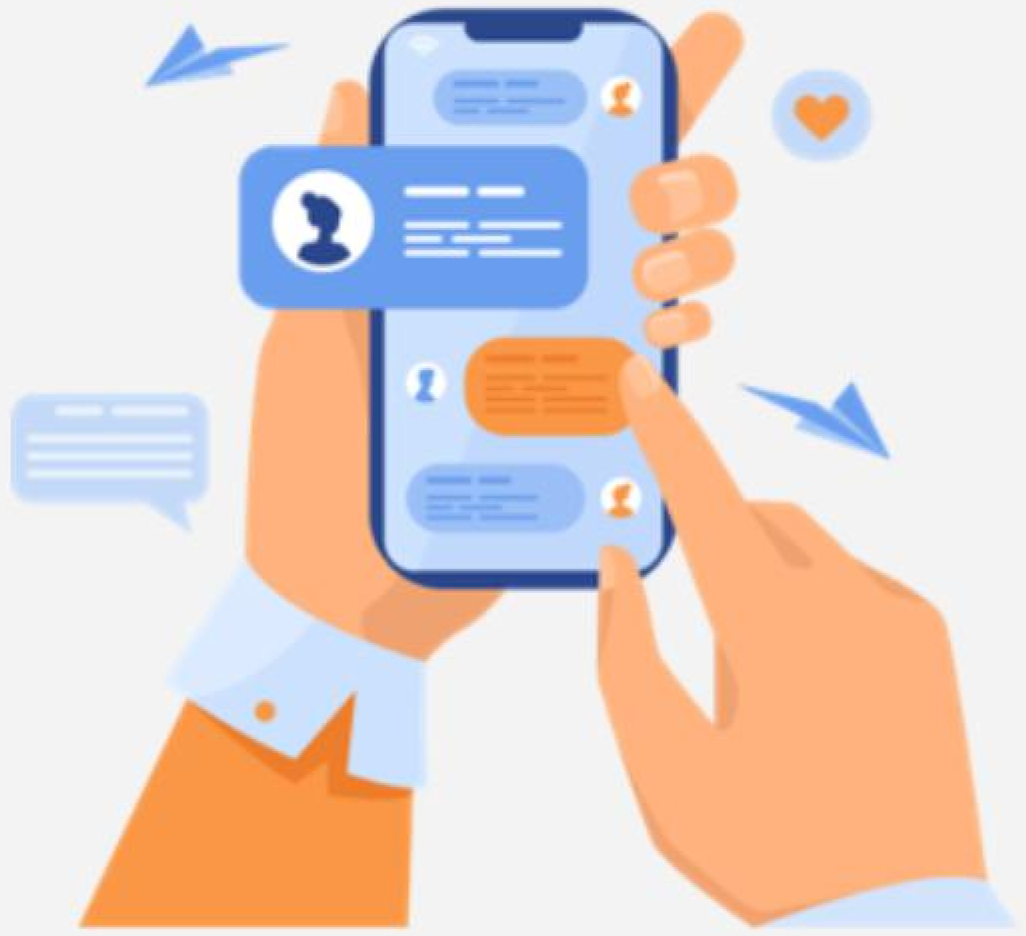
Slide Note



A Chat Application

- How can Andrew create a collection of contacts and ensure the uniqueness of the contacts in the chat group?
- Which collection interface should be used?

Menu

Slide Note

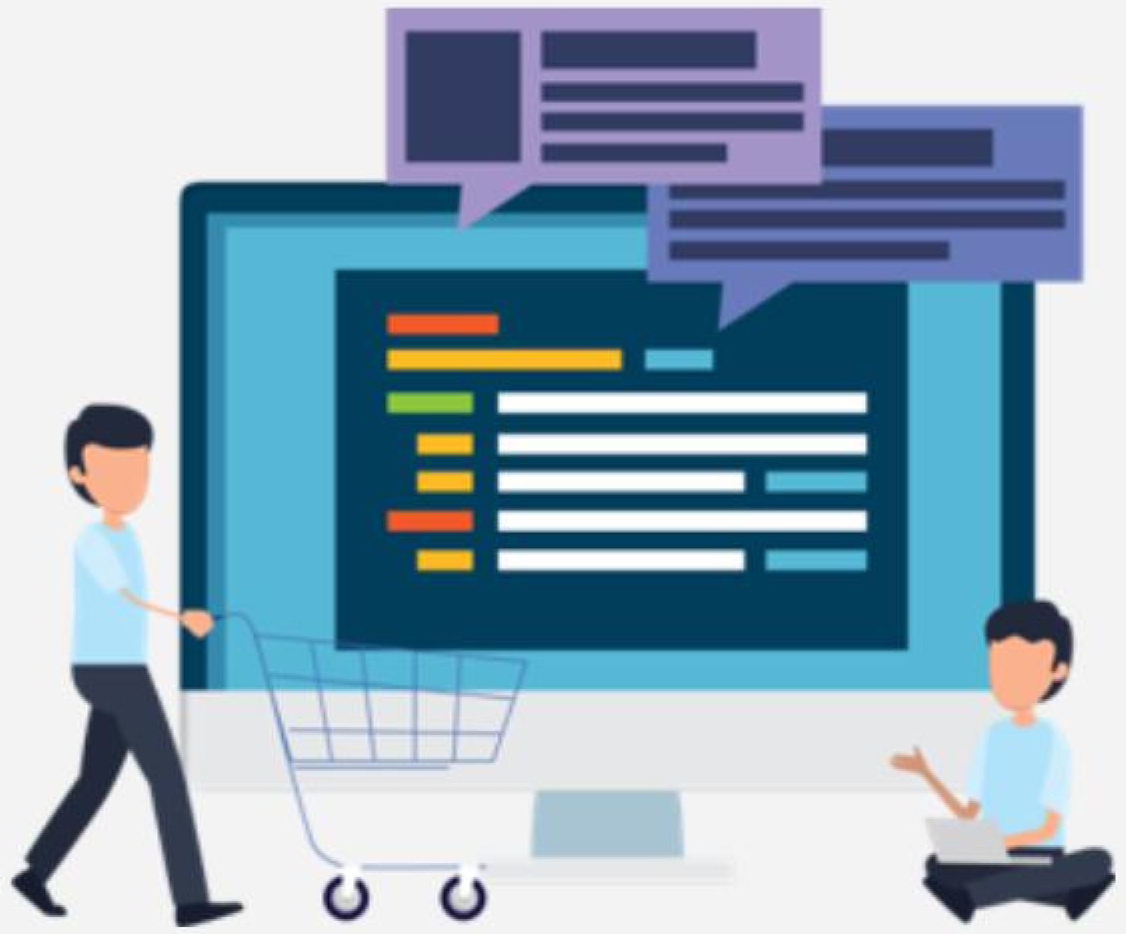
A Chat Application (cont'd)

- Now, Andrew wants to have some details about the chat users; for example, their address, date of birth, etc.
- He wants to retrieve all the details based on a single key, the contact number.
- What type of collection should Andrew use to fulfill this requirement?

Menu

00:02 00:00:30 02/ 30

Slide Note



Customer Details

- Andrew wants to link the chat application to an e-commerce portal where customers can buy various products.
- He wants to analyze the customers' purchasing history. To do this, he needs to use the contact number of the customers to retrieve their purchase history.
- What type of collection should Andrew use to fulfill this requirement?

