println("Total Area: " + totalArea + " square
kilometers");
        System.out.println("Cities and Villages:");
        for (GeographicEntity entity : entities) {
            System.out.println("- " + entity.getName() + ": " +
entity.getArea() + " square kilometers");

        }
    }
}


public class Main {

    public static void main(String[] args) {

        // Create a country

        Country country = new Country("Bangladesh");
        country.setPopulation(20,00,00,000);

       // Set the population of the country

        // Create cities and villages

        City city1 = new City("City 1", 300.0);
        City city2 = new City("City 2", 250.0);
        Village village1 = new Village("Village 1", 100.0);

        // Add cities and villages to the country

        country.addEntity(city1);
        country.addEntity(city2);
        country.addEntity(village1);

        // Print country information

        country.printCountryInfo();
    }
}
```

Output:

```
Country: Bangladesh

Population: 20,00,00,000

Total Area: 1,47,570.00 square kilometers

Cities and Villages:

- City 1: 300.0 square kilometers

- City 2: 250.0 square kilometers

- Village 1: 100.0 square kilometers
```

Problem: 03

```
// Vehicle interface
interface Vehicle {
    void start();
    void stop();
    void honk();
}

// Car class implementing Vehicle

class Car implements Vehicle {
    private String make;
    private String model;

    public Car(String make, String model) {
        this.make = make;
        this.model = model;
    }

    @Override
    public void start() {
        System.out.println("Starting the " + make + " " + model);
    }

    @Override
    public void stop() {
        System.out.println("Stopping the " + make + " " + model);
    }

    @Override
    public void honk() {
        System.out.println("Honking the horn of the " + make + " " +
model);
    }
}

// Bicycle class implementing Vehicle

class Bicycle implements Vehicle {
```

```java
    private String brand;

    public Bicycle(String brand) {
        this.brand = brand;
    }

    @Override
    public void start() {
        System.out.println("Starting the " + brand + " bicycle");
    }

    @Override
    public void stop() {
        System.out.println("Stopping the " + brand + " bicycle");
    }

    @Override
    public void honk() {
        System.out.println("Bicycles don't have horns!");
    }
}

public class Main {
    public static void main(String[] args) {
        Vehicle car = new Car("Toyota", "Camry");
        Vehicle bicycle = new Bicycle("Trek");

        car.start();
        car.honk();
        car.stop();

        System.out.println();

        bicycle.start();
        bicycle.honk();
        bicycle.stop();
    }
}
```

Output:

```
Starting the Toyota Camry

Honking the horn of the Toyota Camry

Stopping the Toyota Camry


Starting the Trek bicycle

Bicycles don't have horns!

Stopping the Trek bicycle
```

Lab Report

Course Tittle: Object Oriented Programming

Course Code: CSE 215

Experiment No: 04

Experiment Name: UML to JAVA code Implementation

Submitted To:

Name: Mst. Umme Ayman

Designation: Lecturer

Department of CSE

Daffodil International University

Submitted By

Name: Md Raduan Ahamed

ID: 0242220005101839

Section: 63_O

Department of CSE

Daffodil International University

Submission Date: 19-09-2023

PROBLEM: 01

```
package UML_performance;

// Define the shape interface

public interface shape {
    double calculatearea();
}

// Create a circle class that implements the shape interface

public class circle implements shape {
    double radious;

    //using constructor
         // using this keyword

    public circle(double radious){
        this.radious=radious;
    }
    public double calculatearea(){
        return 3.1416*radious*radious;
    }
}

// Create a rectangular class that implements the shape interface

public class circle implements shape {

public class rectangular implements shape {
    double length;
    double width;

    //using constructor
        // using this keyword

    public rectangular(double length, double width){
        this.length=length;
        this.width=width;
    }

    //declare the area number and implementation
        //create a main method in this function

    public double calculatearea(){
        return 0.5*length*width;
    }

    public static void main(String[] args) {

        circle cir = new circle(9.0);
        rectangular rec = new rectangular(9, 11);

        System.out.println("The Area of circle: "+ cir.calculatearea());
        System.out.println("The Area of rectangular: "+
rec.calculatearea());

    }
}
```

OUTPUT :

```
The Area of circle: 254.4696

The Area of rectangular: 49.5
```

PROBLEM: 02

```java
package UML_Perform;

// Define the shape interface

public interface Person {
    void display();

}

// Create a professor class that implements the person interface

public class Professor implements Person {
    String name;
    int id;

//using constructor
    //using this keyword

    public Professor(String name, int id) {
        this.id = id;
        this.name = name;
    }

//including display function

    public void display() {
        System.out.println("Professor name: " + name + "\nID: " + id);
    }
}

//define the department class

public class Department {
    String deptname;
    String address;

