LAB PROBLEM 1: Class Diagram for Library System  
  
class Book {

String ISBN;

String title;

String author;

boolean isAvailable;

Book(String ISBN, String title, String author) {

this.ISBN = ISBN;

this.title = title;

this.author = author;

this.isAvailable = true;

}

void borrowBook() {

isAvailable = false;

System.out.println(title + " has been borrowed");

}

void returnBook() {

isAvailable = true;

System.out.println(title + " has been returned");

}

}

class Member {

String memberId;

String name;

String email;

Book[] borrowedBooks;

int bookCount;

Member(String memberId, String name, String email) {

this.memberId = memberId;

this.name = name;

this.email = email;

this.borrowedBooks = new Book[5];

this.bookCount = 0;

}

void borrowBook(Book book) {

if(book.isAvailable && bookCount < 5) {

borrowedBooks[bookCount] = book;

bookCount++;

book.borrowBook();

}

}

void returnBook(Book book) {

book.returnBook();

}

}

class Librarian {

String employeeId;

String name;

Librarian(String employeeId, String name) {

this.employeeId = employeeId;

this.name = name;

}

void issueBook(Member member, Book book) {

System.out.println("Librarian " + name + " issuing book to " + member.name);

member.borrowBook(book);

}

void receiveBook(Member member, Book book) {

System.out.println("Librarian " + name + " receiving book from " + member.name);

member.returnBook(book);

}

}

class Library {

String name;

String address;

Book[] books;

Member[] members;

int bookCount;

int memberCount;

Library(String name, String address) {

this.name = name;

this.address = address;

this.books = new Book[100];

this.members = new Member[50];

this.bookCount = 0;

this.memberCount = 0;

}

void addBook(Book book) {

books[bookCount] = book;

bookCount++;

System.out.println("Book added: " + book.title);

}

void registerMember(Member member) {

members[memberCount] = member;

memberCount++;

System.out.println("Member registered: " + member.name);

}

}

public class Main {

public static void main(String[] args) {

Library library = new Library("City Library", "123 Main Street");

Book book1 = new Book("ISBN001", "Java Programming", "John Doe");

Book book2 = new Book("ISBN002", "Data Structures", "Jane Smith");

Book book3 = new Book("ISBN003", "Algorithms", "Bob Wilson");

library.addBook(book1);

library.addBook(book2);

library.addBook(book3);

Member member1 = new Member("M001", "Rahul", "rahul@email.com");

Member member2 = new Member("M002", "Priya", "priya@email.com");

library.registerMember(member1);

library.registerMember(member2);

Librarian librarian = new Librarian("L001", "Karthik");

System.out.println("\n--- Borrowing Books ---");

librarian.issueBook(member1, book1);

librarian.issueBook(member1, book2);

librarian.issueBook(member2, book3);

System.out.println("\n--- Returning Books ---");

librarian.receiveBook(member1, book1);

}

}

LAB PROBLEM 2: Object Diagram for Student-Teacher Relationship

class Student {

String name;

String rollNo;

String grade;

Student(String name, String rollNo, String grade) {

this.name = name;

this.rollNo = rollNo;

this.grade = grade;

}

void display() {

System.out.println("Student: " + name + ", Roll: " + rollNo + ", Grade: " + grade);

}

}

class Teacher {

String name;

String subject;

String id;

Student student1;

Student student2;

Teacher(String name, String subject, String id) {

this.name = name;

this.subject = subject;

this.id = id;

}

void guideStudents(Student s1, Student s2) {

this.student1 = s1;

this.student2 = s2;

}

void display() {

System.out.println("\nTeacher: " + name + ", Subject: " + subject + ", ID: " + id);

System.out.println("Guides:");

student1.display();

student2.display();

}

}

public class Main {

public static void main(String[] args) {

Student student1 = new Student("Rahul", "S001", "A");

Student student2 = new Student("Sneha", "S002", "B");

Student student3 = new Student("Amit", "S003", "A");

Student student4 = new Student("Divya", "S004", "B");

Teacher teacher1 = new Teacher("Karthik", "Math", "T101");

Teacher teacher2 = new Teacher("Priya", "Physics", "T102");

teacher1.guideStudents(student1, student2);

teacher2.guideStudents(student3, student4);

teacher1.display();

teacher2.display();