Menu

Slide Note

Manipulate Objects Using Unordered Collections and Construct Objects as a Key Value Pair

Menu

00:04:00:00:30 04/ 30

Slide Note

Learning Objectives

- Describing the Set Interface
- Implementing HashSet and TreeSet
- Knowing the Difference Between List and Set
- Understanding the Map Interface
- Implementing HashMap and TreeMap
- Iterating Through Map

Menu

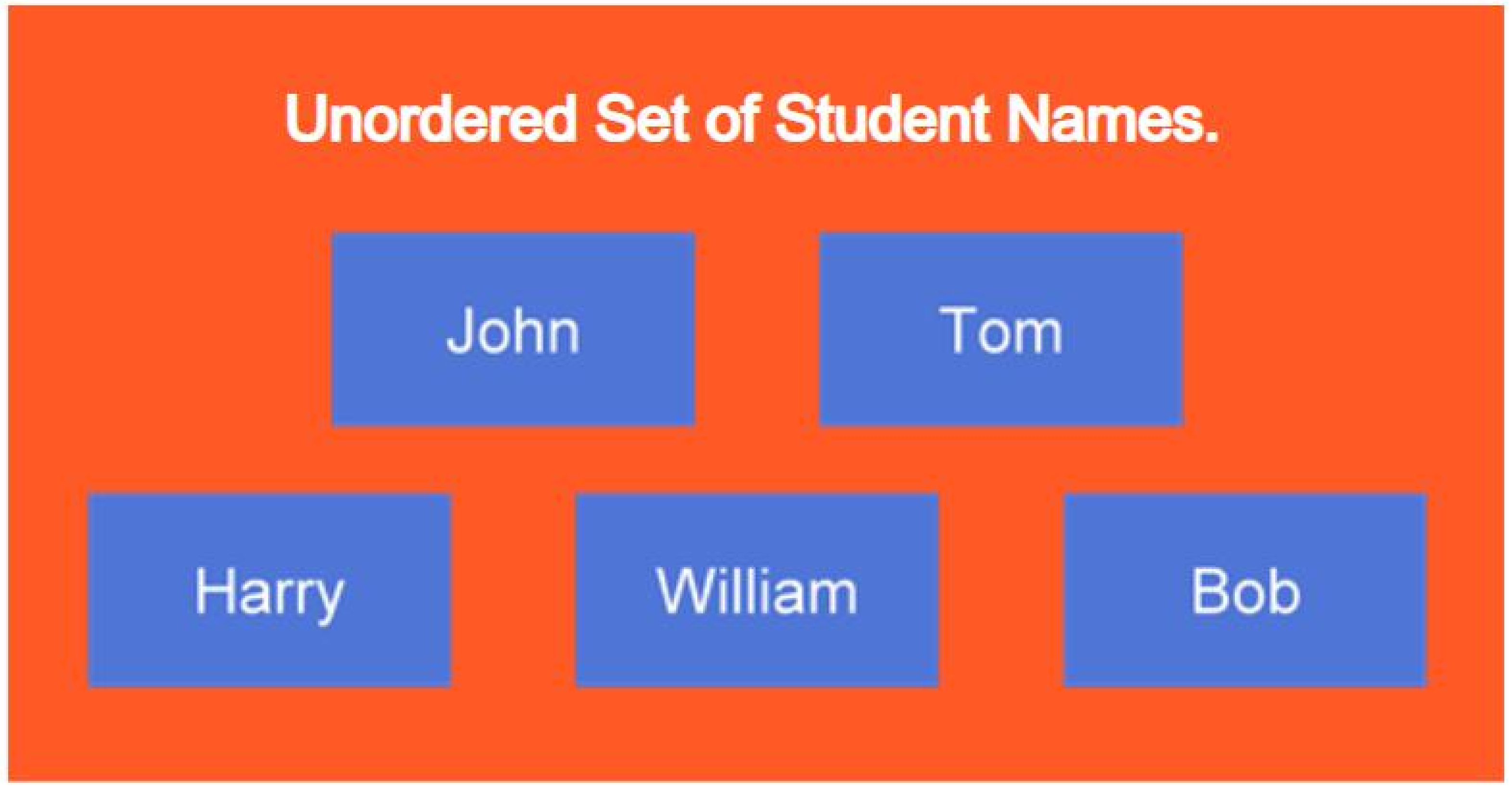
00:05 00:00:30 05/ 30

Set Interface

Slide Note

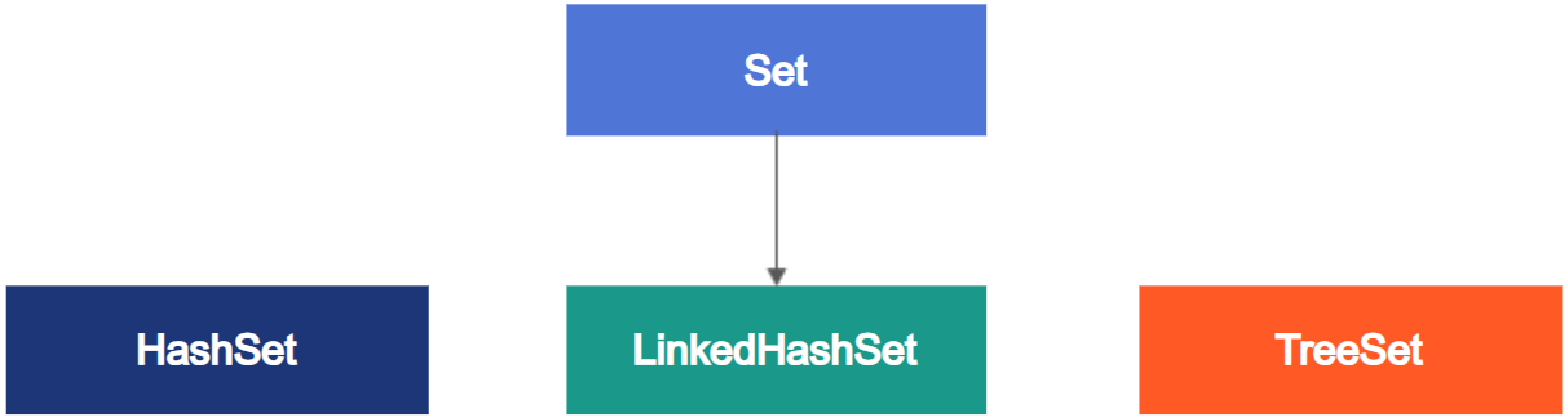
Set Interface

- Set is an interface that is part of the collection framework.
- It is an unordered collection of objects where duplicate values cannot be stored.
- Set has methods to add, remove, clear, and size to enhance the usage of this interface.
- There is no indexing on the elements of the set.
- Hence, a for loop cannot be used on a set, but the Iterator can be used.



Slide Note

Set Interface Implementation Classes



- The HashSet, LinkedHashSet, and TreeSet are the classes that inherit from the set interface and provide their implementations.
- HashSet- It enables you to create a set of unsorted elements with faster insertion.
- TreeSet- it enables you to create a sorted set of objects. The insert or add object process is slow as it sorts the objects on every insertion.
- LinkedHashSet– It maintains a linked list of the entries in the set in the order in which they were inserted.

