OOM TUT 4

Q1. Polymorphic Payment System

```
abstract class PaymentMethod {
   abstract void processPayment(double amount);
   abstract void generatePaymentReceipt();
}

class CreditCardPayment extends PaymentMethod {
   private String cardNumber;
   private String cardHolder;

   public CreditCardPayment(String cardNumber, String cardHolder) {
      this.cardNumber = cardNumber;
      this.cardHolder = cardHolder;
   }

   @Override
   void processPayment(double amount){
      System.out.println("Processing Credit Card payment of Rs." + amount);
   }

   @Override
   void generatePaymentReceipt() {
      System.out.println("Credit Card Payment Receipt: ");
      System.out.println("Card Number: " + cardNumber);
      System.out.println("Card Holder: " + cardHolder);
      System.out.println("Transfer successful!");
   }
}

class PayPalPayment extends PaymentMethod {
   private String email;

   public PayPalPayment(String email) {
      this.email = email;
   }

   @Override
   void processPayment(double amount){
      System.out.println("Processing PayPal payment of Rs." + amount);
   }

   @Override
```

```
    void generatePaymentReceipt() {
        System.out.println("PayPal Payment Receipt: ");
        System.out.println("Email: " + email);
        System.out.println("Transfer successful!");
    }
}

class BitcoinPayment extends PaymentMethod {
    private String bitcoinAddress;

    public BitcoinPayment(String bitcoinAddress) {
        this.bitcoinAddress = bitcoinAddress;
    }

    @Override
    void processPayment(double amount){
        System.out.println("Processing Bitcoin payment of Rs." + amount);
    }

    @Override
    void generatePaymentReceipt() {
        System.out.println("Bitcoin Payment Receipt: ");
        System.out.println("Bitcoin Address: " + bitcoinAddress);
        System.out.println("Transfer successful!");
    }
}

public class PaymentSystem{

    static void processPaymentAndGeneratePaymentReceipt(PaymentMethod
method, double amount){
        method.processPayment(amount);
        method.generatePaymentReceipt();
        System.out.println();
    }
    public static void main(String[] args){
        PaymentMethod creditCard = new
CreditCardPayment("1234-4321-5678-8765", "Mansa Mahendru");
        PaymentMethod payPal = new PayPalPayment("iit2022017@iiita.ac.in");
        PaymentMethod bitcoin = new BitcoinPayment("qweRTYuiop");
        double amount = 150.0;
        System.out.println();
        processPaymentAndGeneratePaymentReceipt(creditCard, amount);
        processPaymentAndGeneratePaymentReceipt(payPal, amount);
        processPaymentAndGeneratePaymentReceipt(bitcoin, amount);
    }
}
```

Q2. Library Management System

```
import java.util.ArrayList;

class Author{
    String name;
    ArrayList<Book> books = new ArrayList<Book>();

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

    public void availableBooks(){
        for(int i=0; i<books.size(); i++){
            if(books.get(i).isAvailable==true)
            System.out.println(books.get(i).title);
        }
    }

    public void addAuthor(String authorName, Book b1, Book b2){
        books.add(b1);
        books.add(b2);
    }

    public void displayAuthorBooks(String authorName){
        System.out.println("Books by " + authorName + ": ");
        for(int i=0; i<books.size(); i++){
            if(books.get(i).a.name == authorName)
            System.out.println(books.get(i).title);
        }
    }
}

class Book{
    String title;
    Author a = new Author();
    String genre;
    boolean isAvailable;

    Book(String title, Author a, String genre, boolean isAvailable){
        this.title = title;
        this.a =a;
        this.genre = genre;
        this.isAvailable = isAvailable;
    }
}

class Patron{
    String name;
    ArrayList<Book> borrowedBooks = new ArrayList<Book>();

    public void newPatron(String patronName){
        this.name = patronName;
    }

    public void borrowBook(Book b){
        if(b.isAvailable==true){
            borrowedBooks.add(b);

            b.isAvailable = false;
        }
        else
        System.out.println( b.title + " book is not available");
    }

    public void returnBook(Book b){
        System.out.println("Book returned: ");
        if(borrowedBooks.contains(b)){
        borrowedBooks.remove(b);
        System.out.println(b.title);
        b.isAvailable = true;
        }
        else
        System.out.println("Book was not borrowed by this patron");
        System.out.println();
    }

    public void displayBorrowedBooks(Patron p){
        System.out.println("Books assigned to " + p.name + ": ");
        if(p.borrowedBooks.size()==0){
            System.out.println("None");
        }
        else{
        for(int i=0; i< p.borrowedBooks.size(); i++)
        System.out.println(p.borrowedBooks.get(i).title);
        }
    }
}

public class Library{
    public static void main(String[] args){
        Author a1 = new Author();
        a1.name = "Agatha Christie";
        Book b1 = new Book("And Then There Were None", a1, "Crime Thriller",
true);
        a1.addBook(b1);

        Author a2 = new Author();
        a2.name = "Rhonda Byrne";
        Book b2 = new Book("The Secret", a2, "Self-Help", true);
        Book b3 = new Book("The Magic", a2, "Self-Help", true);
        a2.addAuthor("Rhonda Byrne", b2, b3);

        Author a3 = new Author();
        a3.name = "Chetan Bhagat";
        Book b4 = new Book("2 States", a3, "Fiction", true);
        Book b5 = new Book("Half-Girlfriend", a3, "Romance", true);
        a3.addAuthor("Chetan Bhagat", b4, b5);

        System.out.println("Available Books: ");
        a1.availableBooks();
        a2.availableBooks();
        a3.availableBooks();
        System.out.println();

        Patron p1 = new Patron();
        p1.newPatron("Mansa");
        p1.borrowBook(b3);
        p1.borrowBook(b4);
        p1.displayBorrowedBooks(p1);
        System.out.println();

        a2.displayAuthorBooks("Rhonda Byrne");
        System.out.println();

        Patron p2 = new Patron();
        p2.newPatron("Bhagat");
        p2.borrowBook(b3);
        p2.displayBorrowedBooks(p2);
        p2.returnBook(b3);
        p1.returnBook(b4);
        System.out.println("---Updated Info---");
        System.out.println();
        System.out.println("Available Books: ");
        a1.availableBooks();
        a2.availableBooks();
        a3.availableBooks();
    }
}
```

