**Experiment 4**

**Student Name:** Shreya Gupta **UID:** 22BCS15461

**Branch:** BE-CSE **Section/Group:**22BCS\_KPIT-901

**Semester:**6th **Date of Performance:**21|2|25 **Subject Name:** Project based learning with Java **Subject Code:**22CSH-359

# Aim:

Easy level : Write a Java program to implement an ArrayList that stores employee details (ID, Name, and Salary). Allow users to add, update, remove, and search employees.

Medium level: Create a program to collect and store all the cards to assist the users in finding all the cards in a given symbol using Collection interface.

Hard level: Develop a ticket booking system with synchronized threads to ensure no double booking of seats. Use thread priorities to simulate VIP bookings being processed first.

1. **Objective:** Following are the objectives:
   * **Understanding List Operations:** To learn how to manipulate lists and understand basic list operations.
   * **Application of Switch-Case:** Learn and apply the switch-case statement to determine specific outputs based on given conditions.
   * **Developing Problem-Solving Skills Objective:** To enhance problem-solving and logical thinking by implementing simple algorithms to manipulate data in a list.
   * **Dynamic Data Retrieval:** Simulate a real-world scenario of retrieving and displaying data dynamically based on input criteria.

# Implementation/Code: Easy level:

import java.util.ArrayList; import java.util.Scanner;

class Employee { int id;

String name; double salary;

public Employee(int id, String name, double salary) { this.id = id;

this.name = name; this.salary = salary;

}

@Override

public String toString() {

return "Employee ID: " + id + ", Name: " + name + ", Salary: " + salary;

}

}

public class Main {

private static ArrayList<Employee> employees = new ArrayList<>(); private static Scanner scanner = new Scanner(System.in);

public static void addEmployee() { System.out.print("Enter Employee ID: "); int id = scanner.nextInt(); scanner.nextLine(); // consume newline

System.out.print("Enter Employee Name: "); String name = scanner.nextLine(); System.out.print("Enter Employee Salary: "); double salary = scanner.nextDouble(); employees.add(new Employee(id, name, salary));

System.out.println("Employee added successfully!\n");

}

public static void updateEmployee() { System.out.print("Enter Employee ID to update: "); int id = scanner.nextInt();

for (Employee emp : employees) { if (emp.id == id) {

scanner.nextLine(); // consume newline System.out.print("Enter new Name: "); emp.name = scanner.nextLine(); System.out.print("Enter new Salary: "); emp.salary = scanner.nextDouble();

System.out.println("Employee updated successfully!\n"); return;

}

}

System.out.println("Employee not found!\n");

}

public static void removeEmployee() { System.out.print("Enter Employee ID to remove: "); int id = scanner.nextInt();

employees.removeIf(emp -> emp.id == id); System.out.println("Employee removed successfully!\n");

}

public static void searchEmployee() { System.out.print("Enter Employee ID to search: "); int id = scanner.nextInt();

for (Employee emp : employees) { if (emp.id == id) {

System.out.println(emp + "\n"); return;

}

}

System.out.println("Employee not found!\n");

}

public static void displayEmployees() { if (employees.isEmpty()) {

System.out.println("No employees found!\n"); return;

}

for (Employee emp : employees) { System.out.println(emp);

}

System.out.println();

}

public static void main(String[] args) { while (true) {

System.out.println("Employee Management System"); System.out.println("1. Add Employee"); System.out.println("2. Update Employee"); System.out.println("3. Remove Employee"); System.out.println("4. Search Employee"); System.out.println("5. Display Employees"); System.out.println("6. Exit");

System.out.print("Enter your choice: "); int choice = scanner.nextInt();

switch (choice) {

case 1: addEmployee(); break; case 2: updateEmployee(); break; case 3: removeEmployee(); break; case 4: searchEmployee(); break; case 5: displayEmployees(); break;

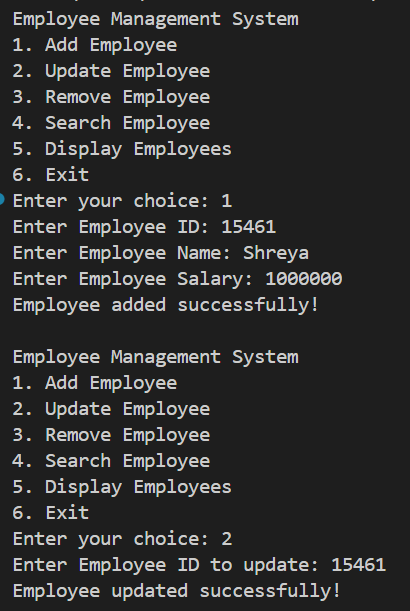
case 6: System.out.println("Exiting...\n"); return; default: System.out.println("Invalid choice!\n");