Menu

00:08:00:00:30 08/ 30

Slide Note

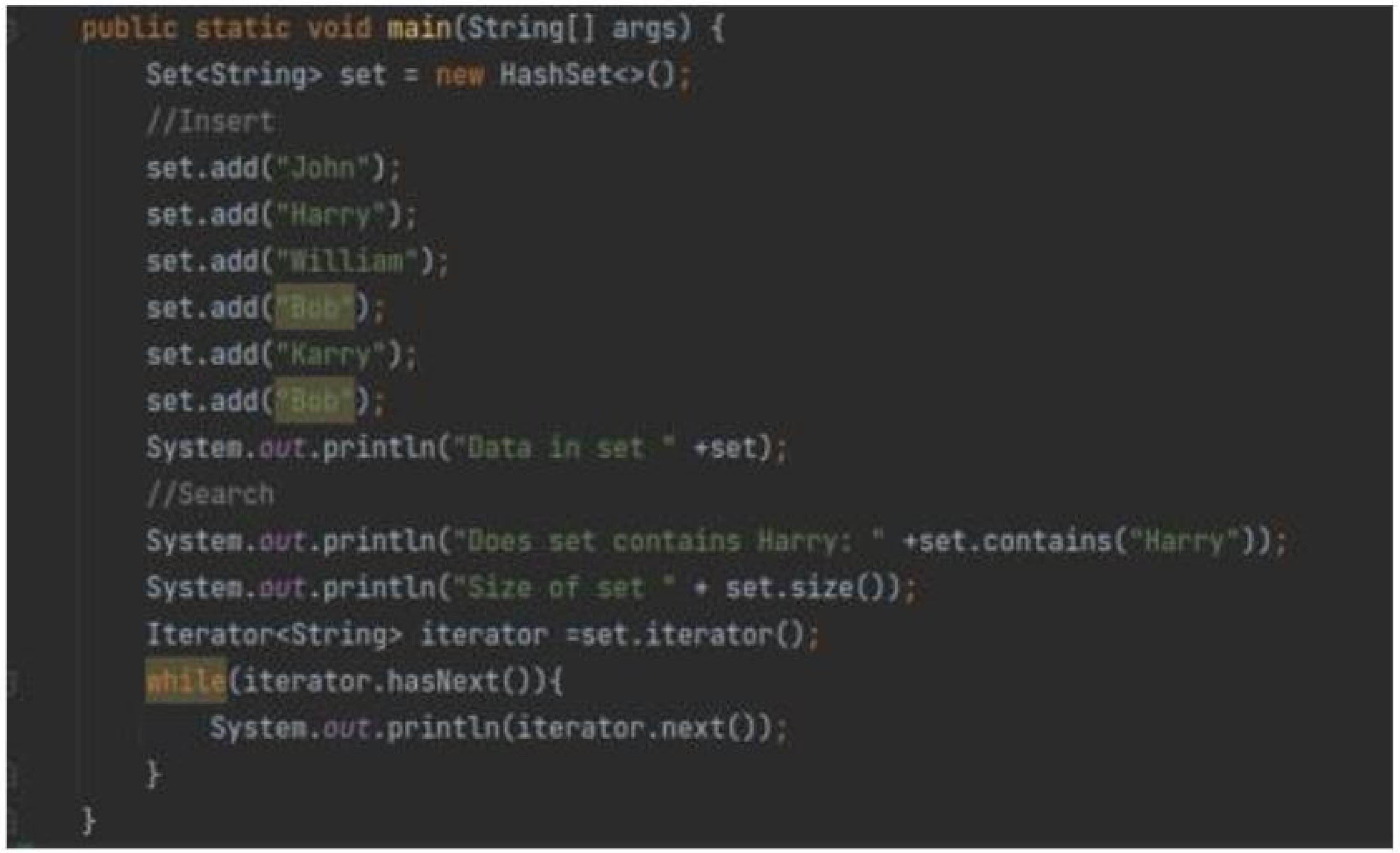
Implementing HashSet and TreeSet

Menu

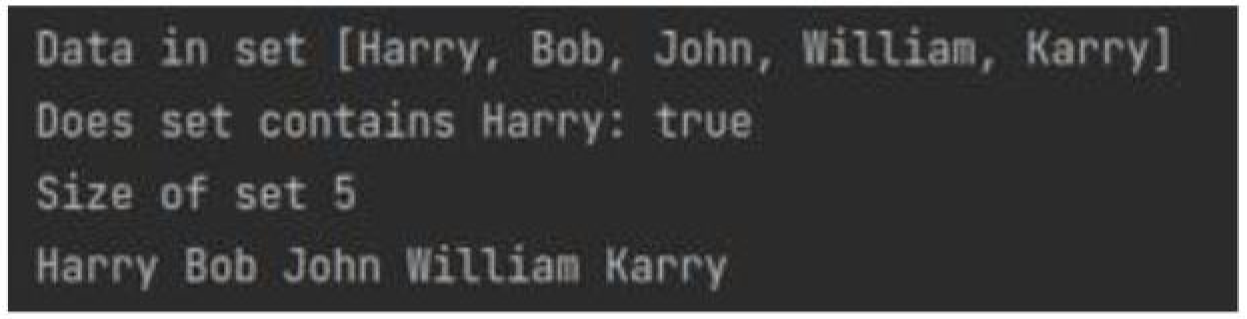
00:09 00:00:30 09/ 30

Slide Note

HashSet Implementation



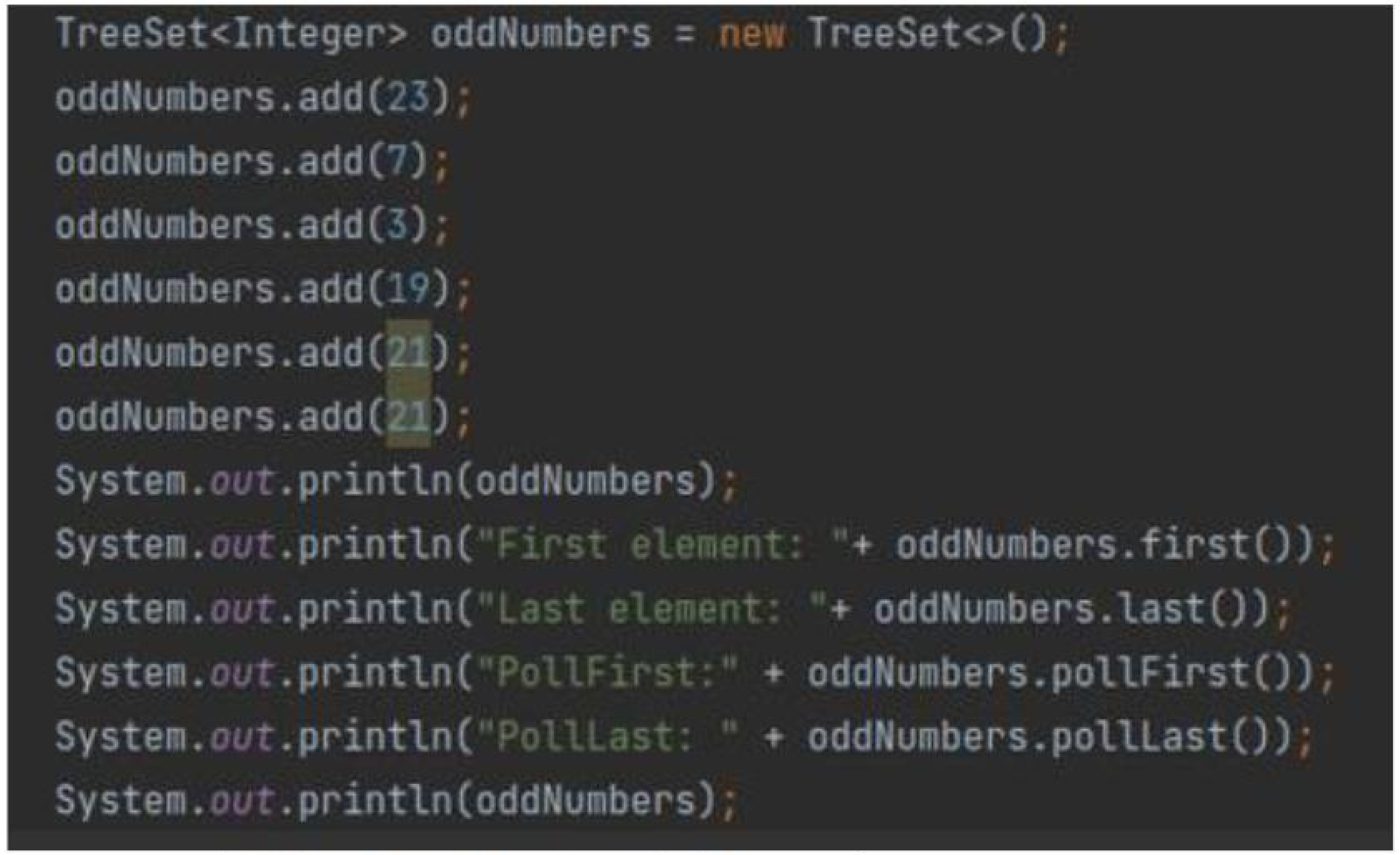
```
public static void main(String[] args) {
    Set<String> set = new HashSet<>();
    //Insert
    set.add("John");
    set.add("Harry");
    set.add("William");
    set.add("Bob");
    set.add("Karry");
    set.add("Bob");
    System.out.println("Data in set " +set);
    //Search
    System.out.println("Does set contains Harry: " +set.contains("Harry"));
    System.out.println("Size of set " + set.size());
    Iterator<String> iterator =set.iterator();
    while(iterator.hasNext()){
        System.out.println(iterator.next());
    }
}
```



