**WEEK 3 – HW**

Q1. Personal Finance Manager

Soln.

public class PersonalFinanceManager {

static class PersonalAccount {

private String accountHolderName;

private String accountNumber;

private double currentBalance;

private double totalIncome;

private double totalExpenses;

private static int totalAccounts = 0;

private static String bankName = "Default Bank";

public PersonalAccount(String name, double initialDeposit) {

this.accountHolderName = name;

this.accountNumber = generateAccountNumber();

this.currentBalance = initialDeposit;

this.totalIncome = initialDeposit;

this.totalExpenses = 0;

totalAccounts++;

}

public void addIncome(double amount, String description) {

if (amount <= 0) {

System.out.println("Invalid income amount");

return;

}

currentBalance += amount;

totalIncome += amount;

System.out.println(accountHolderName + " received income: Rs." + amount + " (" + description + ")");

}

public void addExpense(double amount, String description) {

if (amount <= 0) {

System.out.println("Invalid expense amount");

return;

}

if (amount > currentBalance) {

System.out.println("Insufficient balance for expense: " + description);

return;

}

currentBalance -= amount;

totalExpenses += amount;

System.out.println(accountHolderName + " spent: Rs." + amount + " (" + description + ")");

}

public double calculateSavings() {

return totalIncome - totalExpenses;

}

public void displayAccountSummary() {

System.out.println("===== Account Summary =====");

System.out.println("Bank Name : " + bankName);

System.out.println("Account Holder: " + accountHolderName);

System.out.println("Account Number: " + accountNumber);

System.out.println("Current Balance: Rs." + currentBalance);

System.out.println("Total Income : Rs." + totalIncome);

System.out.println("Total Expenses: Rs." + totalExpenses);

System.out.println("Total Savings : Rs." + calculateSavings());

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

}

public static void setBankName(String name) {

bankName = name;

}

public static int getTotalAccounts() {

return totalAccounts;

}

private static String generateAccountNumber() {

return String.format("AC%03d", totalAccounts + 1);

}

}

public static void main(String[] args) {

PersonalAccount.setBankName("WealthCare Bank");

PersonalAccount acc1 = new PersonalAccount("Alice", 5000);

PersonalAccount acc2 = new PersonalAccount("Bob", 3000);

PersonalAccount acc3 = new PersonalAccount("Charlie", 7000);

acc1.addIncome(2000, "Salary");

acc1.addExpense(1500, "Groceries");

acc1.addExpense(500, "Transport");

acc2.addIncome(4000, "Freelance Work");

acc2.addExpense(2500, "Rent");

acc3.addExpense(2000, "Shopping");

acc3.addIncome(5000, "Bonus");

acc1.displayAccountSummary();

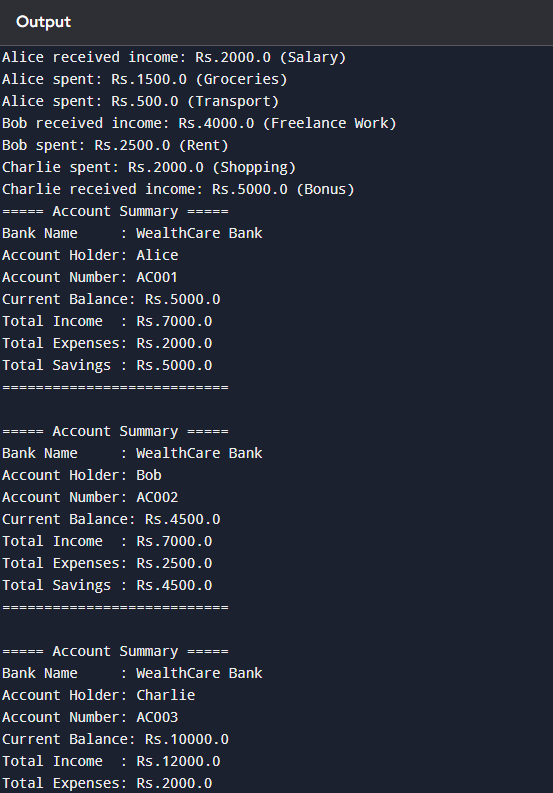
acc2.displayAccountSummary();

acc3.displayAccountSummary();

System.out.println("Total Accounts Created: " + PersonalAccount.getTotalAccounts());

}

}



Q2. Online Shopping Cart System

Soln.

import java.util.\*;

public class ShoppingCartSystem {

static class Product {

private String productId;

private String productName;

private double price;

private String category;

private int stockQuantity;

private static int totalProducts = 0;

private static String[] categories = {"Electronics", "Clothing", "Books", "Home", "Food"};

public Product(String name, double price, String category, int stock) {

this.productId = generateProductId();

this.productName = name;

this.price = price;

this.category = category;

this.stockQuantity = stock;

totalProducts++;