//using constructor
        //using this keyword

    public Department(String deptname, String address) {
        this.deptname = deptname;
        this.address = address;
    }

//define the display function

    public void display() {
        System.out.println("Department name: " + deptname + "\nAddress: " +
address);
    }
}
```

```
//student implement person

public class Student implements Person {
    String name;
    int id;
    Department deptinfo;

// using constructor
    //using this keyword

    public Student(String name, int id, Department deptinfo) {
        this.name = name;
        this.id = id;
        this.deptinfo = deptinfo;
    }

//define display methode

    public void display() {
        System.out.println("Student's name: " + name + "\nStudent's Id: " +
id);
        deptinfo.display();
    }

//create a mian class and declare data of this

    public static void main(String[] args) {

        Department dept = new Department("Computer Science", "Uttara, Rajuk
Appartment");
        Student std = new Student("Md Raduan Ahamed", 0242220005101839,
dept);
        Professor pro = new Professor("Dr. Jubaidul Alam Vuia", 5556120);

        std.display();
        pro.display();

    }
}
```

Output:

```
Student's name: Md Raduan Ahamed

Student's Id: 0242220005101839

Department name: Computer Science

Address: Utttara, Rajuk Appartment

Professor name: Dr. Jubaidul Alam Vuia

ID: 5556120
```

Lab Report

Course Tittle: Object Oriented Programming Lab

Course Code: CSE 215

Experiment No: 05

Experiment Name: Implementation of Encapsulation

Submitted To:

Name: Mst. Umme Ayman

Designation: Lecturer

Department of CSE

Daffodil International University

Submitted By

Name: Md Raduan Ahamed

ID: 0242220005101839

Section: 63_O

Department of CSE

Daffodil International University

Submission Date: 14-11-2023

Problem: 01

Achieving encapsulation (Accessing private variables of an encapsulated class from external package)

```java
package a;
public class class_1 {

    private String name; // declear private variable
    private int age;
    private String address;
    private double ph_num;

    public class_1() {

    }

    public void setName(String name){

        this.name=name; //access this private variable
    }
    public void setAge(int age){

        this.age=age; //access this private variable
    }
    public void setAddress(String address){

        this.address=address; //access this private variable
    }

    public String getName(){

        return name; //return value
    }
    public int getAge(){

        return age; //return value
    }
    public String getAddress(){

        return address; //return value
    }
}


package b;
import a.class_1; //import package
public class class_2 {

    public static void main(String[] args) {

        class_1 c = new class_1(); //object creation

        c.setName("Raduan"); //set value
        c.setAge(21);
        c.setAddress("Rajuk Uttara");
       //getvalue
        System.out.println("Name of this person : "+c.getName());
        System.out.println("Age of this person : "+c.getAge());
        System.out.println("Address of this person : "+c.getAddress());
    }
}
```

<u>Output:</u>

Name of this person : Raduan

Age of this person : 21

Address of this person : Rajuk Uttara

<u>Problem: 02</u>

Example 3, example 4 from slide.

<u>Example 3:</u>

```
class student{

    private String name;
    private int Id;
    private double cgpa;
    public void setName(String name){
                                   //use setter
        this.name=name;           //override
    }
    public void setId(int Id){         //use setter
        this.Id =Id;
    }
    public void setCgpa(double cgpa){  //use setter
        this.cgpa=cgpa;                //override
    }
    public String getName(){           //use getter
        return name;
    }
    public int getId(){
        return Id;
    }
    public double getCgpa(){
        return cgpa;
    }
}

public class myclass {

    public static void main(String[] args)  {

        student s = new student();     //object creation
        s.setName("Raduan");           //set value
        s.setId(1839);
        s.setCgpa(3.59);
        //call getter method

        System.out.println("Name of student : " + s.getName());
        System.out.println("Student's Id : " + s.getId());
        System.out.println("Result : " + s.getCgpa());
    }
}
```

<u>Output:</u>

```
Name of student : Raduan

Student's Id : 1839

Result : 3.59
```

<u>Example 4:</u>

```
package b;
class employee{

private String name;
private int Id;
private double salary;
public void setName(String name){
                               //use setter
this.name=name;                //override
  }
public void setId(int Id){     //use setter
this.Id =Id;
  }
public void setSalary(double salary){ //use setter
this.salary=salary;                   //override
  }
public String getName(){              //use getter
return name;
  }
public int getId(){
return Id;
  }
public double getSalary(){
return salary;
  }
}

public class my_class {

public static void main(String[] args) {

employee e = new employee();   //object creation
e.setName("Raduan");           //set value
e.setId(1839);
e.setSalary(50000);
//call getter method
System.out.println("Name of student : "+e.getName());
System.out.println("Student's Id : "+e.getId());
System.out.println("Result : "+e.getSalary());
```

