Experiment 4.1

Student Name: Vishal Bhatia
UID: 22BCS12608
Branch: BE-CSE
Section/Group: IoT_641(A)
Semester: 6th
Date of Performance: 14/2/25
Subject Name: Project based learning in Java
Subject Code: 22CSH-359

1. **Aim:** 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.
2. **Objective:** This program collects and stores N cards, grouping them by symbol in a map for easy retrieval. It displays distinct symbols in alphabetical order along with their associated cards, total count, and sum of numbers, ensuring efficient organization and user-friendly output.
3. **Implementation/Code:**

```
import java.util.*;
class Card {
    private String symbol;
    private int number;
    public Card(String symbol, int number) {
        this.symbol = symbol;
        this.number = number;
    }
    public String getSymbol() {
        return symbol;
    }
    public int getNumber() {
        return number;
    }
    public String toString() {
        return "Card{" +
                "symbol='" + symbol + '\'' +
                ", number=" + number +
                '}';
    }
}
public class CardGame {
    public static void main(String[] args) {
```

```
        Scanner scanner = new Scanner(System.in);
        Map<String, List<Card>> cardMap = new HashMap<>();
        System.out.print("Enter the number of cards (N): ");
        int N = scanner.nextInt();
        for (int i = 0; i < N; i++) {
            System.out.print("Enter symbol for card " + (i + 1) + ": ");
            String symbol = scanner.nextLine();
            System.out.print("Enter number for card " + (i + 1) + ": ");
            int number = scanner.nextInt();
             Card card = new Card(symbol, number);
            cardMap.putIfAbsent(symbol, new ArrayList<>());
            cardMap.get(symbol).add(card);
        }
        List<String> sortedSymbols = new ArrayList<>(cardMap.keySet());
        Collections.sort(sortedSymbols);
        for (String symbol : sortedSymbols) {
            List<Card> cards = cardMap.get(symbol);
            int count = cards.size();
            int sum = cards.stream().mapToInt(Card::getNumber).sum();
            System.out.println("Symbol: " + symbol);
            System.out.println("Number of cards: " + count);
            System.out.println("Cards: " + cards);
            System.out.println("Sum of numbers: " + sum);
            System.out.println();
        }
        scanner.close();
    }
```

4. Output

```
Enter the number of cards (N): 5
Enter symbol for card 1: A
Enter number for card 1: 1
Enter symbol for card 2: B
Enter number for card 2: 2
Enter symbol for card 3: C
Enter number for card 3: 3
Enter symbol for card 4: D
Enter number for card 4: 4
Enter symbol for card 5: E
Enter number for card 5: 5
```

```
Symbol: A
Number of cards: 1
Cards: [Card{symbol='A', number=1}]
Sum of numbers: 1

Symbol: B
Number of cards: 1
Cards: [Card{symbol='B', number=2}]
Sum of numbers: 2

Symbol: C
Number of cards: 1
Cards: [Card{symbol='C', number=3}]
Sum of numbers: 3

Symbol: D
Number of cards: 1
Cards: [Card{symbol='D', number=4}]
Sum of numbers: 4

Symbol: E
Number of cards: 1
Cards: [Card{symbol='E', number=5}]
Sum of numbers: 5
```

5. Learning Outcomes

- Understand how to use maps (dictionaries) for efficient data storage and retrieval.
- Learn to group and organize data based on a key attribute.
- Gain experience in handling user input and storing objects dynamically.
- Develop skills in sorting and displaying structured data in a meaningful.

Experiment 4.2

1. **Aim**: 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.
2. **Objective**: The objective is to implement array list.
3. **Implementation/Code**:

```
import java.util.ArrayList;
import java.util.Scanner;
class Employee {
    private int id;
    private String name;
    private double salary;
    public Employee(int id, String name, double salary) {
        this.id = id;
        this.name = name;
        this.salary = salary;
    }
    public int getId() {
        return id;
    }
    public String getName() {
        return name;
    }
    public double getSalary() {
        return salary;
    }
    public void setName(String name) {
        this.name = name;
    }
    public void setSalary(double salary) {
        this.salary = salary;
    }
    public String toString() {
        return "Employee [ID=" + id + ", Name=" + name + ", Salary=" + salary +
"]";
    }
}
```

```
public class EmployeeManager {
    private ArrayList<Employee> employees = new ArrayList<>();
    private Scanner scanner = new Scanner(System.in);
    public static void main(String[] args) {
        EmployeeManager manager = new EmployeeManager();
        manager.run();
    }
    public void run() {
        while (true) {
            System.out.println("\nEmployee 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 All Employees");
            System.out.println("6. Exit");
            System.out.print("Enter your choice: ");
            int choice = scanner.nextInt();
            scanner.nextLine();
            switch (choice) {
                case 1: addEmployee(); break;
                case 2: updateEmployee(); break;
                case 3: removeEmployee(); break;
                case 4: searchEmployee();break;
                case 5: displayAllEmployees(); break;
                case 6: System.out.println("Exiting..."); return;
                default: System.out.println("Invalid choice. Please try again.");
            }} }
    private 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();
```

```
        Employee employee = new Employee(id, name, salary);
        employees.add(employee);
        System.out.println("Employee added successfully.");
    }
    private void updateEmployee() {
        System.out.print("Enter Employee ID to update: ");
        int id = scanner.nextInt();
        scanner.nextLine(); // Consume newline
        for (Employee employee : employees) {
            if (employee.getId() == id) {
                System.out.print("Enter new name: ");
                String name = scanner.nextLine();
                System.out.print("Enter new salary: ");
                double salary = scanner.nextDouble();
                employee.setName(name);
                employee.setSalary(salary);
                System.out.println("Employee updated successfully.");
                return;
            }
        }
        System.out.println("Employee not found.");
    }
    private void removeEmployee() {
        System.out.print("Enter Employee ID to remove: ");
        int id = scanner.nextInt();
        for (Employee employee : employees) {
            if (employee.getId() == id) {
                employees.remove(employee);
                System.out.println("Employee removed successfully.");
                return;
            }
        }
        System.out.println("Employee not found.");
    }
    private void searchEmployee() {
        System.out.print("Enter Employee ID to search: ");
```

```
        int id = scanner.nextInt();
        for (Employee employee : employees) {
            if (employee.getId() == id) {
                System.out.println(employee);
                return;
            }
        }
        System.out.println("Employee not found.");
    }
    private void displayAllEmployees() {
        if (employees.isEmpty()) {
            System.out.println("No employees to display.");
        } else {
            for (Employee employee : employees) {
                System.out.println(employee);
            }
        }
    }
}
```

