**JAVA PROGRAMMING**

**LAB 1**

**Name:PUNEETH L**

USN: 1BM24MC069

1. **Write a program to manage books, members, and borrowing transactions according to the**

**instructions**

**Classes to Create:**

**• Book (fields: title, author, ISBN, isAvailable)**

**• Member (fields: name, memberId, borrowedBooks[])**

**• Library**

**• Stores list of Book and Member objects**

**Methods:**

**• borrowBook(String isbn, int memberId)**

**• returnBook(String isbn, int memberId)**

**• listAvailableBooks()**

**• listBorrowedBooks(int memberId)**

**Print the necessary values.**

Programm:

class Book {

String title, author, ISBN;

boolean isAvailable;

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

this.title = title;

this.author = author;

this.ISBN = ISBN;

this.isAvailable = isAvailable;

}

}

class Member {

String name;

int memberId;

Book[] borrowedBooks = new Book[3];

Member(String name, int memberId) {

this.name = name;

this.memberId = memberId;

}

}

class Library {

static Book[] books = new Book[3];

static Member[] members = new Member[2];

public static void addBook(Book book) {

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

if (books[i] == null) {

books[i] = book;

return;

}

}

}

public static void addMember(Member member) {

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

if (members[i] == null) {

members[i] = member;

return;

}

}

}

public static void borrowBook(String isbn, int memberId) {

Member member = members[memberId - 1];

for (Book book : books) {

if (book != null && book.ISBN.equals(isbn) && book.isAvailable) {

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

if (member.borrowedBooks[i] == null) {

member.borrowedBooks[i] = book;

book.isAvailable = false;

System.out.println(member.name + " borrowed: " + book.title);

return;

}

}

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

System.out.println(member.name + " cannot borrow more than 3 books.");

return;

}

}

System.out.println("Book not available.");

}

public static void returnBook(String isbn, int memberId) {

Member member = members[memberId - 1];

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

if (member.borrowedBooks[i] != null && member.borrowedBooks[i].ISBN.equals(isbn)) {

Book book = member.borrowedBooks[i];

member.borrowedBooks[i] = null;

book.isAvailable = true;

System.out.println(member.name + " returned: " + book.title);

return;

}

}

System.out.println(member.name + " did not borrow this book.");

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

}

public static void listAvailableBooks() {

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

for (Book book : books) {

if (book != null && book.isAvailable) {

System.out.println(book.title + " by " + book.author);

}

}

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

}

public static void listBorrowedBooks(int memberId) {

Member member = members[memberId - 1];

System.out.println(member.name + "'s Borrowed Books:");

for (Book book : member.borrowedBooks) {

if (book != null) {

System.out.println(book.title);

}

}

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

}

}

public class LibrarySystem {

public static void main(String[] args) {

Library.addBook(new Book("Java Programming", "John Doe", "123", true));

Library.addBook(new Book("Python Basics", "Jane Smith", "456", true));

Library.addBook(new Book("C++ Basics", "Mark Brown", "789", true));

Library.addMember(new Member("Puneeth", 1));

Library.addMember(new Member("Kumar", 2));

Library.listAvailableBooks();

Library.borrowBook("123", 1);

Library.borrowBook("789", 1);

Library.borrowBook("456", 2);

Library.returnBook("123", 1);

Library.listAvailableBooks();

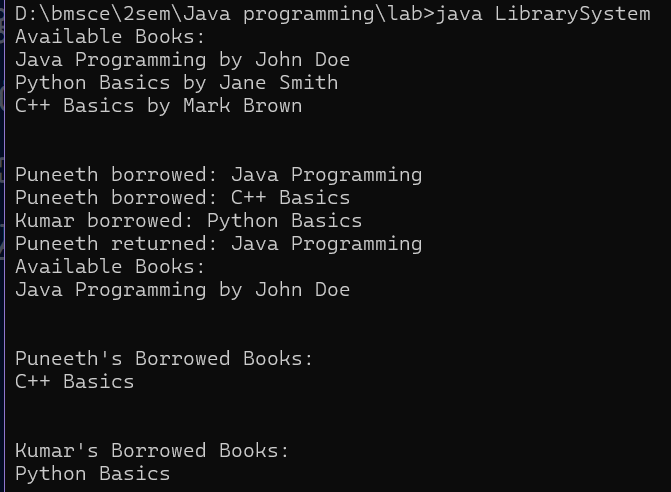
Library.listBorrowedBooks(1);