Q3. Online Shopping System

```
import java.util.ArrayList;

class Product{
    int productId;
    String productName;
    double price;
    int quantityInStock;

    public void setProductId(int productId){
        this.productId = productId;
    }

    public void setProductName(String productName){
        this.productName = productName;
    }

    public void setPrice(double price){
        this.price = price;
    }

    public void setQuantityInStock(int quantityInStock){
        this.quantityInStock = quantityInStock;
    }

    public int getProductId() {
        return productId;
    }

    public String getProductName() {
        return productName;
    }
    public double getPrice() {
        return price;
    }

    public int getQuantityInStock() {
        return quantityInStock;
    }
}

class Customer{
    int customerId;
    String firstName;
    String lastName;
    String email;

    public void setcustomerId(int customerId){
        this.customerId = customerId;
    }

    public void setFirstName(String firstName){
        this.firstName = firstName;
    }

    public void setLastName(String lastName){
        this.lastName = lastName;
    }

    public void setEmail(String email){
        this.email = email;
    }

    public int getCustomerId() {
        return customerId;
    }

    public String getFirstName() {
        return firstName;
    }

    public String getLastName() {
        return lastName;
    }

    public String getEmail() {
        return email;
    }
}

class ShoppingCart{
    int cartId;
    Customer c = new Customer();
    ArrayList<Product> cartItems = new ArrayList<Product>();

    public void addItem(Product product, int quantity){
        if (quantity <= 0) {
            System.out.println("Quantity should be greater than zero!");
            return;
        }
        if (product.getQuantityInStock() < quantity) {
            System.out.println("Insufficient quantity in stock for " +
product.getProductName());
            return;
        }
        cartItems.add(product);
        product.setQuantityInStock(product.getQuantityInStock() - quantity);
        System.out.println(product.getProductName() + " (*" + quantity + ")
added to the cart");
    }

    public void removeItem(Product product){
        if (cartItems.remove(product)) {
            product.setQuantityInStock(product.getQuantityInStock() + 1);
            System.out.println(product.getProductName() + " removed from the
cart");
        } else {
            System.out.println(product.getProductName() + " not found in the
cart");
        }
    }

    public double calculateTotal(){
        double sum = 0;
        for(int i=0; i<cartItems.size(); i++){
            sum = sum + cartItems.get(i).price;
        }
        return sum;
    }
}

public class OnlineShop{
    public static void main(String[] args){
        Product p1 = new Product();
        p1.setProductId(5646);
        p1.setProductName("Astronaut Galaxy Projector");
        p1.setPrice(2949);
        p1.setQuantityInStock(4);

        Product p2 = new Product();
        p2.setProductId(7059);
        p2.setProductName("Gojo Oversized Crop Top");
        p2.setPrice(699);
        p2.setQuantityInStock(2);

        Product p3 = new Product();
        p3.setProductId(8573);
        p3.setProductName("Echo Dot 5th Gen");
        p3.setPrice(5499);
        p3.setQuantityInStock(3);

        Customer c1 = new Customer();
        ShoppingCart sc = new ShoppingCart();
        sc.cartId = 1007;
        sc.c = c1;
        System.out.println();
        sc.addItem(p1, 1);
        sc.addItem(p2, 2);
        sc.addItem(p3, 1);
        System.out.println();

        System.out.println("Total cost of items in the cart: Rs." +
sc.calculateTotal());
        System.out.println();
        sc.removeItem(p3);

        System.out.println();
        System.out.println("Updated total cost of items in the cart: Rs." +
sc.calculateTotal());
        System.out.println();
    }
}
```