}

}

LAB PROBLEM 3: Sequence Diagram for Online Order Process

class InventoryService {

boolean checkStock(String productId, int quantity) {

System.out.println("Checking stock for product: " + productId);

return true;

}

boolean updateStock(String productId, int quantity) {

System.out.println("Updating stock for product: " + productId);

return true;

}

}

class PaymentGateway {

String processPayment(double amount, String cardDetails) {

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

return "TXN" + System.currentTimeMillis();

}

}

class OrderService {

InventoryService inventoryService;

PaymentGateway paymentGateway;

OrderService() {

inventoryService = new InventoryService();

paymentGateway = new PaymentGateway();

}

String placeOrder(String productId, int quantity, double amount, String cardDetails) {

System.out.println("\n--- Starting Order Process ---");

boolean stockAvailable = inventoryService.checkStock(productId, quantity);

if(stockAvailable) {

String transactionId = paymentGateway.processPayment(amount, cardDetails);

if(transactionId != null) {

inventoryService.updateStock(productId, quantity);

String orderId = "ORD" + System.currentTimeMillis();

System.out.println("Order confirmed: " + orderId);

return orderId;

}

}

return null;

}

}

class Customer {

String name;

Customer(String name) {

this.name = name;

}

void buyProduct(OrderService orderService, String productId, int quantity, double amount) {

System.out.println("Customer " + name + " placing order...");

String orderId = orderService.placeOrder(productId, quantity, amount, "1234-5678-9012");

if(orderId != null) {

System.out.println("Customer received order confirmation: " + orderId);

}

}

}

public class Main {

public static void main(String[] args) {

Customer customer = new Customer("Rahul");

OrderService orderService = new OrderService();

customer.buyProduct(orderService, "P001", 2, 1500.00);

}

}

LAB PROBLEM 5: Activity Diagram for Student Registration Process

class RegistrationForm {

String name;

String email;

String phone;

String course;

RegistrationForm(String name, String email, String phone, String course) {

this.name = name;

this.email = email;

this.phone = phone;

this.course = course;

}

boolean validateData() {

if(name == null || name.isEmpty()) {

System.out.println("Error: Name is required");

return false;

}

if(email == null || !email.contains("@")) {

System.out.println("Error: Valid email is required");

return false;

}

if(phone == null || phone.length() != 10) {

System.out.println("Error: Valid 10-digit phone is required");

return false;

}

System.out.println("Data validation successful");

return true;

}

}

class Admin {

String adminName;

Admin(String adminName) {

this.adminName = adminName;

}

boolean reviewApplication(RegistrationForm form) {

System.out.println("\nAdmin " + adminName + " reviewing application for " + form.name);

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

return true;

}

}

class RegistrationSystem {

Admin admin;

int studentIdCounter = 1000;

RegistrationSystem(Admin admin) {

this.admin = admin;

}

void processRegistration(RegistrationForm form) {

System.out.println("=== Student Registration Process ===");

System.out.println("Form filled by: " + form.name);

System.out.println("Submitting form...");

if(!form.validateData()) {

System.out.println("\nRegistration Failed: Invalid Data");

return;

}

System.out.println("Sending to admin for review...");

boolean approved = admin.reviewApplication(form);

if(!approved) {

sendRejectionEmail(form.email);

System.out.println("\nRegistration Rejected");

} else {

String studentId = generateStudentId();

sendConfirmationEmail(form.email, studentId);

System.out.println("\nRegistration Successful!");

System.out.println("Student ID: " + studentId);

}

}

String generateStudentId() {

studentIdCounter++;

return "STU" + studentIdCounter;

}

void sendConfirmationEmail(String email, String studentId) {

System.out.println("Confirmation email sent to: " + email);

System.out.println("Your Student ID: " + studentId);

}

void sendRejectionEmail(String email) {

System.out.println("Rejection email sent to: " + email);

}

}

public class Main {

public static void main(String[] args) {

Admin admin = new Admin("Dr. Sharma");

RegistrationSystem system = new RegistrationSystem(admin);

RegistrationForm form1 = new RegistrationForm("Rahul Kumar", "rahul@email.com", "9876543210", "Computer Science");

system.processRegistration(form1);

System.out.println("\n\n");

RegistrationForm form2 = new RegistrationForm("Priya", "invalid-email", "123", "Mathematics");

system.processRegistration(form2);

}

}