Library.listBorrowedBooks(2);

}

}

**OUTPUT:**



2. **Develop a program for an organization that manages different types of employees. There are**

**general employees, managers, and interns. Each type of employee has different ways of**

**calculating bonuses and benefits.**

**• Abstract Class: Employee, Fields: name, id, baseSalary**

**Constructor: Initializes all fields.**

**Abstract methods: double calculateBonus(), String getDetails()**

**• Subclass: Manager**

**Field: department, Bonus: 20% of base salary**

**Overrides getDetails()**

**• Subclass: Intern**

**Field: university, Bonus: Fixed: $500**

**Overrides getDetails()**

**• Subclass: Developer**

**Field: level (Junior, Mid, Senior), Bonus: Junior (10%), Mid (15%), Senior**

**(25%) of base salary**

**Overrides getDetails()**

**• Interface: Taxable**

**Method: double calculateTax()**

**• All employees are taxable: Tax is 10% of baseSalary + 5% of bonus**

**• Main class: Company**

**➢ Print details of each employee. Display total salary, bonuses and taxes.**

Programm:

interface Taxable {

double calculateTax();

}

abstract class Employee implements Taxable {

String name;

int id;

double baseSalary;

Employee(String name, int id, double baseSalary) {

this.name = name;

this.id = id;

this.baseSalary = baseSalary;

}

abstract double calculateBonus();

abstract String getDetails();

public double calculateTax() {

return 0.10 \* baseSalary + 0.05 \* calculateBonus();

}

}

class Manager extends Employee {

String department;

Manager(String name, int id, double baseSalary, String department) {

super(name, id, baseSalary);

this.department = department;

}

@Override

double calculateBonus() {

return 0.20 \* baseSalary;

}

@Override

String getDetails() {

return "Manager [Name: " + name + ", ID: " + id + ", Department: " + department + "]";

}

}

class Intern extends Employee {

String university;

Intern(String name, int id, double baseSalary, String university) {

super(name, id, baseSalary);

this.university = university;

}

@Override

double calculateBonus() {

return 500.0;

}

@Override

String getDetails() {

return "Intern [Name: " + name + ", ID: " + id + ", University: " + university + "]";

}

}

class Developer extends Employee {

String level;

Developer(String name, int id, double baseSalary, String level) {

super(name, id, baseSalary);

this.level = level;

}

@Override

double calculateBonus() {

switch (level.toLowerCase()) {

case "junior":

return 0.10 \* baseSalary;

case "mid":

return 0.15 \* baseSalary;

case "senior":

return 0.25 \* baseSalary;

default:

return 0.0;

}

}

@Override

String getDetails() {

return "Developer [Name: " + name + ", ID: " + id + ", Level: " + level + "]";

}

}

public class Company {

public static void main(String[] args) {

Employee e1 = new Manager("Ramesh", 101, 80000, "Sales");

Employee e2 = new Intern("Deepak", 102, 20000, "MIT");

Employee e3 = new Developer("Rocky", 103, 70000, "Senior");

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

for (Employee e : employees) {

System.out.println(e.getDetails());

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

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

System.out.println("Tax: $" + e.calculateTax());

System.out.println("Total Salary (Base + Bonus - Tax): Rs." + (e.baseSalary + e.calculateBonus() - e.calculateTax()));

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

}

}

}

OUTPUT:

