Sol.1:

abstract class PaymentMethod {

double amount;

public void processPayment(double amount) {

}

public String generateReceipt() {

return "";

}

}

class CreditCard extends PaymentMethod {

private String cardNumber;

private String cardHolder;

public CreditCard(String cardNumber, String cardHolder) {

this.cardNumber = cardNumber;

this.cardHolder = cardHolder;

}

@Override

public void processPayment(double amount) {

System.out.println("Processing credit card payment of $" + amount);

}

@Override

public String generateReceipt() {

return "Credit Card Payment Receipt";

}

}

class Paypal extends PaymentMethod {

private String email;

public Paypal(String email) {

this.email = email;

}

@Override

public void processPayment(double amount) {

System.out.println("Processing PayPal payment of $" + amount);

}

@Override

public String generateReceipt() {

return "Paypal Payment Receipt";

}

}

class Bitcoin extends PaymentMethod {

private String id;

public Bitcoin(String id) {

this.id = id;

}

@Override

public void processPayment(double amount) {

System.out.println("Processing Bitcoin payment of $" + amount);

}

@Override

public String generateReceipt() {

return "Bitcoin Payment Receipt";

}

}

public class RunPayment {

public static void main(String[] args) {

PaymentMethod CreditCard=new CreditCard("1234-5678-9012-3456", "John Cena");

PaymentMethod Paypal = new Paypal("john@wwe.com");

PaymentMethod Bitcoin= new Bitcoin("nevergiveupcena");

double amount=100;

CreditCard.processPayment(amount);

CreditCard.generateReceipt();

Paypal.processPayment(amount);

Paypal.generateReceipt();

Bitcoin.processPayment(amount);

Bitcoin.generateReceipt();

}

}

Sol.2:

import java.util.\*;

class Author{

private String name;

private List<Book> books;

public Author(String name){

this.name=name;

this.books=new ArrayList<>();

}

public String getName(){

return name;

}

public List<Book> getBooks(){

return books;

}

public void addBooks(Book book){

books.add(book);

}

}

class Book{

private String title;

private Author author;

private String genre;

private boolean isAvailable;

public Book(String title, Author author, String genre, boolean isAvailable){

this.title=title;

this.author=author;

this.genre=genre;

this.isAvailable=isAvailable;

}

public String getTitle(){

return title;

}

public Author getAuthor(){

return author;

}

public String getGenre(){

return genre;

}

public boolean isAvailable(){

return isAvailable;

}

public void setAvailable(boolean available){

isAvailable=available;

}

}

class Patron{

private String name;

private List<Book> borrowedBooks;

public Patron(String name){

this.name=name;

this.borrowedBooks=new ArrayList<>();

}

public String getName(){

return name;

}

public List<Book> getBorrowedBooks(){

return borrowedBooks;

}

public void borrowBook(Book book){

if(book.isAvailable()){

borrowedBooks.add(book);

book.setAvailable(false);

}

else{

System.out.println("Book is not available for borrowing");

}

}

public void returnBook(Book book){

if(borrowedBooks.contains(book)){

borrowedBooks.remove(book);

book.setAvailable(true);

}

else{

System.out.println("Book was not borrowed by this patron.");

}

}

}

class Library{

private List<Book> books;

private Map<String, Author> authors;

private List<Patron> patrons;

public Library(){

this.books=new ArrayList<>();

this.authors=new HashMap<>();

this.patrons=new ArrayList<>();

}

public void addBook(Book book){

books.add(book);

Author author= book.getAuthor();

if(!authors.containsKey(author.getName())){

authors.put(author.getName(), author);

}

author.addBooks(book);

}

public void addAuthor(Author author){

authors.put(author.getName(), author);

}

public void registerPatron(Patron patron){

patrons.add(patron);

}

public void displayAvailableBooks(){

System.out.println("Available Books");

for(Book book: books){

if(book.isAvailable()){

System.out.println("Title: "+book.getTitle()+" Author: "+book.getAuthor()+" Genre: "+book.getGenre());

}

}

}

public void displayBorrowedBooksbyPatron(Patron patron){

System.out.println("Books borrowed by "+patron.getName());

for(Book book: patron.getBorrowedBooks()){

System.out.println("Title: "+book.getTitle()+" Author: "+book.getAuthor()+" Genre: "+book.getGenre());