}

public String getProductId() { return productId; }

public String getProductName() { return productName; }

public double getPrice() { return price; }

public String getCategory() { return category; }

public int getStockQuantity() { return stockQuantity; }

public void reduceStock(int qty) { stockQuantity -= qty; }

public void increaseStock(int qty) { stockQuantity += qty; }

private static String generateProductId() {

return String.format("P%03d", totalProducts + 1);

}

public static Product findProductById(Product[] products, String productId) {

for (Product p : products) {

if (p != null && p.getProductId().equals(productId)) return p;

}

return null;

}

public static Product[] getProductsByCategory(Product[] products, String category) {

int count = 0;

for (Product p : products) if (p != null && p.getCategory().equalsIgnoreCase(category)) count++;

Product[] result = new Product[count];

int idx = 0;

for (Product p : products) {

if (p != null && p.getCategory().equalsIgnoreCase(category)) {

result[idx++] = p;

}

}

return result;

}

public void displayProduct() {

System.out.println(productId + " | " + productName + " | Rs." + price + " | " + category + " | Stock: " + stockQuantity);

}

}

static class ShoppingCart {

private String cartId;

private String customerName;

private Product[] products;

private int[] quantities;

private int itemCount;

private double cartTotal;

public ShoppingCart(String customerName) {

this.cartId = UUID.randomUUID().toString().substring(0,6);

this.customerName = customerName;

this.products = new Product[20];

this.quantities = new int[20];

this.itemCount = 0;

this.cartTotal = 0;

}

public void addProduct(Product product, int quantity) {

if (product == null) {

System.out.println("Invalid product.");

return;

}

if (quantity > product.getStockQuantity()) {

System.out.println("Not enough stock available.");

return;

}

for (int i = 0; i < itemCount; i++) {

if (products[i].getProductId().equals(product.getProductId())) {

quantities[i] += quantity;

product.reduceStock(quantity);

calculateTotal();

System.out.println(quantity + " more " + product.getProductName() + " added to cart.");

return;

}

}

products[itemCount] = product;

quantities[itemCount] = quantity;

product.reduceStock(quantity);

itemCount++;

calculateTotal();

System.out.println(product.getProductName() + " added to cart.");

}

public void removeProduct(String productId) {

for (int i = 0; i < itemCount; i++) {

if (products[i].getProductId().equals(productId)) {

products[i].increaseStock(quantities[i]);

System.out.println(products[i].getProductName() + " removed from cart.");

for (int j = i; j < itemCount - 1; j++) {

products[j] = products[j + 1];

quantities[j] = quantities[j + 1];

}

products[itemCount - 1] = null;

quantities[itemCount - 1] = 0;

itemCount--;

calculateTotal();

return;

}

}

System.out.println("Product not found in cart.");

}

public void calculateTotal() {

cartTotal = 0;

for (int i = 0; i < itemCount; i++) {

cartTotal += products[i].getPrice() \* quantities[i];

}

}

public void displayCart() {

System.out.println("===== Shopping Cart (" + customerName + ") =====");

if (itemCount == 0) {

System.out.println("Cart is empty.");

} else {

for (int i = 0; i < itemCount; i++) {

System.out.println(products[i].getProductId() + " | " + products[i].getProductName() +

" | Qty: " + quantities[i] + " | Rs." + (products[i].getPrice() \* quantities[i]));

}

System.out.println("Cart Total: Rs." + cartTotal);

}

System.out.println("==============================");

}

public void checkout() {

if (itemCount == 0) {

System.out.println("Cart is empty. Cannot checkout.");

return;

}

displayCart();

System.out.println("Checkout successful! Amount paid: Rs." + cartTotal);

products = new Product[20];

quantities = new int[20];

itemCount = 0;

cartTotal = 0;

}

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

Product[] products = {

new Product("Laptop", 60000, "Electronics", 10),

new Product("Phone", 30000, "Electronics", 15),

new Product("Shirt", 1500, "Clothing", 20),

new Product("Jeans", 2500, "Clothing", 25),

new Product("Novel", 500, "Books", 30),

new Product("Cookbook", 800, "Books", 20),

new Product("Sofa", 20000, "Home", 5),

new Product("Chair", 3000, "Home", 12),

new Product("Rice Pack", 1200, "Food", 50),

new Product("Snacks", 200, "Food", 100)

};

System.out.print("Enter your name: ");

String name = sc.nextLine();

ShoppingCart cart = new ShoppingCart(name);

while (true) {

System.out.println("\n=== Online Shopping Menu ===");

System.out.println("1. View All Products");

System.out.println("2. Search by Category");

System.out.println("3. Add to Cart");

System.out.println("4. Remove from Cart");

System.out.println("5. View Cart");

System.out.println("6. Checkout");

System.out.println("7. Exit");

System.out.print("Choose an option: ");

int choice = sc.nextInt();

sc.nextLine();

switch (choice) {

case 1:

System.out.println("=== Product List ===");

for (Product p : products) p.displayProduct();

break;

case 2:

System.out.print("Enter category: ");

String cat = sc.nextLine();

