CODING SHUTTLE

Week 7 LIVE

Collection Framework

In This Lecture

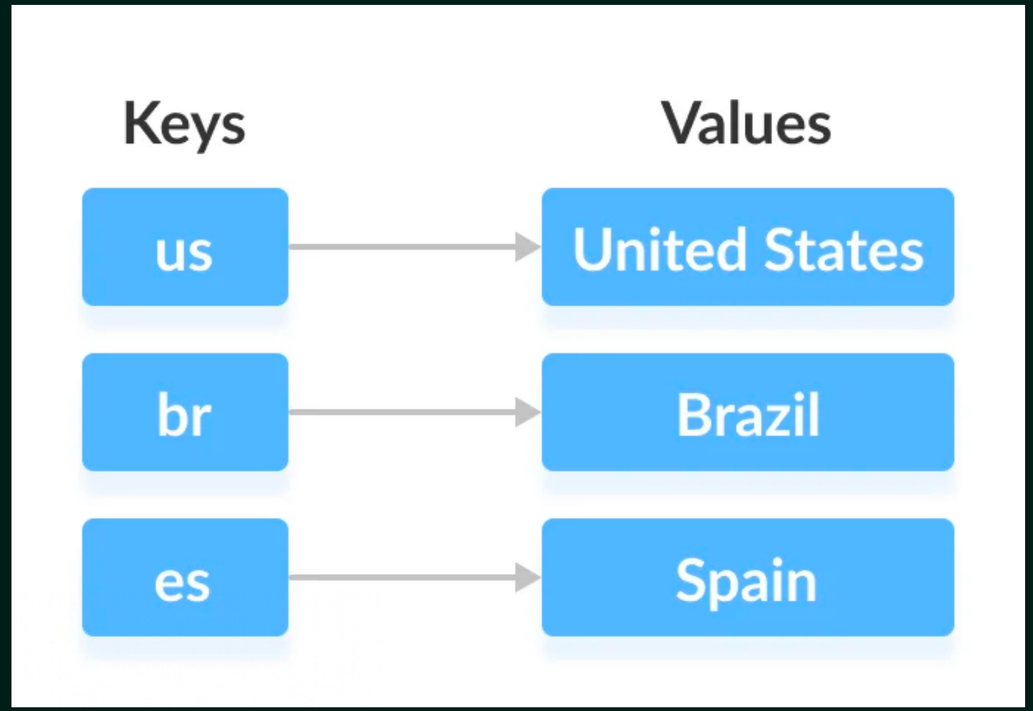
1. The Map Interface
2. Comparable and Comparators

