SYNTAX TECHNOLOGIES

# JAVA

Class 32

## Agenda

Set Interface and HashSet Class

## Task

1. Create a class Insurance that will have an attribute as insuranceName and unimplemented behaviour as getQuote and cancelInsurance. Create 3 subclasses Car, Pet, Health. Car class has it's own attribute as carModel and Class Pet has petType attribute.

- Create 3 objects of the sub classes and store them in ArrayList. Using for loop/advanced for loop/ iterator access methods from different classes.

## Set

Set is a Collection that can’t contain duplicate elements.

There are three main implementations of Set interface:

- HashSet
- LinkedHashSet
- TreeSet

## Set

**HashSet**, which stores its elements in a hash table, is the best-performing implementation.

**TreeSet**, which stores its elements in a red-black tree, orders its elements based on their values; it is substantially slower than HashSet.

**LinkedHashSet**, which is implemented as a hash table with a linked list running through it, orders its elements based on the order in which they were inserted into the set.

## Hashset in Java

- **HashSet** is the class, Which implements the Set interface in Java.
- The HashSet does not add any additional methods beyond those found in the Set interface.
- **HashSet does not guarantee any insertion orders of the set** but it allows null elements.
- HashSet can be used in place of ArrayList to store the object if you require no duplicate and don’t care about insertion order.
- HashSet doesn’t allow duplicates. If you try to add a duplicate element in HashSet, the old value would be overwritten.

## Methods in Hashset

- **boolean add(Element  e)**: It adds the element e to the list.
- **void clear():** It removes all the elements from the list.
- **boolean contains(Object o):** It checks whether the specified Object o is present in the list or not. If the object has been found it returns true else false.
- **boolean isEmpty():** Returns true if there is no element present in the Set.
- **Iterator iterator():** Used to return an iterator over the element in the set.
- **int size():** It gives the number of elements of a Set.
- **Boolean remove (Object o):** It removes the specified Object from the Set.
- **Object clone():** This method returns a shallow copy of the HashSet.

**NOTE: A Set doesn't provide any method for data retrieval.**

```java
import java.util.HashSet;
public class HashSetExample {
  public static void main(String args[]) {
HashSet<String> hset = new
HashSet<String>();

    // Adding elements to the HashSet
    hset.add("Apple");
    hset.add("Mango");
    hset.add("Grapes");
    hset.add("Orange");
    hset.add("Fig");
    //Addition of duplicate elements
    hset.add("Apple");
    hset.add("Mango");
    //Addition of null values
    hset.add(null);
    hset.add(null);

    //Displaying HashSet elements
    System.out.println(hset);
  }
}
```

```java
import java.util.HashSet;
import java.util.Iterator;

class IterateHashSet{
  public static void main(String[] args) {
    // Create a HashSet
    HashSet<String> hset = new
HashSet<String>();

    //add elements to HashSet
    hset.add("Chaitanya");
    hset.add("Rahul");
    hset.add("Tim");
    hset.add("Rick");
    hset.add("Harry");

    Iterator<String> it = hset.iterator();
    while(it.hasNext()){
      System.out.println(it.next());
    }
  }
}
```

## Task

1. Create an HashSet of cities and add duplicates into it.

- Retrieve all values from hashset in 2 different ways.
- Retrieve all values in alphabetical order.

SYNTAX TECHNOLOGIES

# JAVA

Class 33

## Agenda

Map Interface and HashSet Class

## Task

How can you remove all duplicates from ArrayList?

```
List<String> aList=new ArrayList<String>();
        aList.add("John");
        aList.add("Jane");
        aList.add("James");
        aList.add("Jasmine");
        aList.add("Jane");
        aList.add("James");
```

## Map

A Map is an object that maps keys to values.