```
    }
}
```

<u>Output:</u>

```
Name of employee : Raduan

Employee's Id : 1839

Salary : 50000
```

Lab Report

Course Tittle: Object Oriented Programming Lab

Course Code: CSE 215

Experiment No: 06

Experiment Name: Implementation of Abstraction

Submitted To:

Name: Mst. Umme Ayman

Designation: Lecturer

Department of CSE

Daffodil International University

Submitted By

Name: Md Raduan Ahamed

ID: 0242220005101839

Section: 63_O

Department of CSE

Daffodil International University

Submission Date: 14-11-2023

Problem: 01

You can create an abstract Shape class with methods like area() and perimeter(). Then, you can create concrete subclasses like Circle and Rectangle that provide specific implementations of these methods.

```java
abstract class shape {
 //use abstract class

    public abstract double area();
    public abstract double perimeter();
}

class circle extends shape {

    private double r ;
    circle(double r){                          //constructor
        this.r=r;

    }
    public double area(){

        return 3.1416*r*r;
    }
    public double perimeter(){

        return 2*3.1416*r;
    }
}

class rectangle extends shape {

    private double h,l ;
    rectangle(double h,double l){              //constractor
        this.h=h;
        this.l=l;
    }

    public double area(){
        return h*l;
    }
    public double perimeter(){
        return 2*(h+l);
    }
}

public class lab_r {

    public static void main(String[] args) {

        circle c= new circle(9);                //create object for circle
        System.out.println("Area of Circle :"+c.area());
        System.out.println("Perimeter of Circle :"+c.perimeter());

        rectangle re =new rectangle(6, 8);     //create object for rectangle
        System.out.println("Area of Rectangle :"+re.area());
        System.out.println("Perimeter of Rectangle : "+re.perimeter());
        }
}
```

Output:

```
Area of Circle : 153.9384

Perimeter of Circle : 56.5488

Area of Rectangle : 48.0

Perimeter of Rectangle : 28.0
```

Problem: 02

In a banking system, you can use abstraction to model bank accounts. Create an abstract class BankAccount with methods like deposit(), withdraw(), and getBalance(). Then, implement concrete classes for different types of accounts like SavingsAccount and CheckingAccount.

```
package b;
abstract class bankaccount {                    //create abstract class
public abstract void setBalance(double balance);
public abstract void deposit(double amount);    //create abstract method
public abstract void withdraw(double amount);
public abstract double getBalance();
}
class saving_account extends bankaccount {
private double balance;
@Override
public void setBalance(double balance) {        //use setter
this.balance = balance;
}
@Override
public void deposit(double amount) {
```

```
balance += amount;
}
@Override
public void withdraw(double amount) {

balance -= amount;
}
@Override
public double getBalance() {
return balance;
}
}
class checking_account extends bankaccount {
private double balance;
@Override
public void setBalance(double balance) {                    //use setter
this.balance = balance;
}
@Override
public void deposit(double amount) {
balance += amount;
}
@Override
public void withdraw(double amount) {
balance -= amount;
}
@Override
public double getBalance() {
return balance;
```

```
}
}
public class bank {
public static void main(String[] args) {
saving_account save = new saving_account();                    // create object
save.setBalance(10000);
save.deposit(500);
save.withdraw(1000);
System.out.println("Current Savings Account Balance: " +
save.getBalance());
checking_account check = new checking_account();           // create object
check.setBalance(15000);
check.deposit(1000);
check.withdraw(500);
System.out.println("Current Checking Account Balance: " +
check.getBalance());
}
}
```

<u>Output:</u>

```
 Current Savings Account Balance: 9500.0
Current Checking Account Balance: 16000.0
```

Lab Report

Course Tittle: Object Oriented Programming Lab

Course Code: CSE 215

Experiment No: 07

Experiment Name: Designing Library Management System by utilizing Encapsulation, Dependency and Collection Framework of JAVA

Submitted To:

Name: Mst. Umme Ayman

Designation: Lecturer

Department of CSE

Daffodil International University

Submitted By

Name: Md Raduan Ahamed

ID: 0242220005101839

Section: 63_O

Department of CSE

Daffodil International University

Submission Date: 23-11-2023

Suppose, you are developing a LibrarySystem. Where it contains a list of books (Books), a list of librarian (Librarian), and a list of transactions (Transactions). It has methods to add books and librarian, transactions, as well as methods to handle checking out and returning books. Book has title, author, ISBN. Librarian has name, phone_no, a list of transactions. Transaction consists of book, librarian, and date and it has a method as createTransaction which takes parameters such as book, librarian, and date to crate transaction history. LibrarySystem, Book and Librarian have the method as their name.