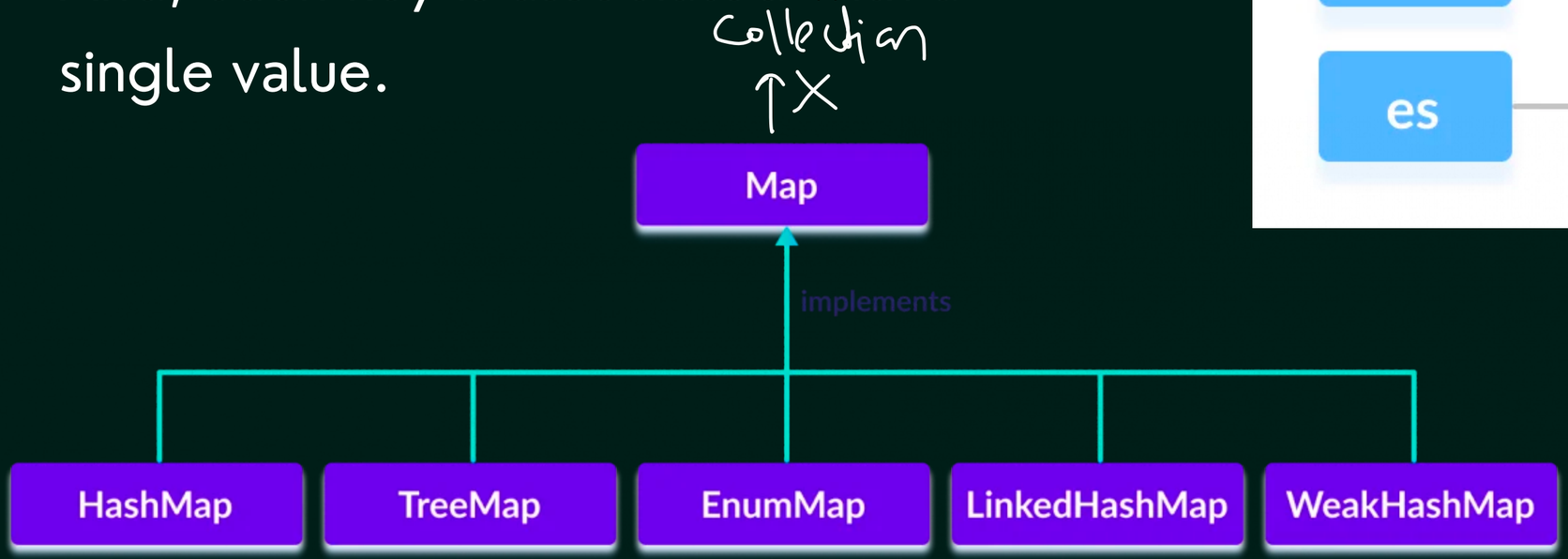
The Map Interface

In Java, elements of Map are stored in key/value pairs. Keys are unique values associated with individual Values.

A map cannot contain duplicate keys.

And, each key is associated with a single value.





Methods of Map

- put(K, V) - Inserts the association of a key K and a value V into the map. If the key is already present, the new value replaces the old value.
- putAll() - Inserts all the entries from the specified map to this map.
- putIfAbsent(K, V) - Inserts the association if the key K is not already associated with the value V.
- get(K) - Returns the value associated with the specified key K. If the key is not found, it returns null.
- getOrDefault(K, defaultValue) - Returns the value associated with the specified key K. If the key is not found, it returns the defaultValue.
- containsKey(K) - Checks if the specified key K is present in the map or not.