Product[] result = Product.getProductsByCategory(products, cat);

if (result.length == 0) {

System.out.println("No products found in this category.");

} else {

for (Product p : result) p.displayProduct();

}

break;

case 3:

System.out.print("Enter product ID to add: ");

String pid = sc.nextLine();

System.out.print("Enter quantity: ");

int qty = sc.nextInt();

sc.nextLine();

Product toAdd = Product.findProductById(products, pid);

cart.addProduct(toAdd, qty);

break;

case 4:

System.out.print("Enter product ID to remove: ");

String rid = sc.nextLine();

cart.removeProduct(rid);

break;

case 5:

cart.displayCart();

break;

case 6:

cart.checkout();

break;

case 7:

System.out.println("Exiting... Thank you for shopping!");

sc.close();

return;

default:

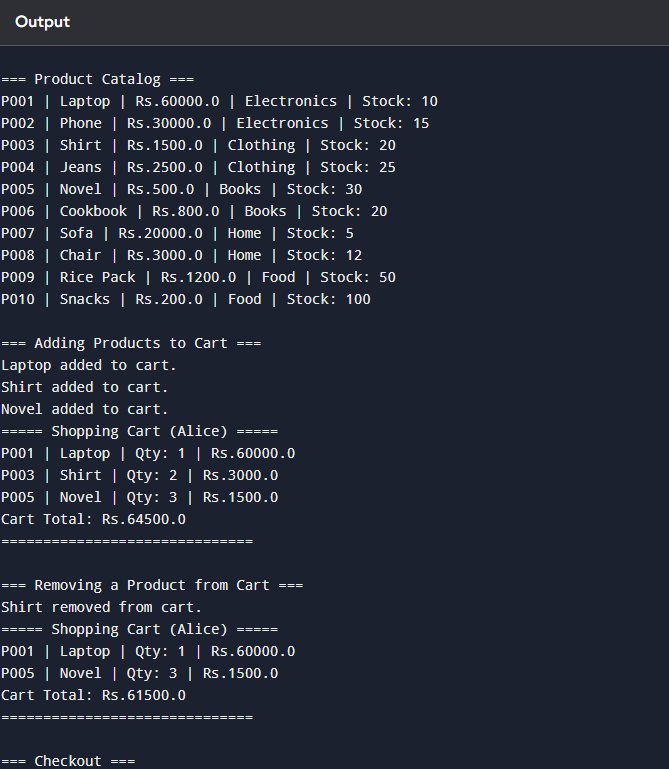
System.out.println("Invalid choice.");

}

}

}

}



Q3. Hotel Reservation System

Soln.

public class HotelReservationSystem {

static class Room {

private String roomNumber;

private String roomType;

private double pricePerNight;

private boolean isAvailable;

private int maxOccupancy;

private int bookingCount; // track popularity

public Room(String roomNumber, String roomType, double pricePerNight, int maxOccupancy) {

this.roomNumber = roomNumber;

this.roomType = roomType;

this.pricePerNight = pricePerNight;

this.isAvailable = true;

this.maxOccupancy = maxOccupancy;

this.bookingCount = 0;

}

public String getRoomNumber() { return roomNumber; }

public String getRoomType() { return roomType; }

public double getPricePerNight() { return pricePerNight; }

public boolean isAvailable() { return isAvailable; }

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

public int getMaxOccupancy() { return maxOccupancy; }

public void incrementBookingCount() { bookingCount++; }

public int getBookingCount() { return bookingCount; }

public void displayRoomInfo() {

System.out.println(roomNumber + " | " + roomType + " | Rs." + pricePerNight + " | Occupancy: " + maxOccupancy + " | Available: " + isAvailable);

}

}

static class Guest {

private String guestId;

private String guestName;

private String phoneNumber;

private String email;

private String[] bookingHistory;

private int bookingCount;

private static int guestCounter = 0;

public Guest(String name, String phone, String email) {

this.guestId = generateGuestId();

this.guestName = name;

this.phoneNumber = phone;

this.email = email;

this.bookingHistory = new String[10];

this.bookingCount = 0;

}

private static String generateGuestId() {

guestCounter++;

return "G" + String.format("%03d", guestCounter);

}

public String getGuestId() { return guestId; }

public String getGuestName() { return guestName; }

public void addBooking(String bookingId) {

if (bookingCount < bookingHistory.length) {

bookingHistory[bookingCount++] = bookingId;

}

}

public void displayGuestInfo() {

System.out.println("Guest: " + guestName + " (" + guestId + "), Phone: " + phoneNumber + ", Email: " + email);

}

}

static class Booking {

private String bookingId;

private Guest guest;

private Room room;

private String checkInDate;

private String checkOutDate;

private double totalAmount;

private static int totalBookings = 0;

private static double hotelRevenue = 0;

private static String hotelName = "Grand Palace";

public Booking(Guest guest, Room room, String checkInDate, String checkOutDate, int nights) {

this.bookingId = generateBookingId();

this.guest = guest;

this.room = room;

this.checkInDate = checkInDate;

this.checkOutDate = checkOutDate;

this.totalAmount = calculateBill(nights);

totalBookings++;

hotelRevenue += totalAmount;

room.setAvailable(false);

room.incrementBookingCount();

guest.addBooking(bookingId);

