PRACTICAL 4 – MAP INTERFACE (ADVANCED JAVA)

AIM

To study and implement Map Interface in Java and perform the following operations: a) Add items in the map
b) Remove items from the map
c) Search a specific key from the map
d) Get value of the specified key
e) Insert map elements of one map into another map
f) Print all keys and values of the map

SOFTWARE REQUIREMENTS

• Operating System: Windows / Linux
• JDK: Version 7 or 8
• IDE: NetBeans IDE 7.x

THEORY

The Map interface is a part of the Java Collection Framework but does not inherit from the Collection interface. A Map stores data in the form of **key–value pairs**, where each key is unique and maps to exactly one value. Common implementations of Map include HashMap, LinkedHashMap, and TreeMap. In this practical, HashMap is used because it provides fast access and does not maintain insertion order.

SOURCE CODE

```
package mapdemo;

import java.util.*;

public class MapOperationsDemo {
    public static void main(String[] args) {

        // a) Add items in the map
        Map<Integer, String> map1 = new HashMap<Integer, String>();
        map1.put(1, "Java");
        map1.put(2, "Advanced Java");
        map1.put(3, "Spring");

        System.out.println("Map after adding elements:");
        System.out.println(map1);

        // b) Remove item from the map
        map1.remove(2);
        System.out.println("\nMap after removing key 2:");
        System.out.println(map1);

        // c) Search specific key
        int searchKey = 3;
        if (map1.containsKey(searchKey)) {
            System.out.println("\nKey " + searchKey + " found in map");
        } else {
            System.out.println("\nKey " + searchKey + " not found");
        }

        // d) Get value of specified key
        System.out.println("Value for key " + searchKey + ": " + map1.get(searchKey));

        // e) Insert map elements into another map
        Map<Integer, String> map2 = new HashMap<Integer, String>();
        map2.put(4, "Hibernate");
        map2.put(5, "Microservices");

        map2.putAll(map1);

        System.out.println("\nSecond Map after merging:");
        System.out.println(map2);

        // f) Print all keys and values
        System.out.println("\nKeys and Values:");
        for (Map.Entry<Integer, String> entry : map2.entrySet()) {
            System.out.println("Key: " + entry.getKey() + " Value: " + entry.getValue());
        }
    }
}
```

OUTPUT

```
Map after adding elements:
{1=Java, 2=Advanced Java, 3=Spring}

Map after removing key 2:
{1=Java, 3=Spring}

Key 3 found in map
Value for key 3: Spring

Second Map after merging:
{1=Java, 3=Spring, 4=Hibernate, 5=Microservices}

Keys and Values:
Key: 1 Value: Java
Key: 3 Value: Spring
Key: 4 Value: Hibernate
Key: 5 Value: Microservices
```

NETBEANS 7 – DETAILED EXECUTION STEPS

1. Open NetBeans IDE 7
2. Click File → New Project
3. Select Java → Java Application → Next
4. Enter Project Name: MapDemo
5. Uncheck 'Create Main Class'
6. Click Finish
7. Right-click Source Packages → New → Java Package
8. Enter Package Name: mapdemo
9. Right-click package → New → Java Class
10. Enter Class Name: MapOperationsDemo
11. Paste the source code
12. Press Shift + F6 to run the program

RESULT

Thus, the Map Interface was successfully implemented and all required operations were performed using HashMap.