COLLECTIONS IN JAVA(Map interface)

A framework is a set of classes and interfaces which provide a ready-made architecture. In order to implement a new feature or a class, there is no need to define a framework itself. However, an optimal object-oriented design often includes a framework with a collection of classes such that all the classes perform the same kind of task.

In Java, a Collection is a framework that provides an architecture to store and manipulate a group of objects. The Java Collection Framework (JCF) is part of java.util package and includes interfaces and classes for managing groups of objects.

Map Interface

In Java, Map Interface is present in the java.util package represents a mapping between a key and a value. Java Map interface is not a subtype of the Collection interface. Therefore it behaves a bit differently from the rest of the collection types.

- **No Duplicates in Keys**: Ensures that keys are unique. However, values can be duplicated.
- **Null Handling:** Allows one null key in implementations like HashMap and LinkedHashMap and allows multiple null values in most implementations.
- **Thread-Safe Alternatives:** Use ConcurrentHashMap for thread-safe operations also, wrap an existing Map using Collections.synchronizedMap() for synchronized access.

The Map data structure in Java is implemented by two interfaces, the Map Interface and the SortedMap Interface. Three primary classes implement these interfaces HashMap, TreeMap, and LinkedHashMap.

Maps are perfect to use for key-value association mapping such as dictionaries. The maps are used to perform lookups by keys or when someone wants to retrieve and update elements by keys. Some common scenarios are as follows:

- A map of error codes and their descriptions.
- A map of zip codes and cities.
- A map of managers and employees. Each manager (key) is associated with a list of employees (value) he manages.
- A map of classes and students. Each class (key) is associated with a list of students (value).

Methods in Java Map Interface

Method	Action Performed
clear()	This method is used in Java Map Interface to clear and remove all of the elements or mappings from a specified Map collection.
containsKey(Object)	This method is used in Map Interface in Java to check whether a particular key is being mapped into the Map or not. It takes the key element as a parameter and returns True if that element is mapped in the map.
containsValue(Object)	This method is used in Map Interface to check whether a particular value is being mapped by a single or more than one key in the Map. It takes the value as a parameter and returns True if that value is mapped by any of the keys in the map.
entrySet()	This method is used in Map Interface in Java to create a set out of the same elements contained in the map. It basically returns a set view of the map or we can create a new set and store the map elements into them.
equals(Object)	This method is used in Java Map Interface to check for equality between two maps. It verifies whether the elements of one map passed as a parameter is equal to the elements of this map or not.
get(Object)	This method is used to retrieve or fetch the value mapped by a particular key mentioned in the parameter. It returns NULL when the map contains no such mapping for the key.
hashCode()	This method is used in Map Interface to generate a hashCode for the given map containing keys and values.

isEmpty()	This method is used to check if a map is having any entry for key and value pairs. If no mapping exists, then this returns true.
keySet()	This method is used in Map Interface to return a Set view of the keys contained in this map. The set is backed by the map, so changes to the map are reflected in the set, and vice-versa.
put(Object, Object)	This method is used in Java Map Interface to associate the specified value with the specified key in this map.
putAll(Map)	This method is used in Map Interface in Java to copy all of the mappings from the specified map to this map.
remove(Object)	This method is used in Map Interface to remove the mapping for a key from this map if it is present in the map.
size()	This method is used to return the number of key/value pairs available in the map.
values()	This method is used in Java Map Interface to create a collection out of the values of the map. It basically returns a Collection view of the values in the HashMap.

1. HashMap

HashMap is a part of Java's collection since Java 1.2. It provides the basic implementation of the Map interface of Java. It stores the data in (Key, Value) pairs. To access a value one must know its key. This class uses a technique called Hashing. Hashing is a technique of converting a large String to a small String that represents the same String. A shorter value helps in indexing and faster searches.

```
// Java Program to illustrate the Hashmap Class

// Importing required classes

import java.util.*;

// Main class
```

```
public class GFG {

    // Main driver method

    public static void main(String[] args)

    {

        // Creating an empty HashMap

        Map<String, Integer> map = new HashMap<>();

        // Inserting entries in the Map

        // using put() method

        map.put("vishal", 10);

        map.put("sachin", 30);

        map.put("vaibhav", 20);

        // Iterating over Map

        for (Map.Entry<String, Integer> e : map.entrySet())

            // Printing key-value pairs

            System.out.println(e.getKey() + " "

                               + e.getValue());

    }

}
```

2. LinkedHashMap

LinkedHashMap is just like HashMap with the additional feature of maintaining an order of elements inserted into it. HashMap provided the advantage of quick insertion, search, and deletion but it never maintained the track and order of insertion which the LinkedHashMap provides where the elements can be accessed in their insertion order