}

private static String generateBookingId() {

return "B" + String.format("%03d", totalBookings + 1);

}

public double calculateBill(int nights) {

return room.getPricePerNight() \* nights;

}

public void cancelReservation() {

room.setAvailable(true);

hotelRevenue -= totalAmount;

System.out.println("Booking " + bookingId + " canceled.");

}

public void displayBookingInfo() {

System.out.println("Booking ID: " + bookingId + " | Guest: " + guest.getGuestName() +

" | Room: " + room.getRoomNumber() + " (" + room.getRoomType() + ")" +

" | Check-In: " + checkInDate + " | Check-Out: " + checkOutDate +

" | Amount: Rs." + totalAmount);

}

public static double getTotalRevenue() { return hotelRevenue; }

public static double getOccupancyRate(Room[] rooms) {

int occupied = 0;

for (Room r : rooms) if (!r.isAvailable()) occupied++;

return (occupied \* 100.0) / rooms.length;

}

public static String getMostPopularRoomType(Room[] rooms) {

Room popular = rooms[0];

for (Room r : rooms) {

if (r.getBookingCount() > popular.getBookingCount()) {

popular = r;

}

}

return popular.getRoomType();

}

public static void displayHotelStats(Room[] rooms) {

System.out.println("\n=== Hotel Report (" + hotelName + ") ===");

System.out.println("Total Bookings : " + totalBookings);

System.out.println("Total Revenue : Rs." + hotelRevenue);

System.out.println("Occupancy Rate : " + getOccupancyRate(rooms) + "%");

System.out.println("Most Popular Room Type: " + getMostPopularRoomType(rooms));

System.out.println("==============================");

}

}

public static void main(String[] args) {

Room[] rooms = {

new Room("101", "Single", 2000, 1),

new Room("102", "Double", 3500, 2),

new Room("103", "Suite", 7000, 4),

new Room("104", "Double", 3500, 2),

new Room("105", "Single", 2000, 1)

};

Guest g1 = new Guest("Alice", "9876543210", "alice@mail.com");

Guest g2 = new Guest("Bob", "9123456789", "bob@mail.com");

System.out.println("=== Available Rooms Initially ===");

for (Room r : rooms) r.displayRoomInfo();

Booking b1 = new Booking(g1, rooms[1], "2025-09-20", "2025-09-22", 2);

Booking b2 = new Booking(g2, rooms[2], "2025-09-21", "2025-09-24", 3);

System.out.println("\n=== Bookings Made ===");

b1.displayBookingInfo();

b2.displayBookingInfo();

System.out.println("\n=== Available Rooms After Booking ===");

for (Room r : rooms) r.displayRoomInfo();

b1.cancelReservation();

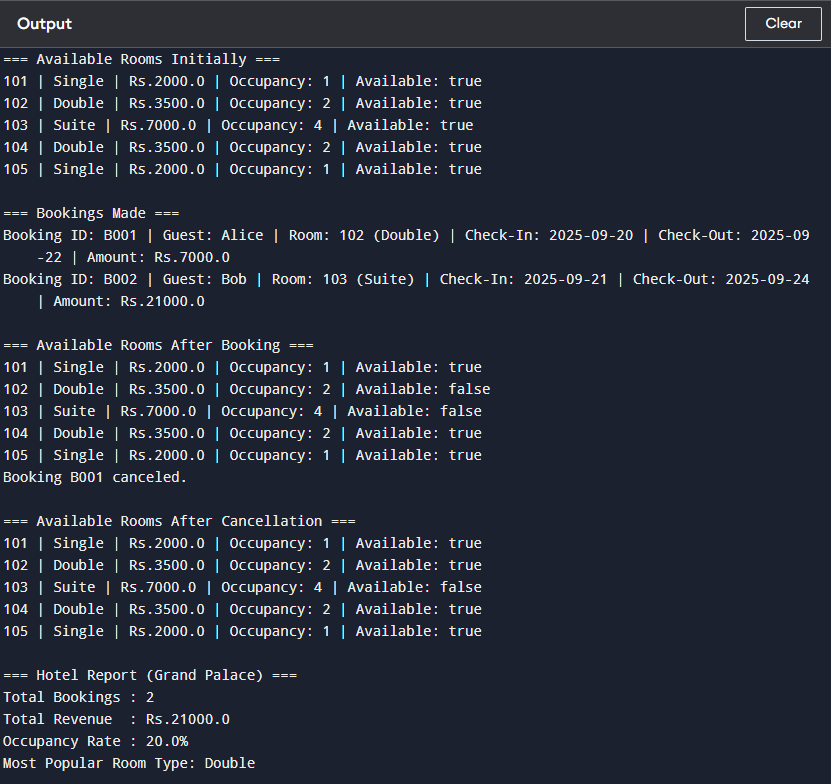
System.out.println("\n=== Available Rooms After Cancellation ===");

for (Room r : rooms) r.displayRoomInfo();