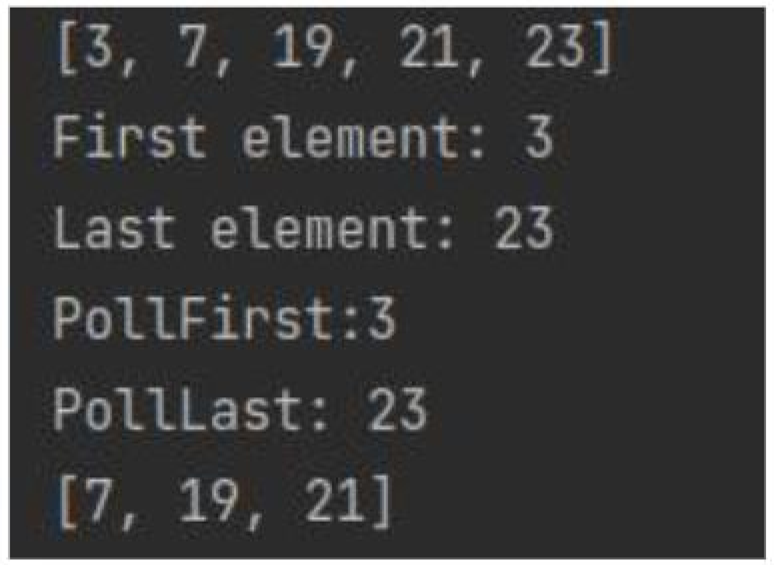
```
Data in set [Harry, Bob, John, William, Karry]
Does set contains Harry: true
Size of set 5
Harry Bob John William Karry
```

- Here we have created a HashSet object.Set is a reference variable, and HashSet is the implementation class.
- Boolean add(E e) - Adds the specified element to this set if it is not already present.
- Boolean contains (Object o) - This method will return true if the set contains the specified object.
- In the set, Bob is added two times.
- But when we iterate through the set and print the values, Bob is printed only once since duplicates are not stored in the set.
- The output produces an unordered and unique string object.

TreeSet Implementation



```
TreeSet<Integer> oddNumbers = new TreeSet<>();
oddNumbers.add(23);
oddNumbers.add(7);
oddNumbers.add(3);
oddNumbers.add(19);
oddNumbers.add(21);
oddNumbers.add(21);
System.out.println(oddNumbers);
System.out.println("First element: "+ oddNumbers.first());
System.out.println("Last element: "+ oddNumbers.last());
System.out.println("PollFirst:" + oddNumbers.pollFirst());
System.out.println("PollLast: " + oddNumbers.pollLast());
System.out.println(oddNumbers);
```



```
[3, 7, 19, 21, 23]
First element: 3
Last element: 23
PollFirst:3
PollLast: 23
[7, 19, 21]
```

The TreeSet contains only sorted elements:

- first()- Returns the first (lowest) element currently in this set.
- last() - Returns the last (highest) element currently in this set.
- pollFirst - Retrieves and removes the first (lowest) element or returns null if this set is empty.
- pollLast - Retrieves and removes the last (highest) element or returns null if this set is empty.

Slide Note

Understanding the Difference Between List and Set

Slide Note

List vs. Set

List	Set
A list is an ordered sequence of objects.	A set is an unordered sequence of objects.
It allows duplicate elements.	The set is unique and does not allow duplicate elements.
As the list has indexing, elements are accessed by their position.	There is no indexing in the set. So, elements cannot be accessed by their position.
A list can hold multiple null elements.	A set can only store one null element.

Menu

00:13:00:00:30 13/ 30

Slide Note

Quick Check

Which of the following classes enables you to create a collection of unique objects?

1. ArrayList
2. HashSet
3. LinkedList
4. Vector

Menu

Quick Check: Solution

Which of the following classes enables you to create a collection of unique objects?

1. ArrayList
2. **HashSet**
3. LinkedList
4. Vector

Student Object in Set

Create a class student with various attributes (name, roll number, total marks). Create getter and setter, parameterized constructor, and override the toString() method.
Create a class called HashSetDemo with different methods to do the following task:

1. Add all the students to the set.
2. Display only student names and their total marks.
3. Remove a student with a given roll number.
4. From the main method, call all the above methods.
5. Use Iterators for all the methods wherever required.

Click here for the solution.
The Intellij IDE must be used for the demonstration.

DEMO

Slide Note

Understanding Map Interface

Menu

00:17:00:00:30 17/ 30

Slide Note

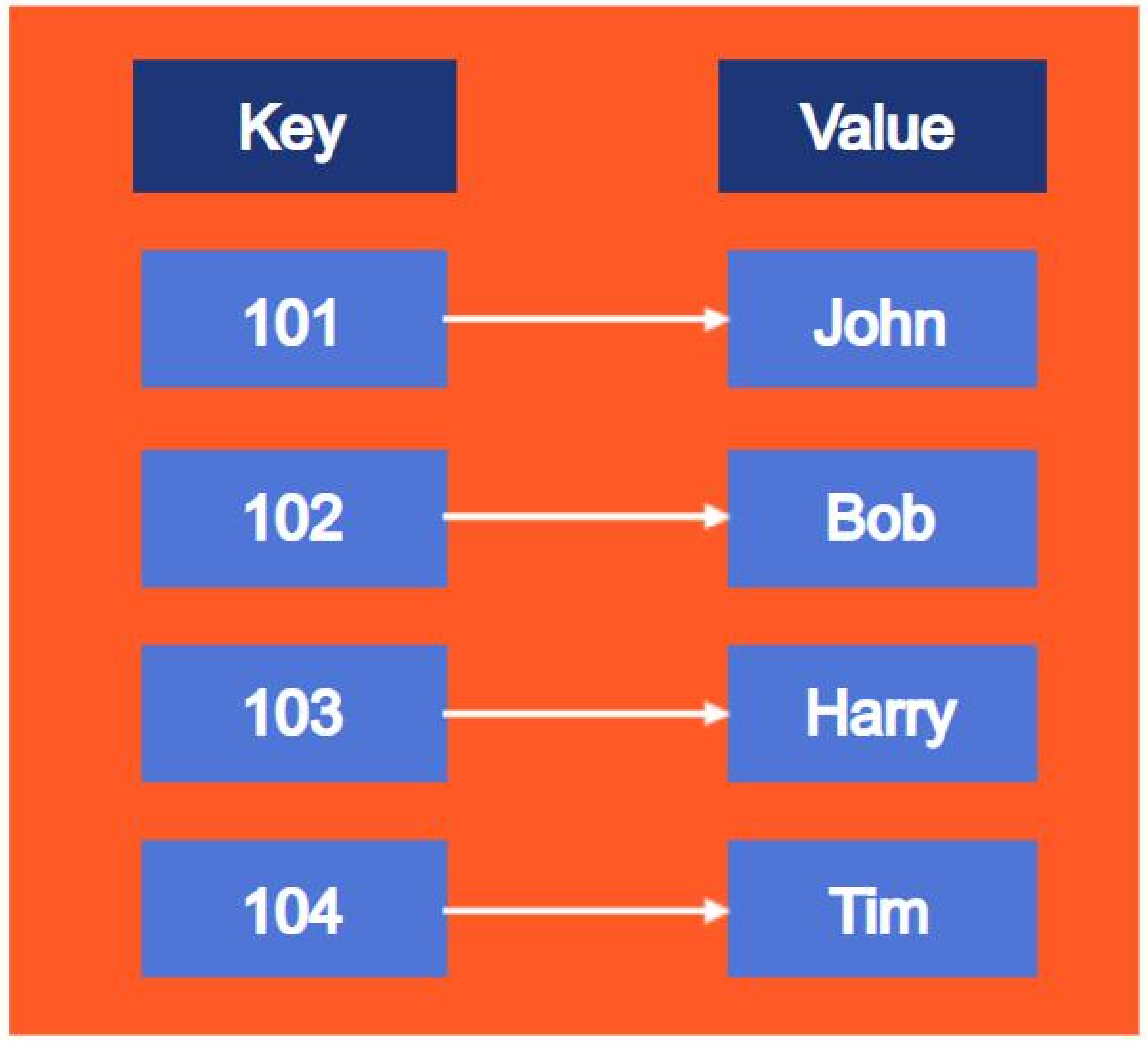
Map interface enables you to create a collection with key-value pair objects.