}

}

public void displayBooksByAuthor(Author author){

System.out.println("Books written by " +author.getName());

for(Book book: author.getBooks()){

System.out.println("Title: "+book.getTitle()+" Genre: "+book.getGenre());

}

}

}

public class RunLibrary{

public static void main(String[] args) {

Library library = new Library();

Author author1 = new Author("Author 1");

Author author2 = new Author("Author 2");

Book book1 = new Book("Book 1", author1, "Genre 1", true);

Book book2 = new Book("Book 2", author1, "Genre 2", true);

Book book3 = new Book("Book 3", author2, "Genre 1", true);

library.addAuthor(author1);

library.addAuthor(author2);

library.addBook(book1);

library.addBook(book2);

library.addBook(book3);

Patron patron1 = new Patron("Patron 1");

Patron patron2 = new Patron("Patron 2");

library.registerPatron(patron1);

library.registerPatron(patron2);

patron1.borrowBook(book1);

patron1.borrowBook(book2);

patron2.borrowBook(book3);

library.displayAvailableBooks();

library.displayBorrowedBooksbyPatron(patron1);

library.displayBorrowedBooksbyPatron(patron2);

library.displayBooksByAuthor(author1);

library.displayBooksByAuthor(author2);

}

}

Sol.3:

import java.util.\*;

class Product {

private int productId;

private String productName;

private double price;

private int quantityInStock;

private List<Review> reviews;

public Product(int productId, String productName, double price, int quantityInStock) {

this.productId = productId;

this.productName = productName;

this.price = price;

this.quantityInStock = quantityInStock;

this.reviews = new ArrayList<>();

}

public int getProductId() {

return productId;

}

public String getProductName() {

return productName;

}

public double getPrice() {

return price;

}

public int getQuantityInStock() {

return quantityInStock;

}

public void setQuantityInStock(int quantityInStock) {

this.quantityInStock = quantityInStock;

}

public List<Review> getReviews() {

return reviews;

}

public void addReview(Review review) {

reviews.add(review);

}

}

class Customer {

private int customerId;

private String firstName;

private String lastName;

private String email;

public Customer(int customerId, String firstName, String lastName, String email) {

this.customerId = customerId;

this.firstName = firstName;

this.lastName = lastName;

this.email = email;

}

public int getCustomerId() {

return customerId;

}

public String getFirstName() {

return firstName;

}

public String getLastName() {

return lastName;

}

public String getEmail() {

return email;

}

}

class Review {

private int rating;

private String comment;

public Review(int rating, String comment) {

this.rating = rating;

this.comment = comment;

}

public int getRating() {

return rating;

}

public String getComment() {

return comment;

}

}

class CartItem {

private Product product;

private int quantity;

public CartItem(Product product, int quantity) {

this.product = product;

this.quantity = quantity;

}

public Product getProduct() {

return product;

}

public int getQuantity() {

return quantity;

}

}

class ShoppingCart {

private int cartId;

private Customer customer;

private List<CartItem> cartItems;

public ShoppingCart(int cartId, Customer customer) {

this.cartId = cartId;

this.customer = customer;

this.cartItems = new ArrayList<>();

}

public void addItem(Product product, int quantity) {

if (product.getQuantityInStock() >= quantity) {

cartItems.add(new CartItem(product, quantity));

product.setQuantityInStock(product.getQuantityInStock() - quantity);

} else {

System.out.println("Insufficient quantity in stock for " + product.getProductName());

}

}

public void removeItem(Product product) {

cartItems.removeIf(cartItem -> cartItem.getProduct() == product);

}

public double calculateTotal() {

double total = 0.0;

for (CartItem cartItem : cartItems) {

total += cartItem.getProduct().getPrice() \* cartItem.getQuantity();

}

return total;

}

public void displayCart() {

System.out.println("Shopping Cart for " + customer.getFirstName() + " " + customer.getLastName());

for (CartItem cartItem : cartItems) {

Product product = cartItem.getProduct();

System.out.println("Product: " + product.getProductName() + ", Quantity: " + cartItem.getQuantity() + ", Price: $" + product.getPrice());

}

System.out.println("Total: $" + calculateTotal());