Booking.displayHotelStats(rooms);

}

}



Q4. Student Grade Management System

Soln.

public class StudentGradeSystem {

static class Subject {

private String subjectCode;

private String subjectName;

private int credits;

private String instructor;

public Subject(String subjectCode, String subjectName, int credits, String instructor) {

this.subjectCode = subjectCode;

this.subjectName = subjectName;

this.credits = credits;

this.instructor = instructor;

}

public String getSubjectCode() { return subjectCode; }

public String getSubjectName() { return subjectName; }

public int getCredits() { return credits; }

public String getInstructor() { return instructor; }

}

static class Student {

private String studentId;

private String studentName;

private String className;

private Subject[] subjects;

private double[] marks;

private double gpa;

private static int totalStudents = 0;

private static String schoolName = "Global High School";

private static String[] gradingScale = {"A", "B", "C", "D", "F"};

private static double passPercentage = 40.0;

public Student(String studentName, String className, Subject[] subjects) {

this.studentId = generateStudentId();

this.studentName = studentName;

this.className = className;

this.subjects = subjects;

this.marks = new double[subjects.length];

this.gpa = 0.0;

totalStudents++;

}

public void addMarks(String subjectCode, double score) {

for (int i = 0; i < subjects.length; i++) {

if (subjects[i].getSubjectCode().equals(subjectCode)) {

marks[i] = score;

}

}

}

public void calculateGPA() {

double total = 0;

for (int i = 0; i < marks.length; i++) {

total += marks[i];

}

double avg = total / marks.length;

if (avg >= 90) gpa = 4.0;

else if (avg >= 80) gpa = 3.5;

else if (avg >= 70) gpa = 3.0;

else if (avg >= 60) gpa = 2.5;

else if (avg >= 50) gpa = 2.0;

else gpa = 1.0;

}

public boolean checkPromotionEligibility() {

for (double m : marks) {

if (m < passPercentage) return false;

}

return true;

}

public void generateReportCard() {

calculateGPA();

System.out.println("===== Report Card =====");

System.out.println("School : " + schoolName);

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

System.out.println("Class : " + className);

for (int i = 0; i < subjects.length; i++) {

System.out.println(subjects[i].getSubjectName() + " (" + subjects[i].getSubjectCode() + "): " + marks[i] + " | Grade: " + getGrade(marks[i]));

}

System.out.println("GPA : " + gpa);

System.out.println("Promotion : " + (checkPromotionEligibility() ? "Eligible" : "Not Eligible"));

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

}

private String getGrade(double score) {

if (score >= 85) return "A";

else if (score >= 70) return "B";

else if (score >= 55) return "C";

else if (score >= 40) return "D";

else return "F";

}

public static void setGradingScale(String[] newScale) {

gradingScale = newScale;

}

public static double calculateClassAverage(Student[] students) {

double total = 0;

int count = 0;

for (Student s : students) {

for (double m : s.marks) {

total += m;

count++;

}

}

return count == 0 ? 0 : total / count;

}

public static Student[] getTopPerformers(Student[] students, int count) {

Student[] sorted = students.clone();

for (int i = 0; i < sorted.length - 1; i++) {

for (int j = i + 1; j < sorted.length; j++) {

if (sorted[j].gpa > sorted[i].gpa) {

Student temp = sorted[i];

sorted[i] = sorted[j];

sorted[j] = temp;

}

}

}

int resultCount = Math.min(count, sorted.length);

Student[] result = new Student[resultCount];

for (int i = 0; i < resultCount; i++) {

result[i] = sorted[i];

}

return result;

}

public static void generateSchoolReport(Student[] students) {

System.out.println("\n=== School Report (" + schoolName + ") ===");

System.out.println("Total Students : " + totalStudents);

System.out.println("Class Average : " + calculateClassAverage(students));

Student[] toppers = getTopPerformers(students, 3);

System.out.println("Top Performers : ");

for (Student s : toppers) {

System.out.println(" - " + s.studentName + " (GPA: " + s.gpa + ")");

}

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

}

private static String generateStudentId() {

return "S" + String.format("%03d", totalStudents + 1);

}

}

public static void main(String[] args) {

Subject[] subjects = {

new Subject("MATH101", "Mathematics", 4, "Mr. Sharma"),

new Subject("ENG101", "English", 3, "Ms. Gupta"),

new Subject("SCI101", "Science", 4, "Dr. Rao")

};

Student s1 = new Student("Alice", "10A", subjects);

Student s2 = new Student("Bob", "10A", subjects);