You need to use the key object to access the corresponding value object.

This interface allows you to duplicate the value objects, but the key object must be unique.

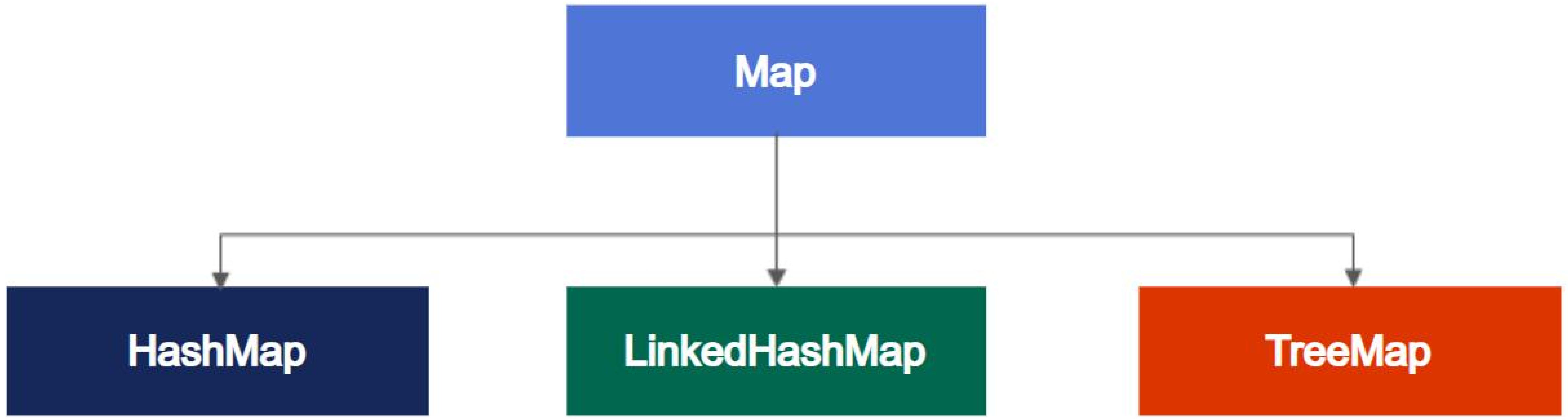
Map Interface

- The Map interface enables you to create a collection with key-value pair objects.
- The Map interface is not a subtype of the Collection interface.
- A map containing the key as an integer and the value as a string is shown below. The key is always unique.



Menu

Map Interface Implementation Classes



- The `HashMap`, `LinkedHashMap`, and `TreeMap` are the classes that inherit from the Map interface and provide their implementations.
- `HashMap`- It enables you to create a collection in which a value is accessible using a key. It allows us to store objects in an unordered form.
- `TreeMap`- The map is sorted according to the natural ordering of its keys. The insert or add object process is slow, as it sorts the objects on every insertion.
- `LinkedHashMap`– This implementation differs from HashMap in maintaining a doubly linked list running through all of its entries.

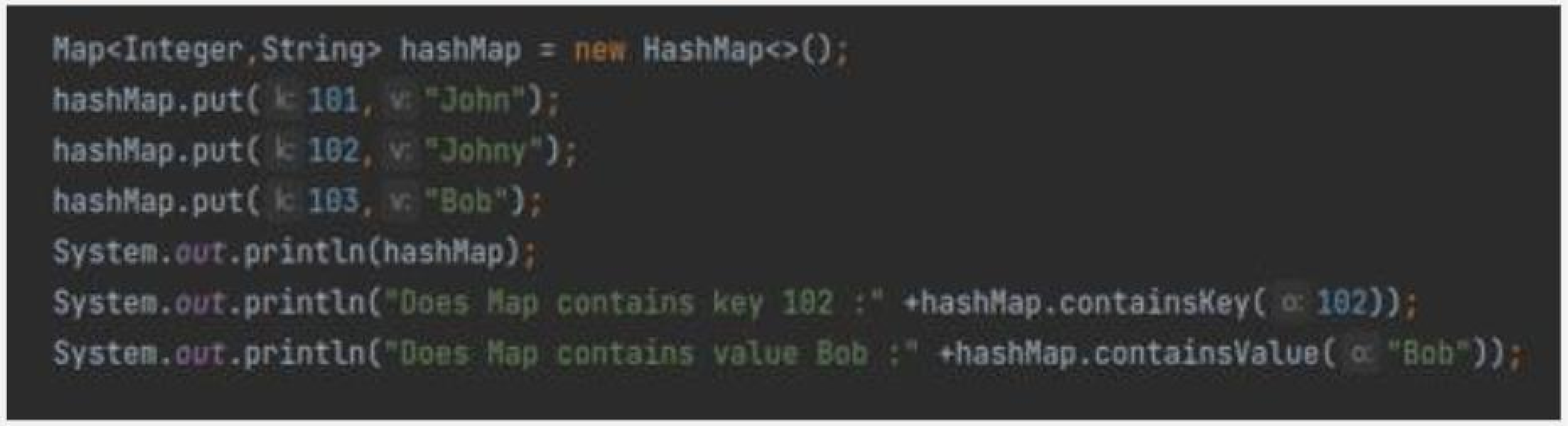
Slide Note

Implementing HashMap and TreeMap

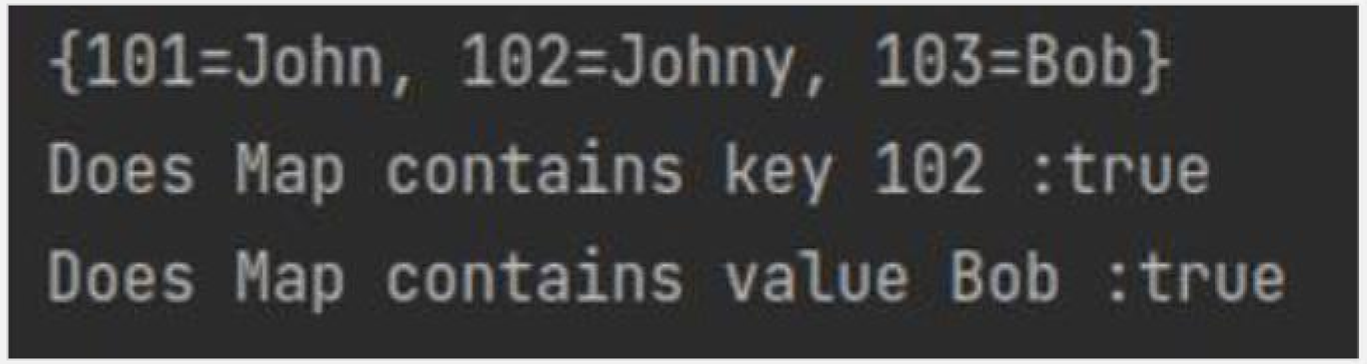
Menu

00:20:00:00:30 20/ 30

HashMap Implementation



```java
Map<Integer,String> hashMap = new HashMap<>();
hashMap.put( k: 101, v: "John");
hashMap.put( k: 102, v: "Johny");
hashMap.put( k: 103, v: "Bob");
System.out.println(hashMap);
System.out.println("Does Map contains key 102 :" +hashMap.containsKey( o: 102));
System.out.println("Does Map contains value Bob :" +hashMap.containsValue( o: "Bob"));
```



