Experiment 4

Student Name: Archit Kaushal
UID: 23BCS10313
Branch: CSE
Section/Group: KRG_2B
Semester: 5th
Date of Performance: 23/09/25
Subject Name: PBLJ
Subject Code: 23CSH-304

1. Aim:

To design and implement Java programs using data structures, collections, and multithreading for efficient data management and manipulation.
• To apply ArrayList, HashMap, and Thread synchronization in solving real-world problems.

- **Part A – Easy Level:**

• To create a Java program using ArrayList to store and manage employee details (ID, Name, Salary).
• To provide menu-driven options for adding, updating, removing, and searching employees.

- **Part B – Medium Level:**

• To create a Java program that stores playing cards in groups based on symbols using HashMap and ArrayList.
• To allow users to input a symbol and retrieve all the cards associated with it.

- **Part C – Hard Level:**

• To create a Java program that simulates a ticket booking system with multithreading.
• To implement synchronization and thread priorities to prevent double booking and prioritize VIP users.

2. Objective:

- ✓ To understand the use of Java Collections (ArrayList, HashMap) for efficient data management.
- ✓ To implement object-oriented programming concepts through custom classes like Employee and Card.
- ✓ To practice performing CRUD operations and grouping data using collection interfaces.

- ✓ To explore multithreading concepts including thread creation, priorities, and synchronization.

3. JAVA script and output:

EASY-LEVEL PROBLEM

```
package exp.pkg4;

import java.util.*;

class Employee {
   int id;
   String name;
   double salary;
   Employee(int id, String name, double salary) {
      this.id=id;
      this.name=name;
      this.salary=salary;
   }
   public String toString() {
      return "ID="+id+", Name="+name+", Salary="+salary;
   }
}

public class Exp4 {
   public static void main(String[] args) {
      Scanner sc=new Scanner(System.in);
      ArrayList<Employee> list=new ArrayList<>();
      while(true) {
         System.out.println("1.Add 2.Update 3.Remove 4.Search 5.Exit");
         int ch=sc.nextInt();
         if(ch==1) {
            System.out.print("Enter ID Name Salary: ");
            int id=sc.nextInt();
            String name=sc.next();
            double sal=sc.nextDouble();
            list.add(new Employee(id,name,sal));
         } else if(ch==2) {
            System.out.print("Enter ID to Update: ");
```

```
            int id=sc.nextInt();
            for(Employee e:list) {
               if(e.id==id) {
                  System.out.print("Enter New Name and Salary: ");
                  e.name=sc.next();
                  e.salary=sc.nextDouble();
                  break;
               }
            }
         } else if(ch==3) {
            System.out.print("Enter ID to Remove: ");
            int id=sc.nextInt();
            list.removeIf(e->e.id==id);
         } else if(ch==4) {
            System.out.print("Enter ID to Search: ");
            int id=sc.nextInt();
            for(Employee e:list) {
               if(e.id==id) {
                  System.out.println("Employee Found: "+e);
               }
            }
         } else break;
      }
   }
}
```

OUTPUT:

```
run:
1.Add 2.Update 3.Remove 4.Search 5.Exit
1
Enter ID Name Salary: 23 Akshara 25000
1.Add 2.Update 3.Remove 4.Search 5.Exit
1
Enter ID Name Salary: 25 Ishika 23000
1.Add 2.Update 3.Remove 4.Search 5.Exit
4
Enter ID to Search: 23
Employee Found: ID=23, Name=Akshara, Salary=25000.0
1.Add 2.Update 3.Remove 4.Search 5.Exit
5
BUILD SUCCESSFUL (total time: 57 seconds)
```

Figure 1: Easy Level

MEDIUM LEVEL PROBLEM:

```
package exp.pkg4;


import java.util.*;


class Card {
    String symbol;
    int number;
    Card(String symbol,int number) {
        this.symbol=symbol;
        this.number=number;
    }
    public String toString() {
        return symbol+" - "+number;
    }
}


public class Exp4 {
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        HashMap<String,ArrayList<Card>> map=new HashMap<>();
        map.put("Spade",new ArrayList<>(Arrays.asList(new Card("Spade",1),new
Card("Spade",3),new Card("Spade",10))));
        map.put("Heart",new ArrayList<>(Arrays.asList(new Card("Heart",2),new
Card("Heart",5))));
        map.put("Diamond",new ArrayList<>(Arrays.asList(new Card("Diamond",7))));
```

```
        System.out.print("Enter symbol: ");
        String s=sc.next();
        if(map.containsKey(s)) {
            System.out.println("Cards with symbol '"+s+"':");
            for(Card c:map.get(s)) {
                System.out.println(c);
            }
        } else System.out.println("No cards found");
    }
}
```

OUTPUT:

```
run:
Enter symbol: Spade
Cards with symbol 'Spade':
Spade - 1
Spade - 3
Spade - 10
BUILD SUCCESSFUL (total time: 6 seconds)
```

Figure 2: Medium Level

HARD LEVEL PROBLEM

```
package exp.pkg4;

class TicketBooking {
    private boolean booked=false;
    public synchronized void bookTicket(String user) {
        if(!booked) {
            System.out.println(user+" booked Seat 1");
```

```
            booked=true;
        } else {
            System.out.println(user+" could not book. Seat already booked.");
        }
    }
}

class UserThread extends Thread {
    TicketBooking tb;
    String user;
    UserThread(TicketBooking tb,String user) {
        this.tb=tb;
        this.user=user;
    }
    public void run() {
        tb.bookTicket(user);
    }
}

public class Exp4 {
    public static void main(String[] args) {
        TicketBooking tb=new TicketBooking();
        UserThread t1=new UserThread(tb,"Normal User");
        UserThread t2=new UserThread(tb,"VIP User");
        t2.setPriority(Thread.MAX_PRIORITY);
        t1.setPriority(Thread.MIN_PRIORITY);
```

```
        t2.start();
        t1.start();
    }
}
```

OUTPUT:

```
VIP User booked Seat 1
Normal User could not book. Seat already booked.
BUILD SUCCESSFUL (total time: 0 seconds)
```