Student s3 = new Student("Charlie", "10A", subjects);

s1.addMarks("MATH101", 95);

s1.addMarks("ENG101", 88);

s1.addMarks("SCI101", 92);

s2.addMarks("MATH101", 75);

s2.addMarks("ENG101", 60);

s2.addMarks("SCI101", 70);

s3.addMarks("MATH101", 45);

s3.addMarks("ENG101", 55);

s3.addMarks("SCI101", 35);

s1.generateReportCard();

s2.generateReportCard();

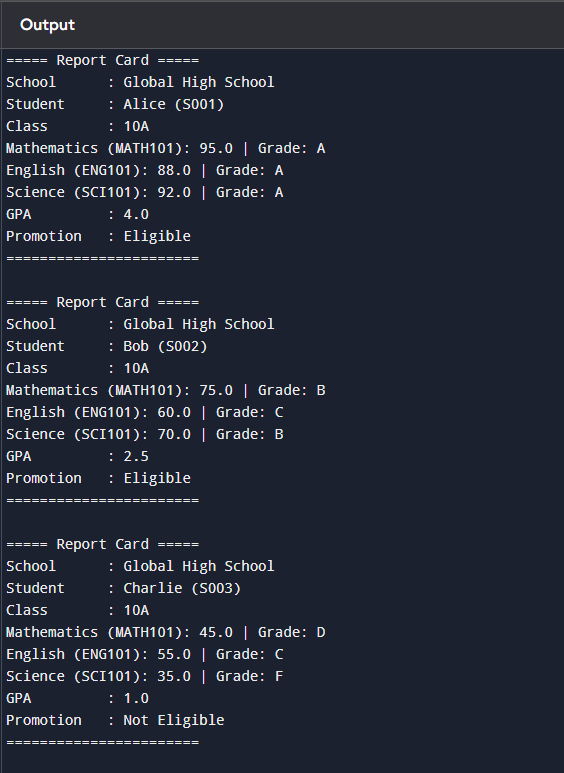
s3.generateReportCard();

Student[] students = {s1, s2, s3};

Student.generateSchoolReport(students);

}

}



Q5. Library Management System with Fine Calculation

Soln.

import java.util.\*;

public class LibraryManagementSystem {

static class Book {

private String bookId;

private String title;

private String author;

private String isbn;

private String category;

private boolean isIssued;

private String issueDate;

private String dueDate;

private int timesIssued;

private static int totalBooks = 0;

public Book(String title, String author, String isbn, String category) {

this.bookId = generateBookId();

this.title = title;

this.author = author;

this.isbn = isbn;

this.category = category;

this.isIssued = false;

this.issueDate = null;

this.dueDate = null;

this.timesIssued = 0;

totalBooks++;

}

private static String generateBookId() {

return "B" + String.format("%03d", totalBooks + 1);

}

public String getBookId() { return bookId; }

public String getTitle() { return title; }

public String getCategory() { return category; }

public boolean isIssued() { return isIssued; }

public int getTimesIssued() { return timesIssued; }

public void issueBook(String issueDate, String dueDate) {

isIssued = true;

this.issueDate = issueDate;

this.dueDate = dueDate;

timesIssued++;

}

public void returnBook() {

isIssued = false;

this.issueDate = null;

this.dueDate = null;

}

public void renewBook(String newDueDate) {

this.dueDate = newDueDate;

}

public void displayBook() {

System.out.println(bookId + " | " + title + " | " + author + " | " + category + " | Issued: " + isIssued);

}

}

static class Member {

private String memberId;

private String memberName;

private String memberType;

private Book[] booksIssued;

private int issuedCount;

private double totalFines;

private String membershipDate;

private static int totalMembers = 0;

private static String libraryName = "City Central Library";

private static double finePerDay = 5.0;

private static int maxBooksAllowed = 3;

public Member(String name, String type, String membershipDate) {

this.memberId = generateMemberId();

this.memberName = name;

this.memberType = type;

this.membershipDate = membershipDate;

this.booksIssued = new Book[maxBooksAllowed];

this.issuedCount = 0;

this.totalFines = 0;

totalMembers++;

}

private static String generateMemberId() {

return "M" + String.format("%03d", totalMembers + 1);

}

public String getMemberId() { return memberId; }

public String getMemberName() { return memberName; }

public double getTotalFines() { return totalFines; }

public void issueBook(Book book, String issueDate, String dueDate) {

if (issuedCount >= maxBooksAllowed) {

System.out.println(memberName + " cannot issue more than " + maxBooksAllowed + " books.");

return;

}

if (book.isIssued()) {

System.out.println(book.getTitle() + " is already issued.");

return;

}

booksIssued[issuedCount++] = book;

book.issueBook(issueDate, dueDate);

System.out.println(memberName + " issued " + book.getTitle());

}

public void returnBook(Book book, int overdueDays) {

for (int i = 0; i < issuedCount; i++) {

if (booksIssued[i].getBookId().equals(book.getBookId())) {

book.returnBook();

if (overdueDays > 0) {

double fine = overdueDays \* finePerDay;

totalFines += fine;