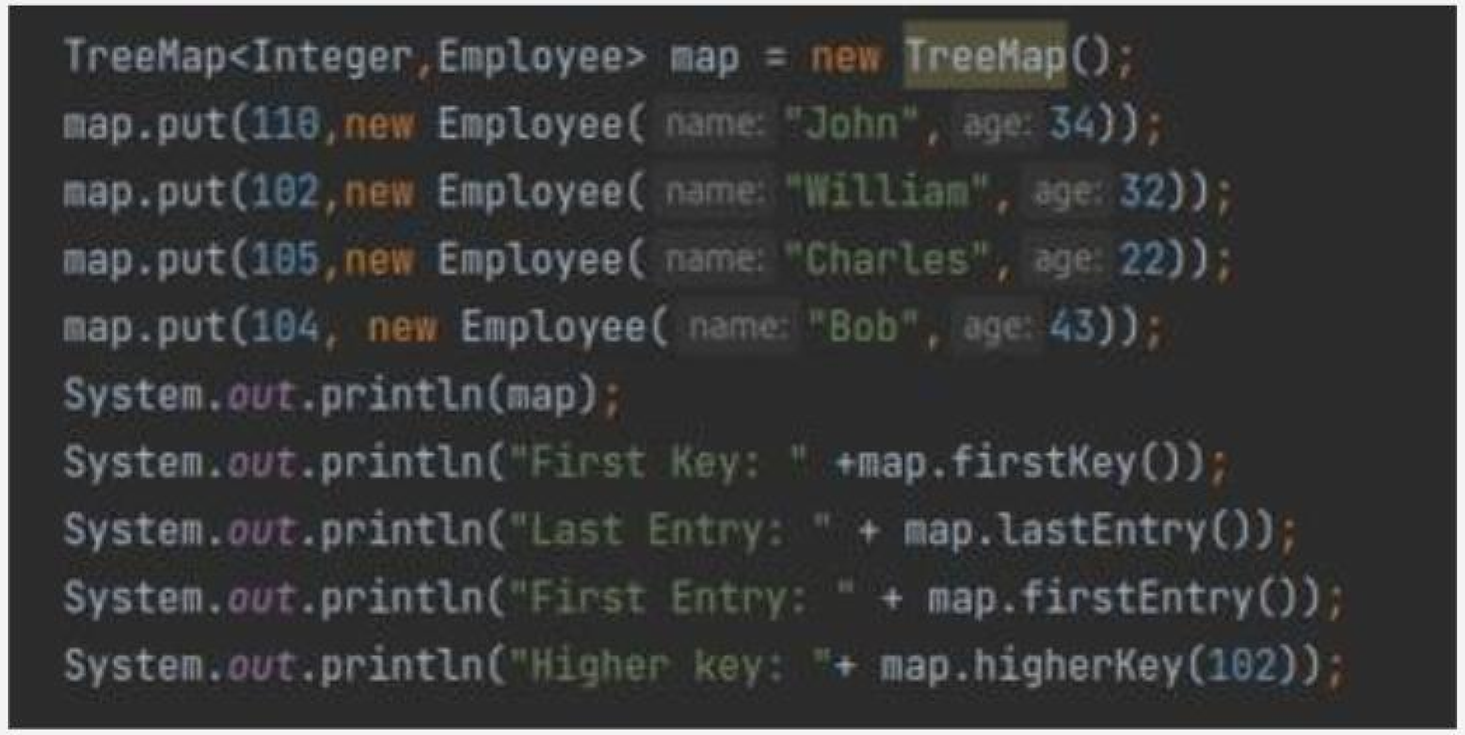
```
{101=John, 102=Johny, 103=Bob}
Does Map contains key 102 :true
Does Map contains value Bob :true
```

- HashMap having a key of type integer and value of type string is declared.
- Since the map does not extend the collection, the add method is not available to a map.

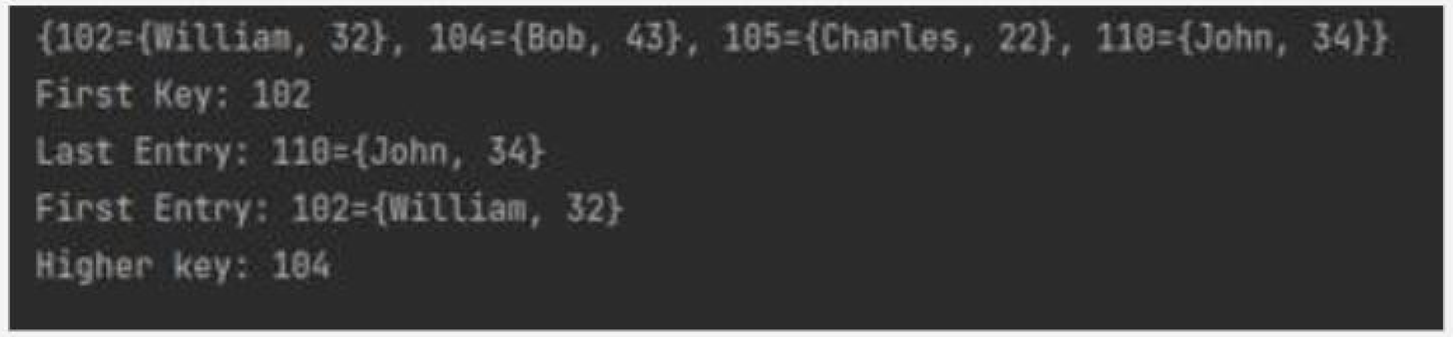
Methods of the Map Interface:

- Put (key,value) - Associates the specified value with the specified key in this map. If the map previously contained a mapping for the key, the old value is replaced with a new value.
- ContainsKey (Object key) - Returns true if this map contains a mapping for the specified key.
- containsValue(Object value)-Returns true if this map maps one or more keys to the specified value.

Slide Note



```
TreeMap<Integer,Employee> map = new TreeMap();
map.put(110,new Employee( name: "John", age: 34));
map.put(102,new Employee( name: "William", age: 32));
map.put(105,new Employee( name: "Charles", age: 22));
map.put(104, new Employee( name: "Bob", age: 43));
System.out.println(map);
System.out.println("First Key: " +map.firstKey());
System.out.println("Last Entry: " + map.lastEntry());
System.out.println("First Entry: " + map.firstEntry());
System.out.println("Higher key: "+ map.higherKey(102));
```



```
{102={William, 32}, 104={Bob, 43}, 105={Charles, 22}, 110={John, 34}}
First Key: 102
Last Entry: 110={John, 34}
First Entry: 102={William, 32}
Higher key: 104
```

TreeMap Implementation

- A TreeMap with key as an integer and value as a string is declared.
- As TreeMap extends SortedMap, the keys are in sorted order.

Methods of the TreeMap:

- firstKey() - Returns the first (lowest) key currently in this map.
- lastEntry() - Returns a key-value mapping associated with the greatest key in this map, or null if the map is empty.
- firstEntry() - Returns a key-value mapping associated with the least key in this map, or null if the map is empty.
- higherKey(Integer I) - Returns the key that is strictly greater than the given key, or null if there is no such key.