```java
import java.util.ArrayList;
import java.util.Date;
import java.util.List;

class Book {
    private String title;
    private String author;
    private String ISBN;

    public Book(String title, String author, String ISBN) {
        this.title = title;
        this.author = author;
        this.ISBN = ISBN;
    }

    // Getters and Setters

    public String getTitle() {
        return title;
    }

    public String getAuthor() {
        return author;
    }

    public String getISBN() {
        return ISBN;
    }
}


import java.util.ArrayList;
import java.util.List;

class Librarian {
    private String name;
    private String phoneNo;
    private List<Transaction> transactions;

    public Librarian(String name, String phoneNo) {
        this.name = name;
        this.phoneNo = phoneNo;
        this.transactions = new ArrayList<>();
    }
    // Getters and Setters

    public String getName() {
        return name;
    }

    public String getPhoneNo() {
        return phoneNo;
    }

    public List<Transaction> getTransactions() {
        return transactions;
    }

    public void addTransaction(Transaction transaction) {
        transactions.add(transaction);
    }
}


import java.util.Date;

class Transaction {
    private Book book;
    private Librarian librarian;
    private Date date;

    public Transaction(Book book, Librarian librarian, Date date) {
        this.book = book;
        this.librarian = librarian;
        this.date = date;
    }

    // Getters

    public Book getBook() {
        return book;
    }

    public Librarian getLibrarian() {
        return librarian;
    }

    public Date getDate() {
        return date;
    }
}


import java.util.ArrayList;
import java.util.Date;
import java.util.List;

public class librarySystem {
    private List<Book> books;
    private List<Librarian> librarians;
    private List<Transaction> transactions;

    public librarySystem() {
        this.books = new ArrayList<>();
        this.librarians = new ArrayList<>();
        this.transactions = new ArrayList<>();
    }

    // Methods to add books and librarians

    public void addBook(Book book) {
        books.add(book);
    }

    public void addLibrarian(Librarian librarian) {
        librarians.add(librarian);
    }

    // Methods to handle transactions

    public void checkOutBook(Book book, Librarian librarian, Date date) {
        if (books.contains(book) && librarians.contains(librarian)) {
            Transaction transaction = new Transaction(book, librarian,
date);
            transactions.add(transaction);
            librarian.addTransaction(transaction);
            System.out.println("Book checked out successfully.");
        } else {
            System.out.println("Book or librarian not found.");
        }
    }

    public void returnBook(Book book, Librarian librarian, Date date) {
        if (books.contains(book) && librarians.contains(librarian)) {
            Transaction transaction = new Transaction(book, librarian,
date);
            transactions.add(transaction);
            librarian.addTransaction(transaction);
            System.out.println("Book returned successfully.");
        } else {
            System.out.println("Book or librarian not found.");
        }
    }

    public static void main(String[] args) {
        librarySystem librarySystem = new librarySystem();

        Book book1 = new Book("Messege", "Dr Mizanur Rahman Azhari", "978-
0-316");
        Book book2 = new Book("English Therapy", "Saiful Islam", "978-0-
06");

        librarySystem.addBook(book1);
        librarySystem.addBook(book2);

        Librarian librarian = new Librarian("Raduan Ahamed",
"01785566224");
        librarySystem.addLibrarian(librarian);

        Date currentDate = new Date();

        librarySystem.checkOutBook(book1, librarian, currentDate);
        librarySystem.returnBook(book2, librarian, currentDate);

        // Print transaction history for the librarian
        for (Transaction transaction : librarian.getTransactions()) {
            System.out.println("Transaction: Book - " +
transaction.getBook().getTitle()
                    + ", Librarian - " +
transaction.getLibrarian().getName()
                    + ", Date - " + transaction.getDate());
        }
    }
}
```

Output:

```
Book checked out successfully.

Book returned successfully.

Transaction: Book - Messege, Librarian - Raduan Ahamed, Date - Sun Dec 10 16:18:07 BDT 2023

Transaction: Book - English Therapy, Librarian - Raduan Ahamed, Date - Sun Dec 10 16:18:07 BDT
2023
```

Lab Report

Course Tittle: Object Oriented Programming Lab

Course Code: CSE 215

Experiment No: 08

Experiment Name: Designing Student Management System by utilizing Encapsulation, Dependency and Collection Framework of JAVA

Submitted To:

Name: Mst. Umme Ayman

Designation: Lecturer

Department of CSE

Daffodil International University

Submitted By

Name: Md Raduan Ahamed

ID: 0242220005101839

Section: 63_O

Department of CSE

Daffodil International University

Submission Date: 29-11-2023

Managing student records is essential for educational institutions. Suppose you are a software developer working at DIU and have been tasked with creating an application to meet their specific needs. The system's primary functions include student registration and the ability to search for students by their student ID or mobile number. Furthermore, upon a student's successful completion of their studies, the system should provide the option to mark them as "completed," ensuring that they no longer appear in search results for active students.