}

}

public class StartShopping{

public static void main(String[] args) {

Customer customer1 = new Customer(1, "John", "Doe", "john@example.com");

Customer customer2 = new Customer(2, "Jane", "Smith", "jane@example.com");

Product product1 = new Product(101, "Laptop", 999.99, 10);

Product product2 = new Product(102, "Phone", 399.99, 20);

Product product3 = new Product(103, "Tablet", 199.99, 15);

ShoppingCart cart1 = new ShoppingCart(1, customer1);

ShoppingCart cart2 = new ShoppingCart(2, customer2);

cart1.addItem(product1, 2);

cart1.addItem(product2, 1);

cart2.addItem(product2, 3);

cart2.addItem(product3, 2);

cart1.displayCart();

cart2.displayCart();

}

}

Sol.4:

import java.util.\*;

class Person {

private int id;

private String firstName;

private String lastName;

private String email;

public Person(int id, String firstName, String lastName, String email) {

this.id = id;

this.firstName = firstName;

this.lastName = lastName;

this.email = email;

}

public int getId() {

return id;

}

public String getFirstName() {

return firstName;

}

public String getLastName() {

return lastName;

}

public String getEmail() {

return email;

}

}

class Student extends Person {

private int studentId;

private List<Course> enrolledCourses;

public Student(int id, String firstName, String lastName, String email, int studentId) {

super(id, firstName, lastName, email);

this.studentId = studentId;

this.enrolledCourses = new ArrayList<>();

}

public void enrollCourse(Course course) {

enrolledCourses.add(course);

course.enrollStudent(this);

}

public void dropCourse(Course course) {

enrolledCourses.remove(course);

course.dropStudent(this);

}

public List<Course> getEnrolledCourses() {

return enrolledCourses;

}

}

class Instructor extends Person {

private int instructorId;

private List<Course> coursesTaught;

public Instructor(int id, String firstName, String lastName, String email, int instructorId) {

super(id, firstName, lastName, email);

this.instructorId = instructorId;

this.coursesTaught = new ArrayList<>();

}

public void assignCourse(Course course) {

coursesTaught.add(course);

course.setInstructor(this);

}

public List<Course> getAssignedCourses() {

return coursesTaught;

}

}

class Course {

private int courseId;

private String courseName;

private Instructor instructor;

private List<Student> studentsEnrolled;

public Course(int courseId, String courseName) {

this.courseId = courseId;

this.courseName = courseName;

this.studentsEnrolled = new ArrayList<>();

}

public void enrollStudent(Student student) {

studentsEnrolled.add(student);

}

public void dropStudent(Student student) {

studentsEnrolled.remove(student);

}

public List<Student> getEnrolledStudents() {

return studentsEnrolled;

}

public void setInstructor(Instructor instructor) {

this.instructor = instructor;

}

public Instructor getInstructor() {

return instructor;

}

public int getCourseId() {

return courseId;

}

public String getCourseName() {

return courseName;

}

}

public class RunUniversity{

public static void main(String[] args) {

Student student1 = new Student(1, "John", "Doe", "john@example.com", 101);

Student student2 = new Student(2, "Jane", "Smith", "jane@example.com", 102);

Instructor instructor1 = new Instructor(101, "Professor", "Johnson", "prof.johnson@example.com", 201);

Instructor instructor2 = new Instructor(102, "Professor", "Smith", "prof.smith@example.com", 202);

Course course1 = new Course(1001, "Math 101");

Course course2 = new Course(1002, "History 101");

student1.enrollCourse(course1);

student1.enrollCourse(course2);

student2.enrollCourse(course1);

instructor1.assignCourse(course1);

instructor2.assignCourse(course2);

displayCourseInfo(course1);

displayCourseInfo(course2);

}

public static void displayCourseInfo(Course course) {

System.out.println("Course ID: " + course.getCourseId());

System.out.println("Course Name: " + course.getCourseName());

Instructor instructor = course.getInstructor();

System.out.println("Instructor: " + instructor.getFirstName() + " " + instructor.getLastName());

System.out.println("Enrolled Students:");

List<Student> students = course.getEnrolledStudents();

for (Student student : students) {

System.out.println(student.getFirstName() + " " + student.getLastName());

}

}

}