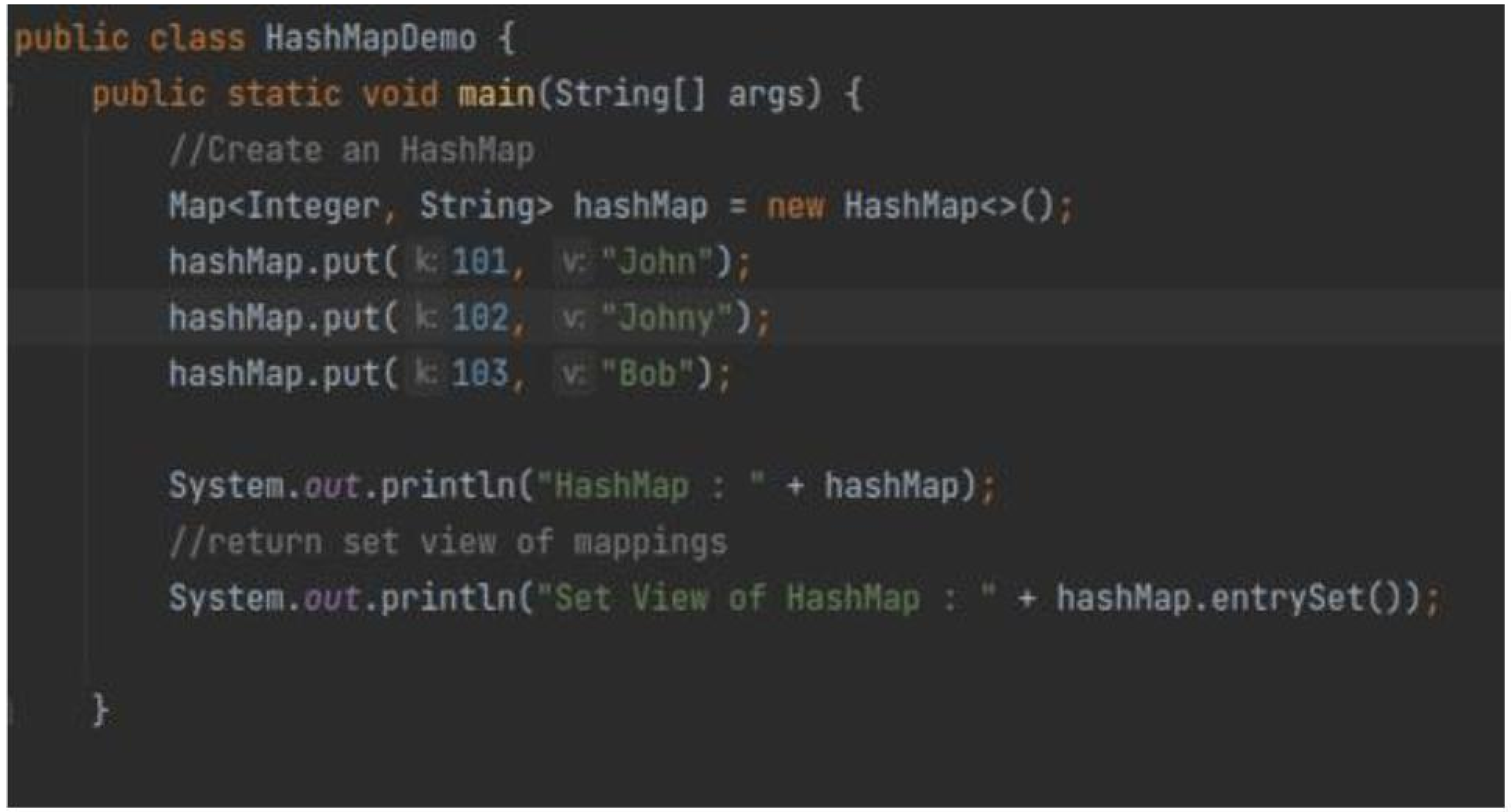
Menu

00:22 00:00:30 22/ 30

Slide Note

Iterating Through Map

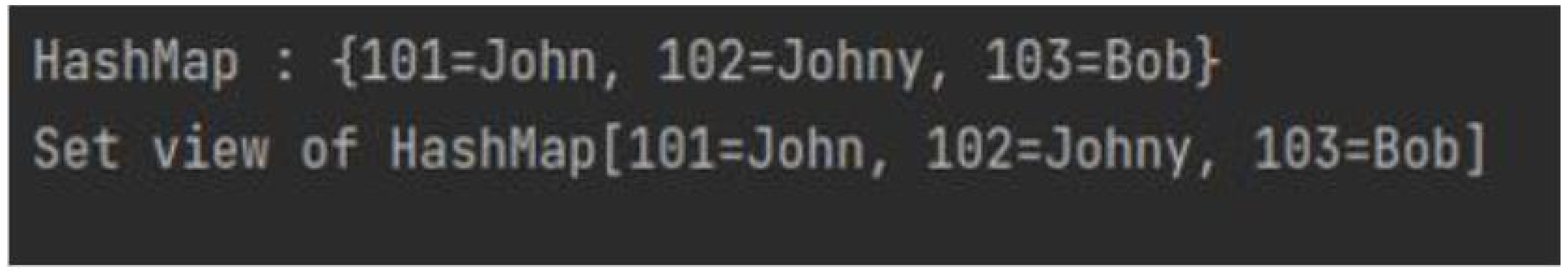
Getting Key and Value Entries From the Map



```
public class HashMapDemo {
    public static void main(String[] args) {
        //Create an HashMap
        Map<Integer, String> hashMap = new HashMap<>();
        hashMap.put( k: 101,  v: "John");
        hashMap.put( k: 102,  v: "Johny");
        hashMap.put( k: 103,  v: "Bob");

        System.out.println("HashMap : " + hashMap);
        //return set view of mappings
        System.out.println("Set View of HashMap : " + hashMap.entrySet());

    }
```

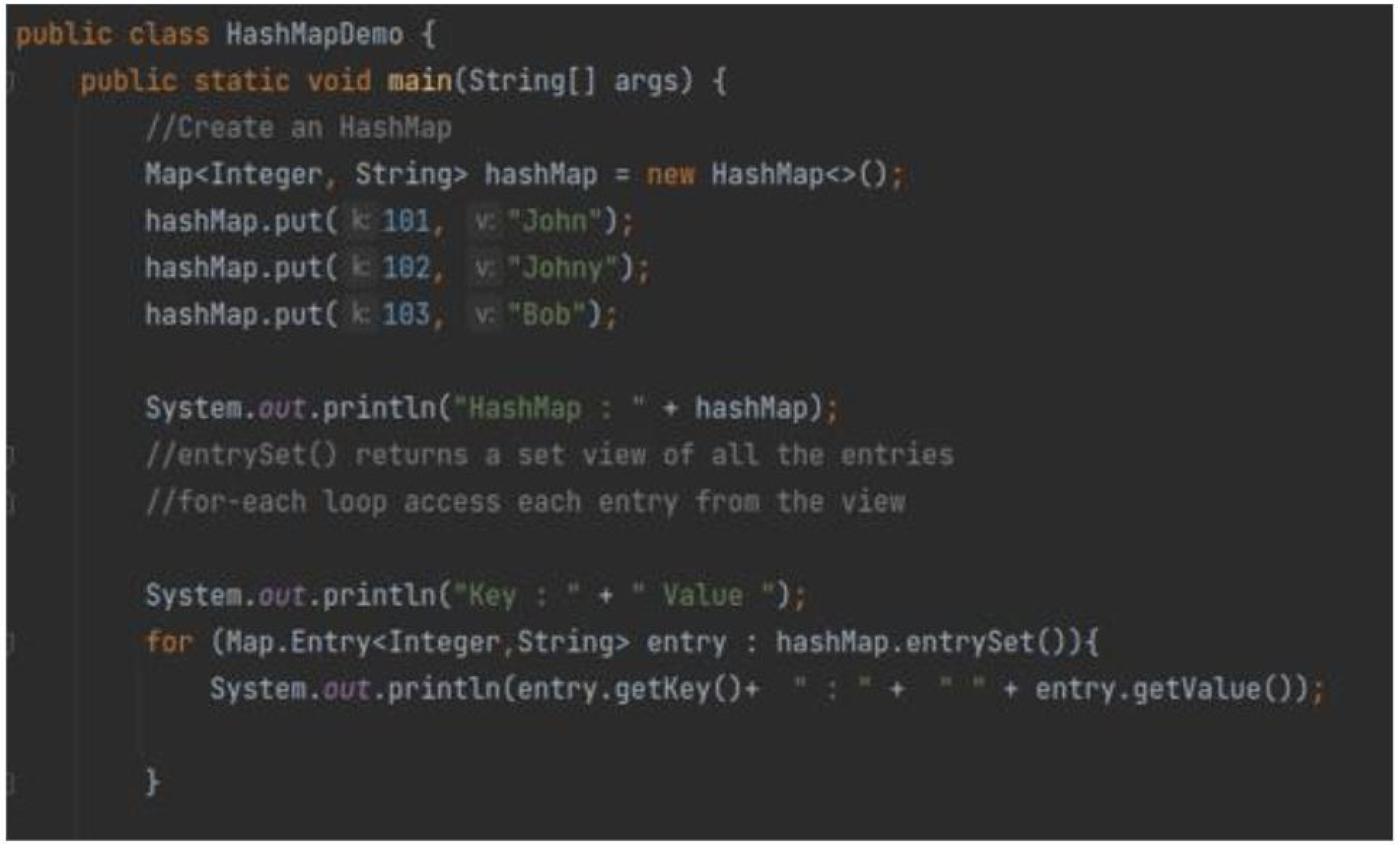


```
HashMap : {101=John, 102=Johny, 103=Bob}
Set view of HashMap[101=John, 102=Johny, 103=Bob]
```

- In this case, a HashMap of integer and string types is created.
- HashMap has a method entrySet() that returns a set view of all the mappings present in the HashMap.
- EntrySet() does not take any parameters but returns a set view of all the entries of a HashMap.