```java
import java.util.*;
import java.util.Scanner;

class Student {
    private String Id;
    private String mobileNumber;
    private boolean isCompleted;

    public Student(String Id, String mobileNumber) {
        this.Id = Id;
        this.mobileNumber = mobileNumber;
        this.isCompleted = false;
    }

    public String getId() {
        return Id;
    }

    public String getMobileNumber() {
        return mobileNumber;
    }

    public boolean isCompleted() {
        return isCompleted;
    }

    public void markCompleted() {
        this.isCompleted = true;
    }

    @Override
    public String toString() {
        return "Student ID: " + Id + ", Mobile Number: " + mobileNumber +
", Completed: " + isCompleted;
    }
}



import java.util.ArrayList;
import java.util.Iterator;
import java.util.Scanner;

public class student_management{
    private ArrayList<Student> students;

    public student_management() {
        this.students = new ArrayList<>();
    }

    public void registerStudent(String Id, String mobileNumber) {
        students.add(new Student(Id, mobileNumber));
        System.out.println("Student registered successfully.");
    }

    public Student searchStudent(String query) {
        for (Student student : students) {
            if (student.getId().equals(query) ||
student.getMobileNumber().equals(query)) {
                return student;
            }
        }
        return null; // Return null if no student is found
    }

    public void markStudentCompleted(String Id) {
        Iterator<Student> iterator = students.iterator();
        while (iterator.hasNext()) {
            Student student = iterator.next();
            if (student.getId().equals(Id)) {
                student.markCompleted();
                System.out.println("Student marked as completed.");
                return;
            }
        }
        System.out.println("Student not found.");
    }

    public void displayAllStudents() {
        System.out.println("All Students:");
        for (Student student : students) {
            System.out.println(student);
        }
    }
    public static void main(String[] args) {
        student_management system = new student_management();
        Scanner scanner = new Scanner(System.in);

        // Example: Register students
        system.registerStudent("1839", "1234567890");
        system.registerStudent("1620", "1234568970");

        // Example: Search for a student
        System.out.print("Enter student ID or mobile number to search: ");
        String searchQuery = scanner.nextLine();
        Student foundStudent = system.searchStudent(searchQuery);
        if (foundStudent != null) {
            System.out.println("Student found: " + foundStudent);
        } else {
            System.out.println("Student not found.");
        }

        // Example: Mark a student as completed
        System.out.print("Enter student ID to mark as completed: ");
        String studentIdToComplete = scanner.nextLine();
        system.markStudentCompleted(studentIdToComplete);


        // Example: Display all students
        system.displayAllStudents();
    }
}
```

Output:

```
Student registered successfully.

Student registered successfully.

Enter student ID or mobile number to search: 1855

Student not found.

Enter student ID to mark as completed: 1839

Student marked as completed.

All Students:

