**Answer to the Question No. 1**

**Employee Class:**

public class Employee {

String name;

String ID;

double salary;

Employee(String name, String ID, double salary){

this.name = name;

this.ID = ID;

this.salary = salary;

}

void work(){

System.out.println("Employee "+name + " is working");

}

double calculateBonus(double performanceRating){

return salary\*0.15\*performanceRating;

}

}

**Manager Class:**

public class Manager extends Employee {

int teamSize;

String department;

Manager(String name, String ID, double salary, int teamSize, String department){

super(name,ID,salary);

this.teamSize = teamSize;

this.department = department;

}

@Override

void work(){

System.out.println("Manager "+ name+" is managing the "+department+" department, with a team of "+teamSize+" employees");

}

@Override

double calculateBonus(double performanceRating){

return salary\*0.2\*performanceRating;

}

void evaluateTeamPerformance(){

System.out.println("Manager "+ name+" is evaluating the performance of the team of the department: "+department);

}

void conductTeamMeeting(){

System.out.println("Manager "+ name+" is conducting a team meeting for the "+department+" department");

}

}

**Director Class:**

public class Director extends Manager {

int numberOfDepartments;

Director(String name, String ID, double salary, int teamSize, String department, int numberOfDepartments){

super(name,ID,salary,teamSize,department);

this.numberOfDepartments = numberOfDepartments;

}

@Override

void work(){

System.out.println("Director "+ name+" is overseeing "+numberOfDepartments+" departments and setting company strategies");

}

@Override

double calculateBonus(double performanceRating){

return salary\*0.25\*performanceRating;

}

void setCompanyStrategy(){

System.out.println("Director "+name+" is setting the company strategy");

}

void evaluateManagers(){

System.out.println("Director "+name+" is evaluating "+numberOfDepartments+" managers");

}

}

**Main Class:**

public class LabAssignment {

public static void main(String[] args) {

Employee e = new Employee("Omar", "2023", 10000);

Manager m = new Manager("Tasin", "2024", 20000,5,"A");

Director d = new Director("Kayes", "2025", 30000,10,"B", 5);

System.out.println("Employee details:");

e.work();

System.out.println("Bonus for employee "+e.name+": "+e.calculateBonus(0.5));

System.out.println();

System.out.println("Manager details:");

m.work();

System.out.println("Bonus for manager "+m.name+": "+m.calculateBonus(0.6));

m.evaluateTeamPerformance();

m.conductTeamMeeting();

System.out.println();

System.out.println("Director details:");

d.work();

System.out.println("Bonus for director "+d.name+": "+d.calculateBonus(0.7));

d.setCompanyStrategy();

d.evaluateManagers();

}

}

**Answer to the Question No. 2**

**AbstractUser Class:**

public abstract class AbstractUser {

abstract void login();

abstract void logout();

abstract void viewProfile();

}

**BookHandler Interface:**

public interface BookHandler {

void borrowBook(String title);

void returnBook(String title);

void addBook(String title);

void removeBook(String title);

void manageBook(String title);

}

**BookList Class:**

public class BookList {

public static List<String> books = new ArrayList<>();

public static void showBooks(){

System.out.print("Books currently available: ");

for(String book: books){

System.out.print(book + ", ");

}

System.out.println();

}

}

**LoggedInUsers Class:**

public class LoggedInUsers {

public static HashMap<String, String> users = new HashMap<>();

public static void addUser(String id, String password) {

users.put(id, password);

}

public static void removeUser(String id, String password) {

if (password.equals(users.get(id))) {

users.remove(id);

}

}

public static void printAllUsers() {

if (users.isEmpty()) {

System.out.println("No users are currently logged in.");

return;

}

System.out.println("Logged-in users:");

for (Map.Entry<String, String> entry : users.entrySet()) {

System.out.println("ID: " + entry.getKey() + ", Password: " + entry.getValue());

}

}

}

**Member Class:**

public class Member extends AbstractUser implements BookHandler {

String ID;

String password;

Member(String ID, String password){

this.ID = ID;

this.password = password;

}

@Override

void login(){

System.out.println("Member with ID - "+ID+" and password - "+password+" has logged in");

LoggedInUsers.addUser(ID, password);

}

@Override

void logout(){

System.out.println("Member with ID - "+ID+" and password - "+password+" has logged out");

LoggedInUsers.removeUser(ID, password);

}

@Override

void viewProfile(){

System.out.println("Member with ID - "+ID+" and password - "+password+" is viewing profile");

}

@Override

public void borrowBook(String title){

System.out.println("Member with ID - "+ID+" has borrowed the book titled: "+title);

BookList.books.remove(title);

}

@Override

public void returnBook(String title){

System.out.println("Member with ID - "+ID+" has returned the book titled: "+title);

BookList.books.add(title);

}

@Override

public void addBook(String title){

System.out.println("Sorry, members are not eligible for this action");

}

@Override

public void removeBook(String title){

System.out.println("Sorry, members are not eligible for this action");

}

@Override

public void manageBook(String title){

System.out.println("Sorry, members are not eligible for this action");

}

}

**Librarian Class:**

public class Librarian extends AbstractUser implements BookHandler {

String ID;

String password;

Librarian(String ID, String password){

this.ID = ID;

this.password = password;

}

@Override

void login(){

System.out.println("Librarian with ID - "+ID+" and password - "+password+" has logged in");

LoggedInUsers.addUser(ID, password);

}

@Override

void logout(){

System.out.println("Librarian with ID - "+ID+" and password - "+password+" has logged out");

LoggedInUsers.removeUser(ID, password);

}

@Override

void viewProfile(){

System.out.println("Librarian with ID - "+ID+" and password - "+password+" is viewing profile");

}

@Override

public void borrowBook(String title){

System.out.println("Sorry, librarians cannot perform this action");

}

@Override

public void returnBook(String title){

System.out.println("Sorry, librarians cannot perform this action");

}

@Override

public void addBook(String title){

System.out.println("Librarian with ID - "+ID+" has added the book titled: "+title);

BookList.books.add(title);

}

@Override

public void removeBook(String title){

System.out.println("Librarian with ID - "+ID+" has removed the book titled: "+title);

BookList.books.remove(title);

}

@Override

public void manageBook(String title){

System.out.println("Librarian with ID - "+ID+" has managed the book titled: "+title);

}

}

**Main Class:**

public class LabAssignmentQ2 {

public static void main(String[] args) {

Librarian Omar = new Librarian("000","123");

Omar.login();

Omar.viewProfile();

Omar.addBook("A");

Omar.addBook("B");

Omar.addBook("C");

Omar.removeBook("C");

Omar.borrowBook("A");

Omar.returnBook("A");

Omar.manageBook("A");

Omar.logout();

BookList.showBooks();

Member Tasin = new Member("001","456");

Tasin.login();

Tasin.viewProfile();

Tasin.borrowBook("A");

BookList.showBooks();

Tasin.returnBook("A");

Tasin.addBook("D");

Tasin.removeBook("A");

Tasin.manageBook("A");

Tasin.logout();

Member Kayes = new Member("002","789");

Member Aoyon = new Member("003","987");

Kayes.login();

Aoyon.login();

LoggedInUsers.printAllUsers();

Kayes.logout();

Librarian Tamim = new Librarian("004","654");

Librarian Nouman = new Librarian("005","321");

Librarian Rakin = new Librarian("006","000");

Tamim.login();

Nouman.login();

Rakin.login();

LoggedInUsers.printAllUsers();

}

}