Q4. University Course Management System

```
import java.util.ArrayList;

class Person {
    String firstName;
    String lastName;
    String email;

    public Person(String firstName, String lastName, String email) {
        this.firstName = firstName;
        this.lastName = lastName;
        this.email = email;
    }

    public void setFirstName(String firstName){
        this.firstName = firstName;
    }

    public void setLastName(String lastName){
        this.lastName = lastName;
    }

    public void setEmail(String email){
        this.email = email;
    }

    public String getFirstName() {
        return firstName;
    }

    public String getLastName() {
        return lastName;
    }

    public String getEmail() {
        return email;
    }
}

class Student extends Person {
    int studentId;
    ArrayList<Course> enrolledCourses = new ArrayList<>();
    ArrayList<Course> droppedCourses = new ArrayList<>();
    public Student(String firstName, String lastName, String email, int
studentId) {
        super(firstName, lastName, email);
        this.studentId = studentId;
        this.enrolledCourses = new ArrayList<>();
        this.droppedCourses = new ArrayList<>();
    }

    public void enrollCourse(Course course) {
        enrolledCourses.add(course);
        course.enrollStudent(this);
    }

    public void dropCourse(Course course) {
        enrolledCourses.remove(course);
        droppedCourses.add(course);
        course.dropStudent(this);
    }

    public void getEnrolledCourses(Student s) {
        System.out.println("Enrolled Courses: ");
        for(int i=0; i<s.enrolledCourses.size(); i++){
            System.out.println(s.enrolledCourses.get(i).courseName);
        }
    }

    public void getDroppedCourses(Student s) {
        if(s.droppedCourses.size() ==0){
            System.out.println("No courses dropped!");
        }
        else
        System.out.println("Dropped Courses: ");
        for(int i=0; i<s.droppedCourses.size(); i++){
            System.out.println(s.droppedCourses.get(i).courseName);
        }
    }

    public void studentInfo(){
        System.out.println("Student Name: " + firstName + " " + lastName);
        System.out.println("Student ID: " + studentId);
    }
}

class Instructor extends Person {
    int instructorId;
    ArrayList<Course> coursesTaught = new ArrayList<>();

    public Instructor(String firstName, String lastName, String email, int
instructorId) {
        super(firstName, lastName, email);
        this.instructorId = instructorId;
        this.coursesTaught = new ArrayList<>();
    }

    public void assignCourse(Course course) {
        coursesTaught.add(course);
        course.setInstructor(this);
    }

    public void getAssignedCourses() {
        for(int i=0; i<coursesTaught.size(); i++){
            System.out.println(coursesTaught.get(i));
        }
    }
}

class Course {
    int courseId;
    String courseName;
    Instructor instructor;
    ArrayList<Student> studentsEnrolled;
    ArrayList<Course> prerequisites;

    public Course(int courseId, String courseName) {
        this.courseId = courseId;
        this.courseName = courseName;
        this.studentsEnrolled = new ArrayList<>();
        this.prerequisites = new ArrayList<>();
    }

    public void enrollStudent(Student student) {
        studentsEnrolled.add(student);
    }

    public void dropStudent(Student student) {
        studentsEnrolled.remove(student);
    }

    public void setInstructor(Instructor instructor) {
        this.instructor = instructor;
```

```java
    }

    public void addPrerequisite(Course prerequisite) {
        prerequisites.add(prerequisite);
    }

    public boolean canEnroll(Student student) {
        for (Course prerequisite : prerequisites) {
            if (!student.enrolledCourses.contains(prerequisite)) {
                return false;
            }
        }
        return true;
    }

    public int getCourseId() {
        return courseId;
    }

    public String getCourseName() {
        return courseName;
    }

    public Instructor getInstructor() {
        return instructor;
    }

    public void getEnrolledStudents() {
        for(int i=0; i<studentsEnrolled.size(); i++){
            System.out.println(studentsEnrolled.get(i));
        }
    }

    public void getPrerequisites() {
        if(prerequisites.size() == 0){
            System.out.println("No prerequisites required!");
        }
        else{
        for(int i=0; i<prerequisites.size(); i++){
            System.out.println(prerequisites.get(i).courseName);
            }
        }
    }
}
```