Student ID: 1839, Mobile Number: 1234567890, Completed: true

Student ID: 1620, Mobile Number: 1234568970, Completed: false
```

Lab Report

Course Tittle: Object Oriented Programming Lab

Course Code: CSE 215

Experiment No: 09

Experiment Name: Designing Student Management System by utilizing Encapsulation, Dependency and Collection Framework of JAVA

Submitted To:

Name: Mst. Umme Ayman

Designation: Lecturer

Department of CSE

Daffodil International University

Submitted By

Name: Md Raduan Ahamed

ID: 0242220005101839

Section: 63_O

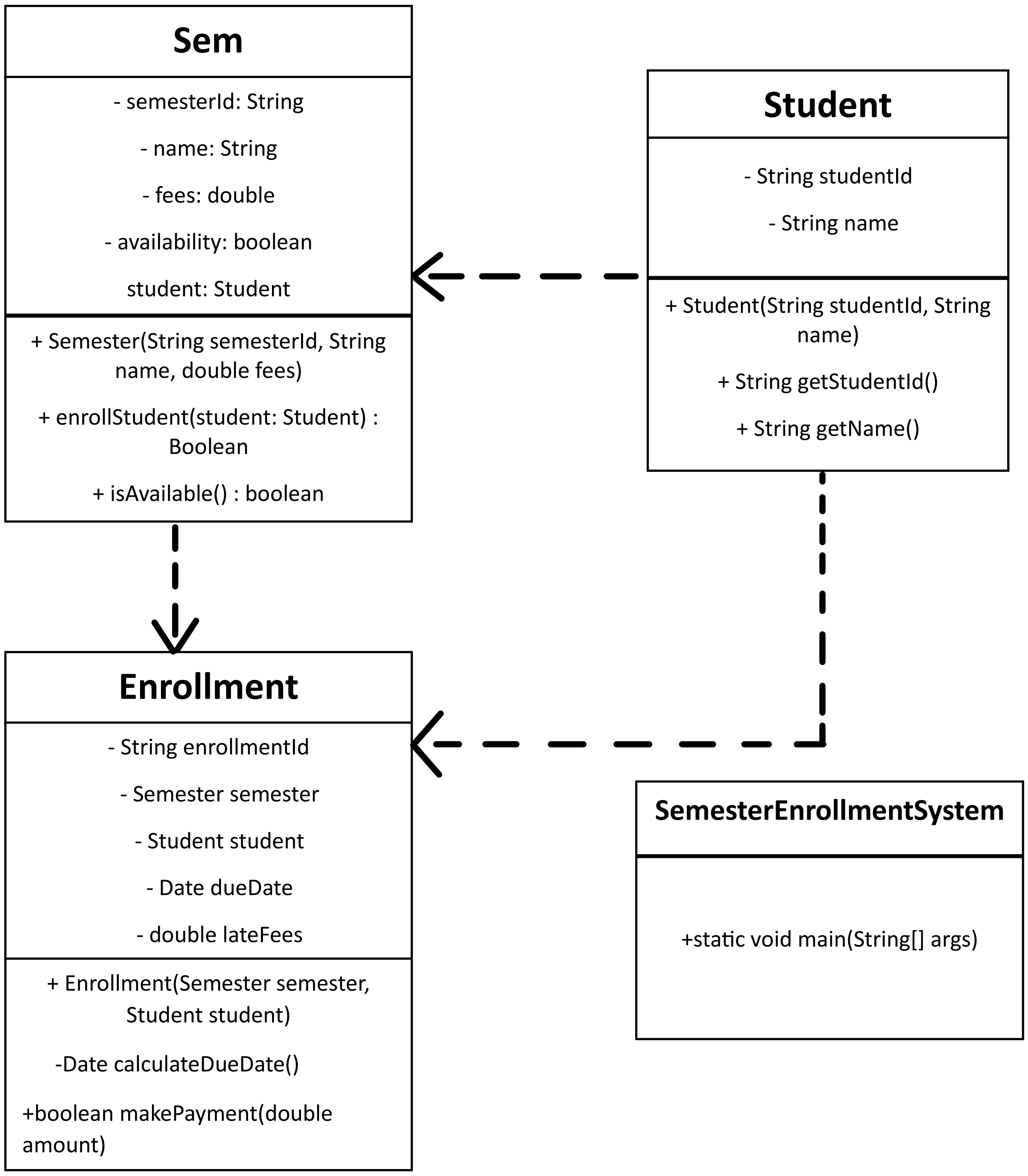
Department of CSE

Daffodil International University

Submission Date: 29-11-2023

Design a SemesterEnrollmentSystem with classes to manage students, semester, enrollment.Semester has smester_id, name, fees, availability status.Student can enroll to the Semester.The system also keeps track of Enrollment with due dates and late fees.Now,

1.Design the UML diagram of above mentioned scenario.



2.Write the java code of above UML.

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

    public String getStudentId() {
        return studentId;
    }

    public String getName() {
        return name;
    }

    @Override
    public String toString() {
        return "Student ID: " + studentId + ", Name: " + name+"\n";
    }
}
```

```
class sem {
   private int semesterId;
   private String name;
   private double fees;
   private boolean isAvailable;

   public sem(int semesterId, String name, double fees, boolean isAvailable) {
      this.semesterId = semesterId;
      this.name = name;
      this.fees = fees;
      this.isAvailable = isAvailable;
   }

   public int getSemesterId() {
      return semesterId;
   }

   public String getName() {
      return name;
   }

   public double getFees() {
      return fees;
   }

   public boolean isAvailable() {
      return isAvailable;
   }

   @Override
```

```
  public String toString() {
    return "Semester ID: " + semesterId + " Name: " + name + ", Fees: $" + fees + ", Available: " +
isAvailable;
  }
}
```

```
import java.util.Date;

class Enrollment {

    private Student student;
    private sem semester;
    private Date dueDate;
    private double lateFee;

    public Enrollment(Student student, sem semester, Date dueDate, double
lateFee) {
        this.student = student;
        this.semester = semester;
        this.dueDate = dueDate;
        this.lateFee = lateFee;
    }

    public Student getStudent() {
        return student;
    }

    public sem getSemester() {
        return semester;
    }

    public Date getDueDate() {
        return dueDate;
    }

    public double getLateFee() {
        return lateFee;
    }

    public boolean isLate() {
        Date currentDate = new Date();
        return currentDate.after(dueDate);
    }

    @Override
    public String toString() {
        return "Enrollment Details - " + student + "\n" + semester + ", Due
Date: " + dueDate + ", Late Fee: $" + lateFee;
    }
}
```

```
import java.util.Date;

public class SemesterEnrollmentSystem {
    public static void main(String[] args) {
        // Create students
        Student student1 = new Student("1839", "Raduan Ahamed");
        Student student2 = new Student("1830", "Rawnok Riddi");

        // Create semesters
        sem semester1 = new sem(1, "Fall 2022", 4000.0, true);
        sem semester2 = new sem(2, "Fall 2023", 3200.0, true);

        // Create enrollments
        Enrollment enrollment1 = new Enrollment(student1, semester1, new
Date(), 200.0);
        Enrollment enrollment2 = new Enrollment(student2, semester2, new
Date(), 150.0);

        // Display enrollment details
        System.out.println(enrollment1);
        System.out.println("Is late: " + enrollment1.isLate());

        System.out.println(enrollment2);
        System.out.println("Is late: " + enrollment2.isLate());
    }
}
```