System.out.println(memberName + " returned " + book.getTitle() + " with fine Rs." + fine);

} else {

System.out.println(memberName + " returned " + book.getTitle() + " on time.");

}

for (int j = i; j < issuedCount - 1; j++) {

booksIssued[j] = booksIssued[j + 1];

}

booksIssued[issuedCount - 1] = null;

issuedCount--;

return;

}

}

System.out.println("Book not found in " + memberName + "'s issued list.");

}

public void renewBook(Book book, String newDueDate) {

for (int i = 0; i < issuedCount; i++) {

if (booksIssued[i].getBookId().equals(book.getBookId())) {

book.renewBook(newDueDate);

System.out.println(memberName + " renewed " + book.getTitle() + " until " + newDueDate);

return;

}

}

System.out.println("Cannot renew. Book not found in issued list.");

}

public void displayMemberInfo() {

System.out.println("Member: " + memberName + " (" + memberType + "), Total Fines: Rs." + totalFines);

}

public static void generateLibraryReport(Member[] members, Book[] books) {

System.out.println("\n=== Library Report (" + libraryName + ") ===");

System.out.println("Total Books : " + Book.totalBooks);

System.out.println("Total Members : " + totalMembers);

System.out.println("Fine per Day : Rs." + finePerDay);

double totalFinesCollected = 0;

for (Member m : members) {

totalFinesCollected += m.getTotalFines();

}

System.out.println("Total Fines Collected: Rs." + totalFinesCollected);

Book popular = books[0];

for (Book b : books) {

if (b.getTimesIssued() > popular.getTimesIssued()) {

popular = b;

}

}

System.out.println("Most Popular Book: " + popular.getTitle() + " (Issued " + popular.getTimesIssued() + " times)");

System.out.println("Overdue Books:");

for (Book b : books) {

if (b.isIssued()) {

System.out.println(" - " + b.getTitle() + " is overdue.");

}

}

System.out.println("===============================");

}

}

public static void main(String[] args) {

Book[] books = {

new Book("1984", "George Orwell", "ISBN001", "Fiction"),

new Book("Clean Code", "Robert Martin", "ISBN002", "Programming"),

new Book("A Brief History of Time", "Stephen Hawking", "ISBN003", "Science")

};

Member m1 = new Member("Alice", "Student", "2025-01-10");

Member m2 = new Member("Bob", "Faculty", "2025-02-01");

m1.issueBook(books[0], "2025-03-01", "2025-03-10");

m1.issueBook(books[1], "2025-03-01", "2025-03-10");

m2.issueBook(books[2], "2025-03-05", "2025-03-15");

m1.returnBook(books[0], 3);

m2.returnBook(books[2], 0);

m1.renewBook(books[1], "2025-03-20");

System.out.println("\n=== Member Info ===");

m1.displayMemberInfo();

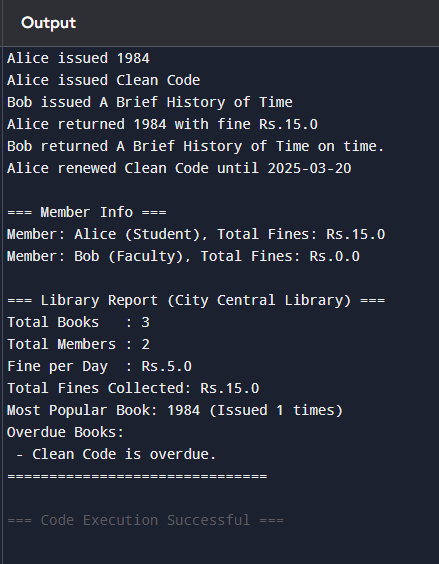
m2.displayMemberInfo();

Member[] members = {m1, m2};

Member.generateLibraryReport(members, books);

}

}



Q6. Employee Payroll and Attendance System

Soln.

public class EmployeePayrollSystem {

static class Employee {

private String empId;

private String empName;

private String department;

private String designation;

private double baseSalary;

private String joinDate;

private boolean[] attendanceRecord;

private String empType; // Full-time, Part-time, Contract

private int leavesTaken;

private static int totalEmployees = 0;

private static String companyName = "Tech Solutions Ltd";

private static double totalSalaryExpense = 0;

private static int workingDaysPerMonth = 30;

public Employee(String empName, String department, String designation, double baseSalary, String joinDate, String empType) {

this.empId = generateEmpId();

this.empName = empName;

this.department = department;

this.designation = designation;

this.baseSalary = baseSalary;

this.joinDate = joinDate;

this.attendanceRecord = new boolean[workingDaysPerMonth];

this.empType = empType;

this.leavesTaken = 0;

totalEmployees++;

}

private static String generateEmpId() {

return "E" + String.format("%03d", totalEmployees + 1);

}

public void markAttendance(int day, boolean present) {

if (day < 1 || day > workingDaysPerMonth) return;

attendanceRecord[day - 1] = present;

if (!present) leavesTaken++;