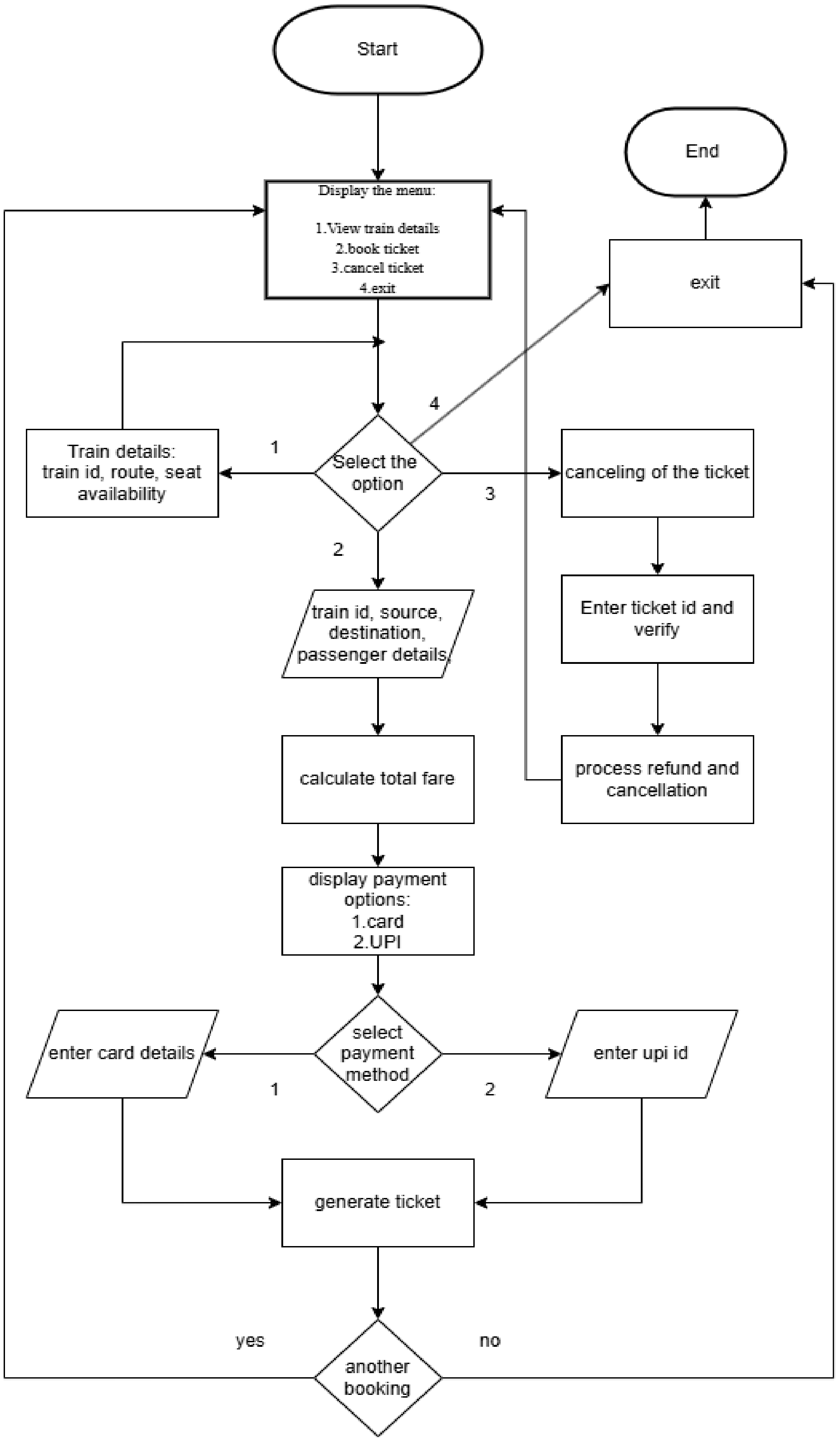
```
import java.util.*;

public class GFG {

    public static void main(String[] args)

    {

        // Creating an empty LinkedHashMap

        Map<String, Integer> map = new LinkedHashMap<>();

        // Inserting pair entries in above Map

        // using put() method

        map.put("vishal", 10);

        map.put("sachin", 30);

        map.put("vaibhav", 20);

        // Iterating over Map

        for (Map.Entry<String, Integer> e : map.entrySet())


            // Printing key-value pairs

            System.out.println(e.getKey() + " "
                        + e.getValue());
    }
}
```

3. TreeMap

The TreeMap in Java is used to implement the Map interface and NavigableMap along with the Abstract Class. The map is sorted according to the natural ordering of its keys, or by a Comparator provided at map creation time, depending on which constructor is used. This proves to be an efficient way of sorting and storing the key-value pairs. The storing order maintained by the treemap must be consistent with equals just like any other sorted map, irrespective of the explicit comparators

```
// Java Program to Illustrate TreeMap Class

// Importing required classes

import java.util.*;

// Main class

public class GFG {

    // Main driver method

    public static void main(String[] args)

    {

        // Creating an empty TreeMap

        Map<String, Integer> map = new TreeMap<>();

        // Inserting custom elements in the Map

        // using put() method

        map.put("vishal", 10);
        map.put("sachin", 30);

        map.put("vaibhav", 20);

        // Iterating over Map using for each loop

        for (Map.Entry<String, Integer> e : map.entrySet())

            // Printing key-value pairs

            System.out.println(e.getKey() + " "

                               + e.getValue());

    }

}
```

Flow chart of Train ticket purchase console app


Start
End
Display the menu:
1.View train details
2.book ticket
3.cancel ticket
4.exit
exit
4
1
Train details: train id, route, seat availability
Select the option
canceling of the ticket
3
2
train id, source, destination, passenger details
Enter ticket id and verify
calculate total fare
process refund and cancellation
display payment options: 1.card 2.UPI
enter card details
select payment method
enter upi id
1
2
generate ticket
yes
no
another booking