Output:

```
Enrollment Details - Student ID: 1839, Name: Raduan Ahamed


Semester ID: 1 Name: Fall 2022, Fees: $4000.0, Available: true, Due Date: Sat Dec 09
16:45:24 BDT 2023, Late Fee: $200.0

Is late: true

Enrollment Details - Student ID: 1830, Name: Rawnok Riddi


Semester ID: 2 Name: Fall 2023, Fees: $3200.0, Available: true, Due Date: Sat Dec 09
16:45:24 BDT 2023, Late Fee: $150.0

Is late: true
```

Lab Report

Course Tittle: Object Oriented Programming Lab

Course Code: CSE 215

Experiment No: 10

Experiment Name: Designing Student Management System by utilizing Encapsulation, Dependency and Collection Framework of JAVA

Submitted To:

Name: Mst. Umme Ayman

Designation: Lecturer

Department of CSE

Daffodil International University

Submitted By

Name: Md Raduan Ahamed

ID: 0242220005101839

Section: 63_O

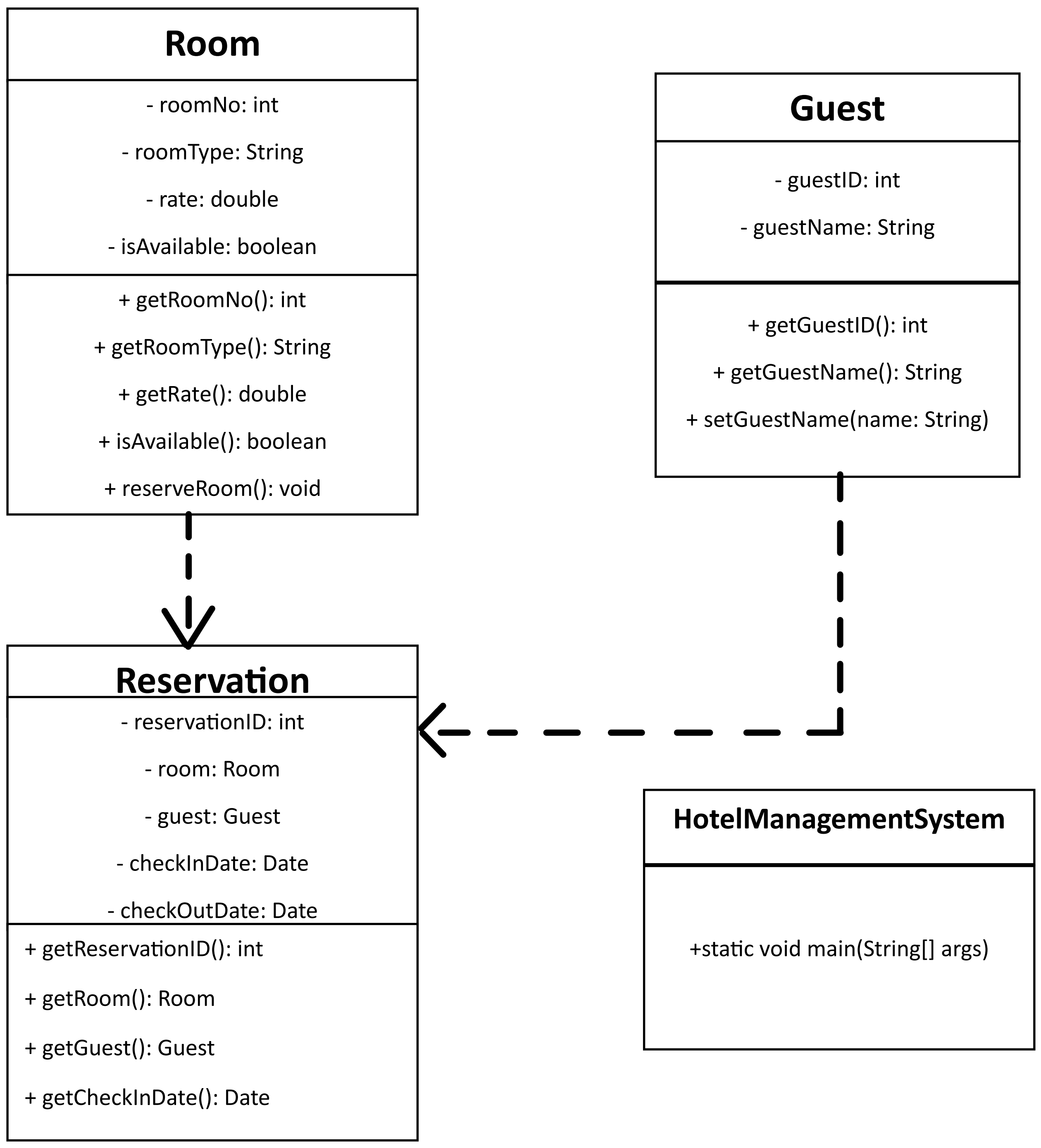
Department of CSE

Daffodil International University

Submission Date: 29-11-2023

You are asked with designing a Hotel Management System for a hotel. This system should have classes to manage Room, Guest, Reservation.Where Room has information like room_no, room_type, rate, availability status.Guests can reserve the room.The system also keeps tracks of Reservation.

1.Design the UML diagram of above mentioned scenario



2.Write the java code of above UML.

```
import java.util.ArrayList;
import java.util.List;

class Room {
    private int roomNo;
    private String roomType;
    private double rate;
    private boolean isAvailable;

    public Room(int roomNo, String roomType, double rate) {
        this.roomNo = roomNo;
        this.roomType = roomType;
        this.rate = rate;
        this.isAvailable = true; // Room is initially available
    }

    public int getRoomNo() {
        return roomNo;
    }

    public String getRoomType() {
        return roomType;
    }

    public double getRate() {
        return rate;
    }

    public boolean isAvailable() {
        return isAvailable;
    }

    public void setAvailable(boolean available) {
        isAvailable = available;
    }
}
```

```
class Guest {
  private String guestName;

  public Guest(String guestName) {
    this.guestName = guestName;
  }

  public String getGuestName() {
    return guestName;
  }
}
```

```
package lab_11;
```

```java
class Reservation {
    private Guest guest;
    private Room room;