4. **Output**:

```
Employee Management System
1. Add Employee
2. Update Employee
3. Remove Employee
4. Search Employee
5. Display All Employees
6. Exit
Enter your choice: 1
Enter Employee ID: 12608
Enter Employee Name: Vishal Bhatia
Enter Employee Salary: 50000
Employee added successfully.

Employee Management System
1. Add Employee
2. Update Employee
3. Remove Employee
4. Search Employee
5. Display All Employees
6. Exit
Enter your choice: 5
Employee [ID=12608, Name=Vishal Bhatia, Salary=50000.0]
```

5. **Learning Outcomes:**
   - Understand how to use ArrayList for efficient data storage and retrieval.
   - Learn to group and organize data based on a key attribute.
   - Gain experience in handling user input and storing objects dynamically.
   - Develop skills in sorting and displaying structured data in a meaningful.

Experiment 4.3

1. **Aim**: 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.
2. **Objective**: The objective is to implement array list.
3. **Implementation/Code**:

```
import java.util.*;
class TicketBookingSystem {
   private int availableSeats;
   private final Lock lock = new ReentrantLock();
   public TicketBookingSystem(int seats) {
      this.availableSeats = seats;
   }
   public void bookSeat(String customerType) {
      lock.lock();
      try {
         if (availableSeats > 0) {
            System.out.println(customerType + " booked a seat. Seats left: " + (--
    availableSeats));
         } else {
            System.out.println(customerType + " tried to book a seat, but no seats
    are available.");
         }
      } finally {
         lock.unlock();
      }
   }
}
class BookingThread extends Thread {
   private final TicketBookingSystem bookingSystem;
```

```
    private final String customerType;

    public BookingThread(TicketBookingSystem bookingSystem, String
     customerType, int priority) {
        this.bookingSystem = bookingSystem;
        this.customerType = customerType;
        this.setPriority(priority);
    }
    public void run() {
        bookingSystem.bookSeat(customerType);
    }
}
public class TicketBookingApp {
    public static void main(String[] args) {
        int totalSeats = 10;
        TicketBookingSystem bookingSystem = new
     TicketBookingSystem(totalSeats);
        for (int i = 0; i < 5; i++) {
            new BookingThread(bookingSystem, "VIP Customer " + (i + 1),
     Thread.MAX_PRIORITY).start();
        }
        for (int i = 0; i < 10; i++) {
            new BookingThread(bookingSystem, "Regular Customer " + (i + 1),
     Thread.NORM_PRIORITY).start();
        }
    }
}
```

4. **Output**:

```
VIP Customer 1 booked a seat. Seats left: 9
VIP Customer 2 booked a seat. Seats left: 8
VIP Customer 3 booked a seat. Seats left: 7
VIP Customer 4 booked a seat. Seats left: 6
VIP Customer 5 booked a seat. Seats left: 5
Regular Customer 1 booked a seat. Seats left: 4
Regular Customer 2 booked a seat. Seats left: 3
Regular Customer 3 booked a seat. Seats left: 2
Regular Customer 5 booked a seat. Seats left: 1
Regular Customer 4 booked a seat. Seats left: 0
Regular Customer 6 tried to book a seat, but no seats are available.
Regular Customer 7 tried to book a seat, but no seats are available.
Regular Customer 8 tried to book a seat, but no seats are available.
Regular Customer 9 tried to book a seat, but no seats are available.
Regular Customer 10 tried to book a seat, but no seats are available.
```

5. Learning Outcomes:

- Understand how to threads.
- Learn to group and organize data based on a key attribute.
- Gain experience in handling user input and storing objects dynamically.
- Develop skills in sorting and displaying structured data in a meaningful.