Introduction
The Collections Framework in Java is a unified architecture for representing and manipulating groups of objects. It allows common operations such as searching, sorting, inserting, deleting, etc. to be performed efficiently and with different data structures such as lists, sets, and maps.

Interfaces of the Collections API

1. List: An ordered collection that allows duplicate elements. Common implementations include:
    a. ArrayList
    b. LinkedList

2. Set: A collection that does not allow duplicate elements. Common implementations include:
    a. HashSet
    b. TreeSet

3. Map: A structure that associates keys with values. It does not allow duplicate keys, but allows duplicate values. Common implementations include:
    a. HashMap
    b. TreeMap

List in Java
List is an ordered collection that allows duplicates. Elements can be accessed by their index.
Example:

```
import java.util.ArrayList;
import java.util.List;

public class ListExample {
    public static void main(String[] args) {
        List<String> fruits = new ArrayList<>();
        fruits.add("Apple");
        fruits.add("Banana");
        fruits.add("Orange");
        fruits.add("Banana"); // Permite duplicados

        System.out.println("Fruits List: " + fruits);

        // Accediendo por índice
        System.out.println("First Fruit: " + fruits.get(0));
    }
}
```

Expected Output:
Fruits List: [Apple, Banana, Orange, Banana]
First Fruit: Apple

Set in Java
Set is a collection that does not allow duplicates and does not guarantee any particular order (in the case of HashSet). Example:

```
import java.util.HashSet;
import java.util.Set;

public class SetExample {
    public static void main(String[] args) {
        Set<String> fruits = new HashSet<>();
        fruits.add("Apple");
        fruits.add("Banana");
        fruits.add("Orange");
        fruits.add("Banana"); // Ignorado, ya que Banana ya existe

        System.out.println("Fruits Set: " + fruits);
    }
}
```

Expected Output:
Fruits Set: [Banana, Orange, Apple]

Map in Java
Map is a collection of key-value pairs, where each key is unique. It is useful for quickly looking up values by their associated key. Example:

```
import java.util.HashMap;
import java.util.Map;

public class MapExample {
    public static void main(String[] args) {
        Map<String, Integer> fruitPrices = new HashMap<>();
        fruitPrices.put("Apple", 50);
        fruitPrices.put("Banana", 20);
        fruitPrices.put("Orange", 30);
        fruitPrices.put("Banana", 25); // Sobrescribe el valor anterior

        System.out.println("Fruit Prices: " + fruitPrices);

        // Obteniendo el precio de un ítem específico
        System.out.println("Price of Apple: " + fruitPrices.get("Apple"));
    }
}
```

Expected Output:
Fruit Prices: {Apple=50, Banana=25, Orange=30}
Price of Apple: 50

Differences between List, Set and Map:

1. List:
    a. Permite duplicados.
    b. Los elementos están ordenados y accesibles por índice.

2. Set:
    a. No permite duplicados.
    b. No garantiza orden.

3. Map:
    a. Almacena pares clave-valor.
    b. No permite claves duplicadas, pero permite valores duplicados.