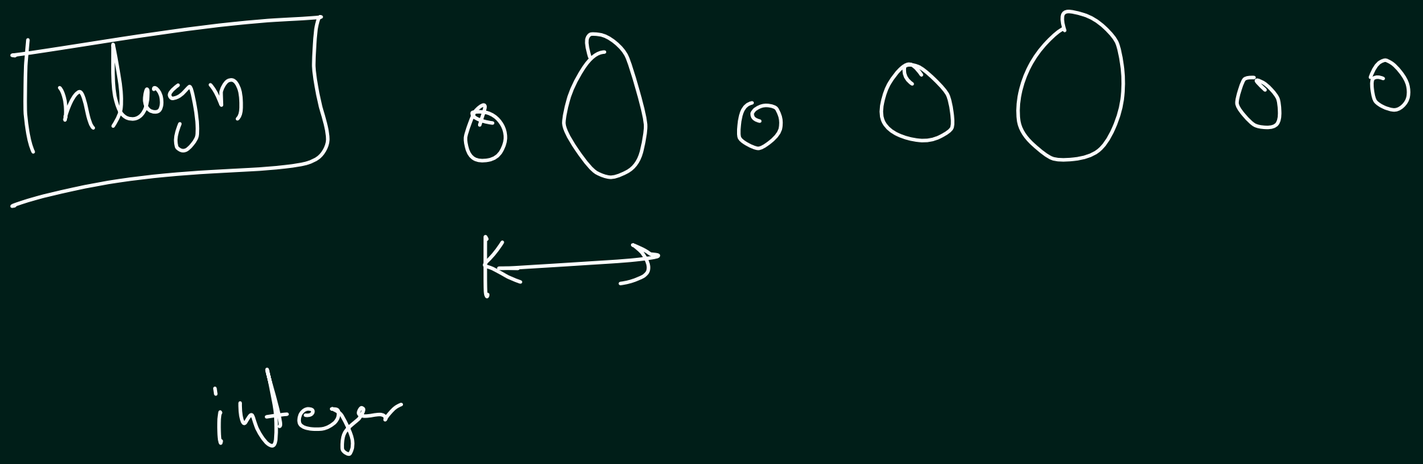
Methods of Map

- containsKey(K) - Checks if the specified key K is present in the map or not.
- containsValue(V) - Checks if the specified value V is present in the map or not.
- replace(K, V) - Replace the value of the key K with the new specified value V.
- replace(K, oldValue, newValue) - Replaces the value of the key K with the new value newValue only if the key K is associated with the value oldValue.
- remove(K) - Removes the entry from the map represented by the key K.
- remove(K, V) - Removes the entry from the map that has key K associated with value V.
- keySet() - Returns a set of all the keys present in a map.
- values() - Returns a set of all the values present in a map.
- entrySet() - Returns a set of all the key/value mapping present in a map.

Comparable and Comparator

To sort custom objects like Student or Employee, we need to provide explicit sorting logic.

To achieve this, Java provides the Comparable and Comparator interfaces. Comparable and Comparator in Java allow us to define custom sorting behavior for objects, including sorting based on multiple data members.



The Comparable Interface

This interface is found in java.lang package and contains only one method named compareTo(Object). It provides a single sorting sequence only, i.e., you can sort the elements on the basis of single data member only. For example, it may be rollno, name, age or anything else.

public int compareTo(Object obj): It is used to compare the current object with the specified object. It returns

- positive integer, if the current object is greater than the specified object.
- negative integer, if the current object is less than the specified object.
- zero, if the current object is equal to the specified object.

The Comparator Interface

The Comparator interface defines a compare(arg1, arg2) method with two arguments that represent compared objects, and works similarly to the Comparable.compareTo() method.

```
public class PlayerAgeComparator implements Comparator<Player> {

    @Override
    public int compare(Player firstPlayer, Player secondPlayer) {
       return Integer.compare(firstPlayer.getAge(), secondPlayer.getAge());
    }

}
```

The Java 8 Comparator Interface

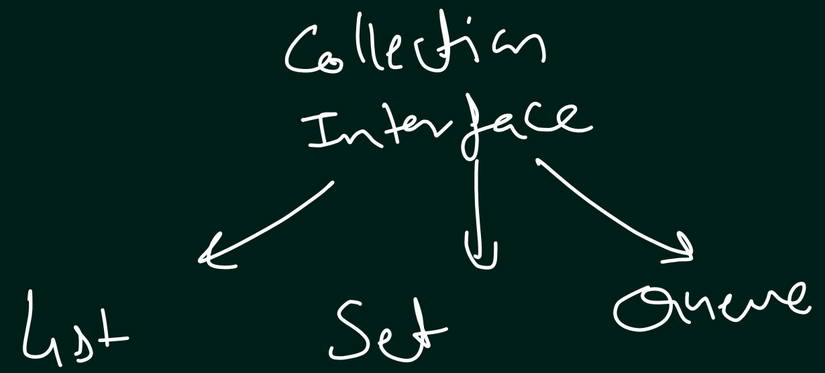
Java 8 provides new ways of defining Comparators by using lambda expressions, and the comparing() static factory method.

```
Comparator byRanking =
  (Player player1, Player player2) -> Integer.compare(player1.getRanking(),
player2.getRanking());
```

```
Comparator<Player> byRanking = Comparator
  .comparing(Player::getRanking);
Comparator<Player> byAge = Comparator
  .comparing(Player::getAge);
```

The Java 8 Comparator Interface

Collection framework



Iterator

Map