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

      Student student1 = new Student("Mansa", "Mahendru",
"iit2022017@iiit.ac.in", 2022017);
      Student student2 = new Student("Prernendu", "Bhagat",
"iit2022016@iiit.ac.in", 2022016);


      Instructor instructor1 = new Instructor("Dr. Parikshit", "Joshi",
"drparikshitjoshi@iiita.ac.in", 2001);
      Instructor instructor2 = new Instructor("Dr. Radhika", "Gour",
"drradhikagour@iiita.ac.in", 2002);
      Instructor instructor3 = new Instructor("Dr. Vibha", "Yadav",
"drvibhayadav@iiita.ac.in", 2003);


      Course course1 = new Course(401, "Introduction to Marketing");
      Course course2 = new Course(402, "Principles of Economics");
      Course course3 = new Course(403, "Fundamentals of Electronics
Engineering");
      Course course4 = new Course(103, "Circuits");
      Course course5 = new Course(105, "Semiconductors");

      instructor1.assignCourse(course1);
      instructor1.assignCourse(course2);
      instructor2.assignCourse(course3);

      course3.addPrerequisite(course4);
      course3.addPrerequisite(course5);

      student1.enrollCourse(course1);
      student1.enrollCourse(course2);
      student2.enrollCourse(course4);
      student2.enrollCourse(course5);
      if(course3.canEnroll(student2)){
      student2.enrollCourse(course3);
      System.out.println(student2.firstName + " has the prerequisites covered
for " + course3.courseName);
      }
      else{
         System.out.println("Prerequisites not covered! Can't assign course to
" + student2.firstName);
      }
      System.out.println();
```

```
        System.out.println("Course Information:");
        System.out.println("Course 1: " + course1.getCourseName());
        System.out.println("Instructor: " + course1.getInstructor().getFirstName()
+ " " + course1.getInstructor().getLastName());
        System.out.println("Enrolled Students: " +
course1.studentsEnrolled.size());
        System.out.println("Prerequisites: ");
        course1.getPrerequisites();
        System.out.println();

        System.out.println("Course 2: " + course2.getCourseName());
        System.out.println("Instructor: " + course2.getInstructor().getFirstName()
+ " " + course2.getInstructor().getLastName());
        System.out.println("Enrolled Students: " +
course2.studentsEnrolled.size());
        System.out.println("Prerequisites: ");
        course2.getPrerequisites();
        System.out.println();

        System.out.println("Course 3: " + course3.getCourseName());
        System.out.println("Instructor: " + course3.getInstructor().getFirstName()
+ " " + course3.getInstructor().getLastName());
        System.out.println("Enrolled Students: " +
course3.studentsEnrolled.size());
        System.out.println("Prerequisites: ");
        course3.getPrerequisites();
        System.out.println();

        System.out.println("Student Information:");
        System.out.println("Student 1");
        student1.studentInfo();
        student1.getEnrolledCourses(student1);
        student1.getDroppedCourses(student1);
        System.out.println();
        System.out.println("Student 2");
        student2.studentInfo();
        student2.getEnrolledCourses(student2);
        student2.getDroppedCourses(student2);

        student1.dropCourse(course2);
        course2.setInstructor(instructor3);

        System.out.println();
        System.out.println("---Updated Info---");
        System.out.println();
        System.out.println("Course Information:");
        System.out.println("Course 1: " + course1.getCourseName());
        System.out.println("Instructor: " + course1.getInstructor().getFirstName()
+ " " + course1.getInstructor().getLastName());
        System.out.println("Enrolled Students: " +
course1.studentsEnrolled.size());
        System.out.println("Prerequisites: ");
        course1.getPrerequisites();
        System.out.println();

        System.out.println("Course 2: " + course2.getCourseName());
        System.out.println("Instructor: " + course2.getInstructor().getFirstName()
+ " " + course2.getInstructor().getLastName());
        System.out.println("Enrolled Students: " +
course2.studentsEnrolled.size());
        System.out.println("Prerequisites: ");
        course2.getPrerequisites();
        System.out.println();

        System.out.println("Course 3: " + course3.getCourseName());
        System.out.println("Instructor: " + course3.getInstructor().getFirstName()
+ " " + course3.getInstructor().getLastName());
        System.out.println("Enrolled Students: " +
course3.studentsEnrolled.size());
        System.out.println("Prerequisites: ");
        course3.getPrerequisites();
        System.out.println();

        System.out.println("Student Information:");
        System.out.println("Student 1");
        student1.studentInfo();
        student1.getEnrolledCourses(student1);
        student1.getDroppedCourses(student1);
        System.out.println();
        System.out.println("Student 2");
        student2.studentInfo();
        student2.getEnrolledCourses(student2);
        student2.getDroppedCourses(student2);
        System.out.println();
    }
}
```