    public Reservation(Guest guest, Room room) {
        this.guest = guest;
        this.room = room;
    }

    public Guest getGuest() {
        return guest;
    }

    public Room getRoom() {
        return room;
    }
}


import java.util.ArrayList;
import java.util.List;

public class hotel_management{
    private List<Room> rooms;
    private List<Reservation> reservations;

    public hotel_management() {
        rooms = new ArrayList<>();
        reservations = new ArrayList<>();
    }

    public void addRoom(Room room) {
        rooms.add(room);
    }

    public void displayAvailableRooms() {
        System.out.println("Available Rooms:");
        for (Room room : rooms) {
            if (room.isAvailable()) {
                System.out.println("Room No: " + room.getRoomNo() +
                        ", Type: " + room.getRoomType() +
                        ", Rate: $" + room.getRate());
            }
        }
    }

    public void makeReservation(Guest guest, Room room) {

        if (room.isAvailable()) {
            room.setAvailable(false);
            Reservation reservation = new Reservation(guest, room);
            reservations.add(reservation);
            System.out.println("Reservation made for " +
guest.getGuestName() +
                    " in Room No " + room.getRoomNo());
        } else {
            System.out.println("Room is not available for reservation.");
        }
    }

    public static void main(String[] args) {
        hotel_management hotelSystem = new hotel_management();

        // Adding rooms to the system
        hotelSystem.addRoom(new Room(111, "Single", 300.0));
        hotelSystem.addRoom(new Room(112, "Double", 750.0));
        hotelSystem.addRoom(new Room(113, "Suite", 600.0));

        // Display available rooms
        hotelSystem.displayAvailableRooms();

        // Making a reservation
        Guest guest1 = new Guest("John Doe");
        hotelSystem.makeReservation(guest1, hotelSystem.rooms.get(0));

        // Display available rooms after reservation
        hotelSystem.displayAvailableRooms();
    }
}
```

Output:

```
Available Rooms:

Room No: 111, Type: Single, Rate: $300.0

Room No: 112, Type: Double, Rate: $750.0

Room No: 113, Type: Suite, Rate: $600.0

Reservation made for John Doe in Room No 111

Available Rooms:

Room No: 112, Type: Double, Rate: $750.0

Room No: 113, Type: Suite, Rate: $600.0
```