}

public double calculateSalary() {

int presentDays = 0;

for (boolean b : attendanceRecord) if (b) presentDays++;

double salary = 0;

switch (empType) {

case "Full-time":

salary = baseSalary;

break;

case "Part-time":

salary = (baseSalary / workingDaysPerMonth) \* presentDays;

break;

case "Contract":

salary = baseSalary; // fixed contract salary

break;

}

salary += calculateBonus();

totalSalaryExpense += salary;

return salary;

}

public double calculateBonus() {

int presentDays = 0;

for (boolean b : attendanceRecord) if (b) presentDays++;

double attendanceRate = (presentDays \* 100.0) / workingDaysPerMonth;

if (attendanceRate >= 95) return baseSalary \* 0.2;

else if (attendanceRate >= 85) return baseSalary \* 0.1;

else return 0;

}

public void generatePaySlip() {

double salary = calculateSalary();

System.out.println("===== Pay Slip =====");

System.out.println("Company : " + companyName);

System.out.println("Employee ID : " + empId);

System.out.println("Name : " + empName);

System.out.println("Department : " + department);

System.out.println("Designation : " + designation);

System.out.println("Employee Type: " + empType);

System.out.println("Base Salary : Rs." + baseSalary);

System.out.println("Bonus : Rs." + calculateBonus());

System.out.println("Leaves Taken: " + leavesTaken);

System.out.println("Net Salary : Rs." + salary);

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

}

public void requestLeave(int days) {

leavesTaken += days;

System.out.println(empName + " requested " + days + " leave days. Total leaves: " + leavesTaken);

}

public static double calculateCompanyPayroll(Employee[] employees) {

double total = 0;

for (Employee e : employees) {

total += e.calculateSalary();

}

return total;

}

public static void getAttendanceReport(Employee[] employees) {

System.out.println("\n=== Attendance Report ===");

for (Employee e : employees) {

int presentDays = 0;

for (boolean b : e.attendanceRecord) if (b) presentDays++;

System.out.println(e.empName + " (" + e.empType + "): Present " + presentDays + "/" + workingDaysPerMonth + " days");

}

}

public static int getTotalEmployees() { return totalEmployees; }

}

static class Department {

private String deptId;

private String deptName;

private Employee manager;

private Employee[] employees;

private double budget;

private static int deptCount = 0;

public Department(String deptName, Employee manager, double budget, int capacity) {

this.deptId = generateDeptId();

this.deptName = deptName;

this.manager = manager;

this.budget = budget;

this.employees = new Employee[capacity];

this.employees[0] = manager; // assign manager

}

private static String generateDeptId() {

deptCount++;

return "D" + String.format("%03d", deptCount);

}

public void addEmployee(Employee e, int index) {

if (index >= 0 && index < employees.length) employees[index] = e;

}

public double getDepartmentExpense() {

double total = 0;

for (Employee e : employees) {

if (e != null) total += e.calculateSalary();

}

return total;

}

public void displayDepartmentInfo() {

System.out.println("Department " + deptName + " (" + deptId + ")");

System.out.println("Manager: " + manager.empName);

System.out.println("Budget : Rs." + budget);

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

for (Employee e : employees) {

if (e != null) System.out.println(" - " + e.empName + " (" + e.designation + ")");

}

System.out.println();

}

public static void getDepartmentWiseExpenses(Department[] depts) {

System.out.println("\n=== Department Expenses ===");

for (Department d : depts) {

System.out.println(d.deptName + " Expense: Rs." + d.getDepartmentExpense());

}

}

}

public static void main(String[] args) {

Employee e1 = new Employee("Alice", "IT", "Software Engineer", 50000, "2024-01-01", "Full-time");

Employee e2 = new Employee("Bob", "IT", "Intern", 15000, "2024-02-01", "Part-time");

Employee e3 = new Employee("Charlie", "HR", "Recruiter", 40000, "2023-11-15", "Full-time");

Employee e4 = new Employee("David", "HR", "Consultant", 30000, "2025-01-10", "Contract");

// Mark attendance for simulation (20 days present, 10 absent for Bob, full attendance for Alice)

for (int d = 1; d <= 30; d++) {

e1.markAttendance(d, true);

e2.markAttendance(d, d <= 20);

e3.markAttendance(d, d != 5);

e4.markAttendance(d, d <= 25);

}

Department itDept = new Department("IT", e1, 200000, 3);

itDept.addEmployee(e2, 1);

Department hrDept = new Department("HR", e3, 150000, 3);

hrDept.addEmployee(e4, 1);

e1.generatePaySlip();

e2.generatePaySlip();

e3.generatePaySlip();

e4.generatePaySlip();

Employee[] employees = {e1, e2, e3, e4};

Department[] depts = {itDept, hrDept};

Employee.getAttendanceReport(employees);

Department.getDepartmentWiseExpenses(depts);

double payroll = Employee.calculateCompanyPayroll(employees);

System.out.println("\nTotal Company Payroll: Rs." + payroll);

}

}