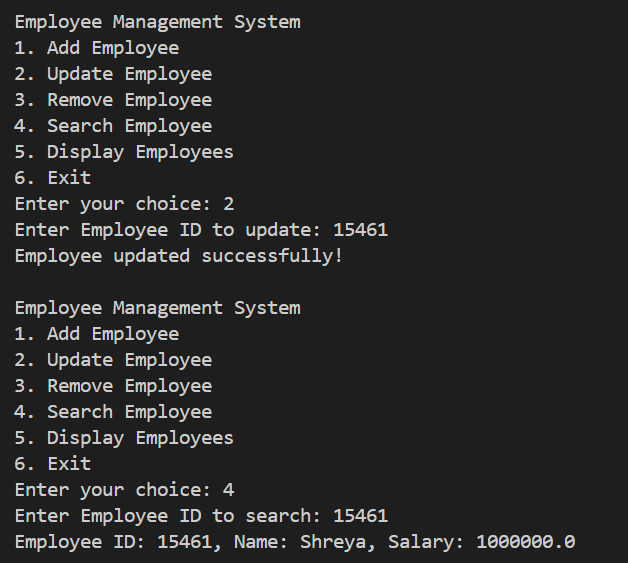
}

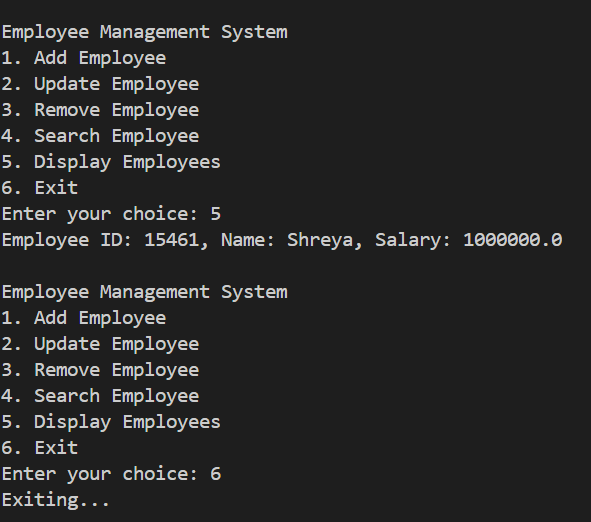
}

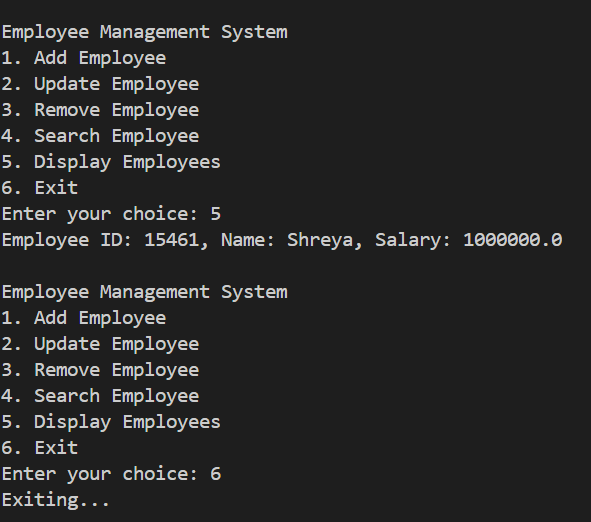
}

}









# Medium Level:

# import java.util.\*; class Card

# {

String symbol;

String value;

Card(String symbol, String value)

{

this.symbol = symbol;

this.value = value;

}

public String toString() {

return value + " of " + symbol;

}

}

public class CardCollection {

static Collection<Card> cards = new ArrayList<>(); static Scanner scanner = new Scanner(System.in);

static void addCard() {

System.out.print("Enter Card Symbol (Hearts, Diamonds, Clubs, Spades): "); String symbol = scanner.next();

System.out.print("Enter Card Value (e.g., Ace, 2, 3, King): "); String value = scanner.next();

cards.add(new Card(symbol, value)); System.out.println("Card added successfully!");

}

static void findCardsBySymbol() {

System.out.print("Enter symbol to search (Hearts, Diamonds, Clubs, Spades): "); String symbol = scanner.next();

boolean found = false; for (Card card : cards) {

if (card.symbol.equalsIgnoreCase(symbol)) { System.out.println(card);

found = true;

}

}

if (!found) {

System.out.println("No cards found for this symbol.");

}

}

static void displayAllCards() { if (cards.isEmpty()) {

System.out.println("No cards in the collection!");

} else {

for (Card card : cards) { System.out.println(card);

}

}

}

public static void main(String[] args) { while (true) {

System.out.println("\n1. Add Card\n2. Find Cards by Symbol\n3. Display All Cards\n4. Exit");

System.out.print("Choose an option: "); int choice = scanner.nextInt();

switch (choice) {

case 1 -> addCard();

case 2 -> findCardsBySymbol(); case 3 -> displayAllCards(); case 4 -> {

System.out.println("Exiting..."); return;