Note – The set view means all the entries of the HashMap are viewed as a set. Entries are not converted to a set.

Getting Key and Value Entries From the Map (cont'd)

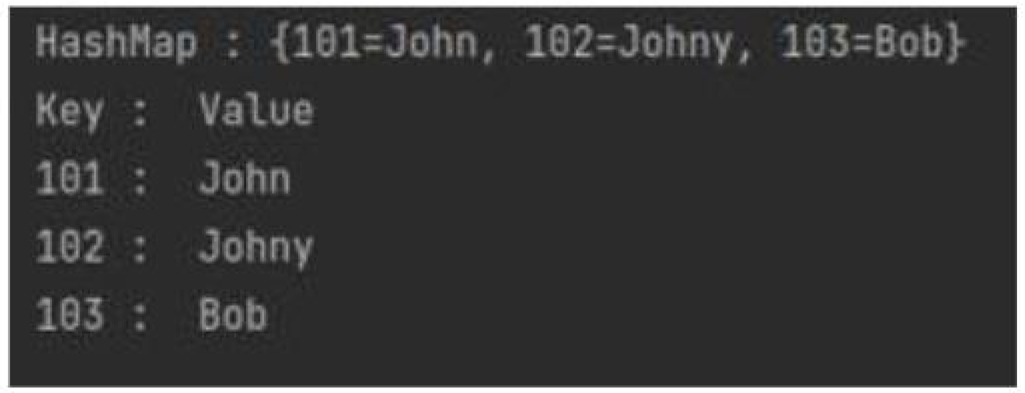


```
public class HashMapDemo {
    public static void main(String[] args) {
        //Create an HashMap
        Map<Integer, String> hashMap = new HashMap<>();
        hashMap.put( k: 101,  v: "John");
        hashMap.put( k: 102,  v: "Johny");
        hashMap.put( k: 103,  v: "Bob");

        System.out.println("HashMap : " + hashMap);
        //entrySet() returns a set view of all the entries
        //for-each loop access each entry from the view

        System.out.println("Key : " + " Value ");
        for (Map.Entry<Integer,String> entry : hashMap.entrySet()){
            System.out.println(entry.getKey()+  " : " +  " " + entry.getValue());

        }
```

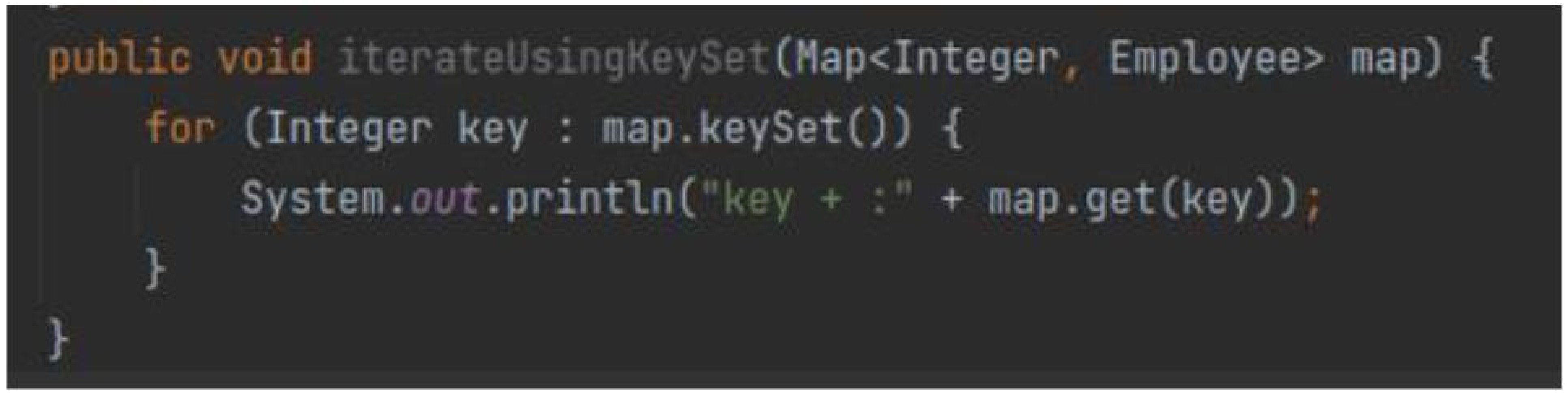


```
HashMap : {101=John, 102=Johny, 103=Bob}
Key :  Value
101 :  John
102 :  Johny
103 :  Bob
```

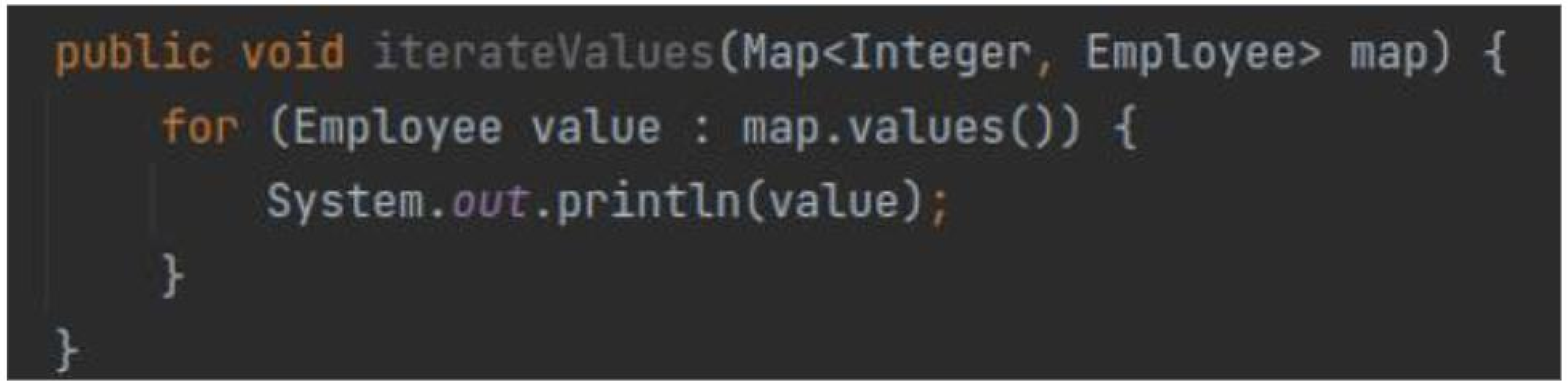
- Iterate through the map using a for-each loop to retrieve all the keys and their values.
- EntrySet returns a set view of the map, which is Map.Entry<k,V>.
- Iterating through the map. Entry to retrieve key and value pair using the entry getKey() and getvalue() methods.

Slide Note

Getting Key and Value Entries From the Map (cont'd)



```java
public void iterateUsingKeySet(Map<Integer, Employee> map) {
    for (Integer key : map.keySet()) {
        System.out.println("key + :" + map.get(key));
    }
}
```



```java
public void iterateValues(Map<Integer, Employee> map) {
    for (Employee value : map.values()) {
        System.out.println(value);
    }
}
```

- Map has the method keySet() which returns all the keys from the map.
- Map.values() is the method that returns all the values from the map.
- These methods are used with for loop to get all the keys and values from the map.

Slide Note

Quick Check

Which of the following methods can obtain a set of all keys on a map?

1. getAll()
2. getKeys()
3. keySet()
4. getAllKeySet()

Slide Note

Quick Check: Solution

Which of the following methods can obtain a set of all keys on a map?

1. getAll()
2. getKeys()
3. [✓ keySet()]
4. getAllKeySet()

Slide Note

Employee Details in HashMap

Create a class called "Employee with attributes (name, empId, department). Generate getter and setter functions, a parameterized constructor, and override the toString() method for all the attributes.
Create a MapEmployeeDemo class with different methods for the following tasks:

1. Add all employee objects inside the map object of type integer and employee. Here the key is an integer, which will start from 1, 2, and so on.
2. Display all the employee objects.
3. Display employees belonging to some specific department.

Click here for the solution.
The Intellij IDE must be used for the demonstration.

DEMO

Menu