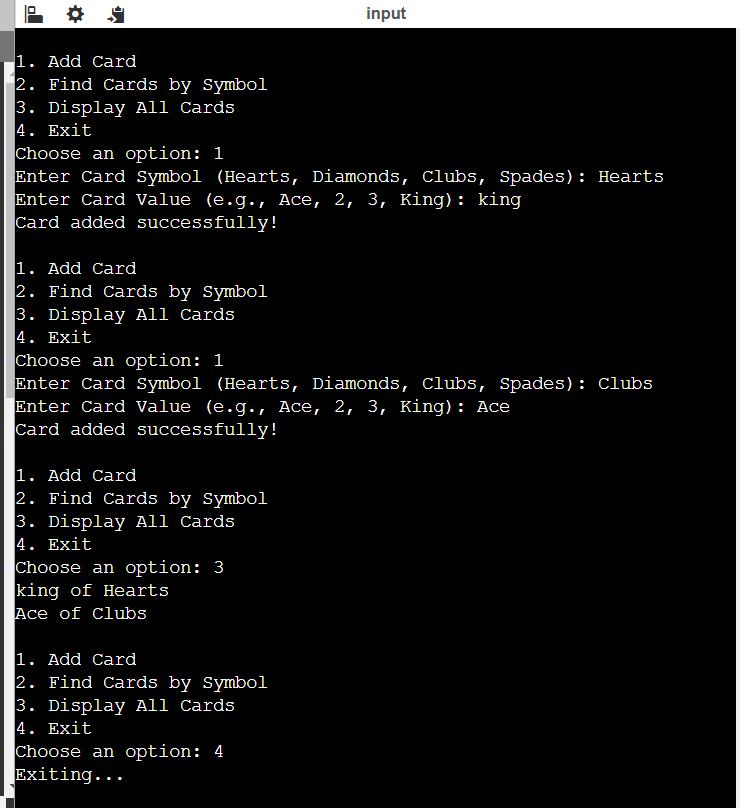
}

default -> System.out.println("Invalid choice!");

}

}

}

}

# Hard Level:

import java.util.Scanner;

import java.util.concurrent.locks.\*;

class TicketBookingSystem { private int availableSeats;

private final Lock lock = new ReentrantLock();

public TicketBookingSystem(int seats) { this.availableSeats = seats;

}

public void bookSeat(String customerType, int seats) { lock.lock();

try {

if (availableSeats >= seats) {

System.out.println(customerType + " booked " + seats + " seat(s) successfully.");

availableSeats -= seats;

} else {

System.out.println(customerType + " failed to book. Not enough seats available.");

}

} finally { lock.unlock();

}

}

}

class Customer extends Thread {

private TicketBookingSystem system; private int seats;

private String customerType;

public Customer(TicketBookingSystem system, int seats, String customerType, int priority) {

this.system = system; this.seats = seats;

this.customerType = customerType; setPriority(priority);

}

@Override

public void run() { system.bookSeat(customerType, seats);

}

}

public class Main {

public static void main(String[] args) { Scanner scanner = new Scanner(System.in);

System.out.print("Enter total number of available seats: "); int totalSeats = scanner.nextInt();

TicketBookingSystem system = new TicketBookingSystem(totalSeats);

System.out.print("Enter number of customers: "); int numCustomers = scanner.nextInt();

for (int i = 0; i < numCustomers; i++) { System.out.print("Enter customer type (VIP/Regular): "); String type = scanner.next();

System.out.print("Enter number of seats to book: "); int seats = scanner.nextInt();

int priority = type.equalsIgnoreCase("VIP") ? Thread.MAX\_PRIORITY : Thread.NORM\_PRIORITY;

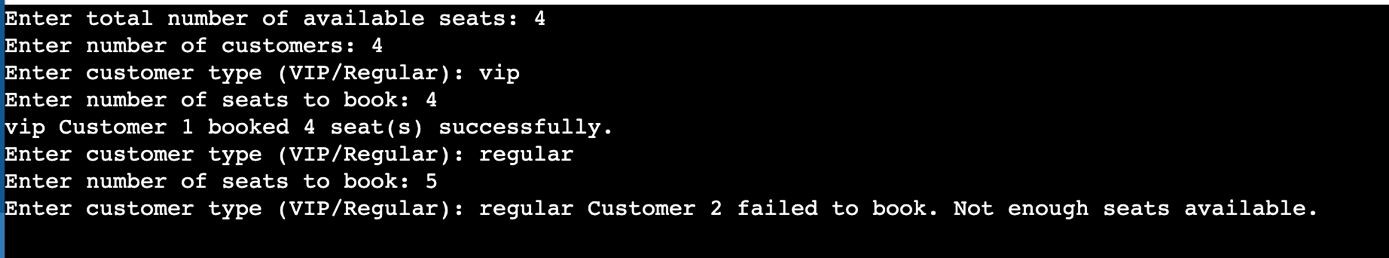
new Customer(system, seats, type + " Customer " + (i + 1), priority).start();

}

scanner.close();

}

}



# Learning Outcomes:

Following are the learning outcomes:

* + **Understanding Java Fundamentals:** I developed Improved understanding of basic Java concepts such as arrays, strings,, and conditionals.
  + **Working with Data Structures:** I Learnt how to store and manipulate data using data structures or related information.
  + **Switch Case Usage :** I Developed skills in using switch-case statements to determine designations and calculate allowances based on specific conditions.
  + **Command-Line Applications :** I Learnt how to build simple command-line applications that can take user inputs dynamically and produce desired outputs.
  + **Learnt string manipulation:** I developed deep understanding about how to manipulate string to perform various operations.