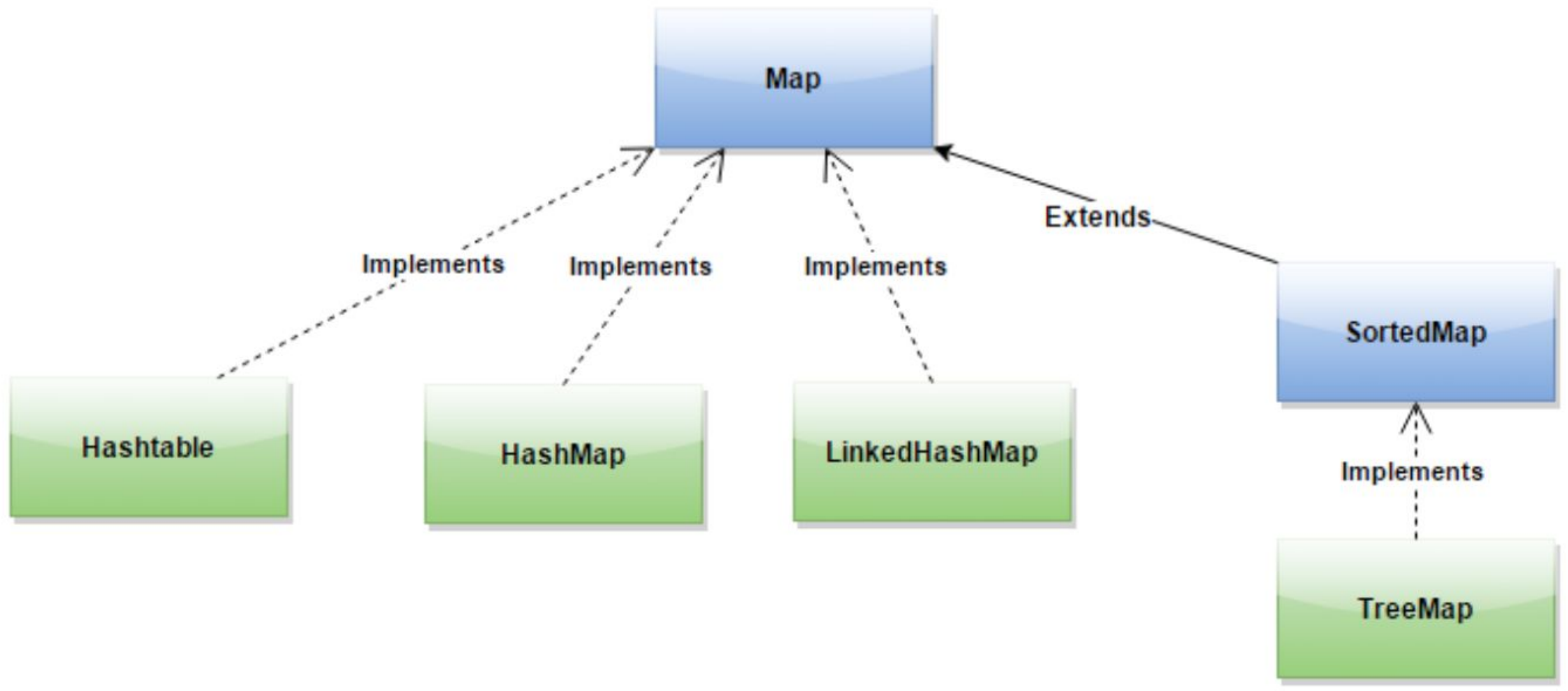
Map stores the values in Key-Value pair and each pair is known as an entry.

Values in the Map can be duplicate but keys always must be unique

The Map interface is implemented by different Java classes, such as HashMap, HashTable, and TreeMap.
Each class provides different functionality and can be either synchronized or not.

## Map

Maps are not part of collection but built based on the collection concepts



## Map

A map has the form Map <K, V> where:

**K : Specifies the Keys**
**V : Specifies the Values**

Main implementations of Map interfaces.

HashMap: it makes no guarantees concerning the order of iteration, HashMap doesn’t maintain the insertion order of elements.

LinkedHashMap: It orders its elements based on the order in which they were inserted into the set (insertion-order).

TreeMap: It stores its elements in a red-black tree, orders its elements based on their values; it is substantially slower than HashMap.

## Methods in Map

- **public Object put(object key,Object value)**: This method is used to insert an entry in this map.
- **public void putAll(Map map)**: This method is used to insert the specified map in this map.
- **public Object remove(object key)**: This method is used to delete an entry for the specified key.
- **public Object get(Object key)**: This method is used to return the value for the specified key.
- **public boolean containsKey(Object key)**: This method is used to search the specified key from this map.
- **public boolean containsValue(Object value)**: This method is used to search the specified value from this map.

# HashMap in Java

- HashMap class implements the Map interface in Java.
- HashMap stores the elements in Key-Value pair.
- HashMap contains only unique elements i.e You can’t use the same key-value pair again.
- HashMap may have one null key and multiple null values.
- HashMap maintains no order.
- It is not an ordered collection which means it does not return the keys and values in the same order in which they have been inserted into the HashMap.
- It neither does any kind of sorting to the stored keys and Values.
- You must need to import java.util.HashMap or its super class in order to use the HashMap class and methods.

## HashMap Class Methods

| |
|---|
| **void clear(): It removes all the key and value pairs from the specified Map.** |
| **Object clone()**: It returns a copy of all the mappings of a map and used for cloning them into another map. |
| **boolean containsKey(Object key)**: It is a boolean function which returns true or false based on whether the specified key is found in the map. |
| **boolean containsValue(Object Value)**: Similar to containsKey() method, however it looks for the specified value instead of key. |
| **Value get(Object key)**: It returns the value for the specified key. |
| **boolean isEmpty()**: It checks whether the map is empty. If there are no key-value mapping present in the map then this function returns true else false. |

# HashMap Class Methods

**Set .keySet(): It returns the Set of the keys fetched from the map.**

**value .put(Key k, Value v)**: Inserts key value mapping into the map. Used in the above example.

**int .size()**: Returns the size of the map – Number of key-value mappings.

**Collection .values()**: It returns a collection of values of map.

**Value .remove(Object key)**: It removes the key-value pair for the specified key. Used in the above example.

**void .putAll(Map m): Copies all the elements of a map to the another specified map.**

## HashMap

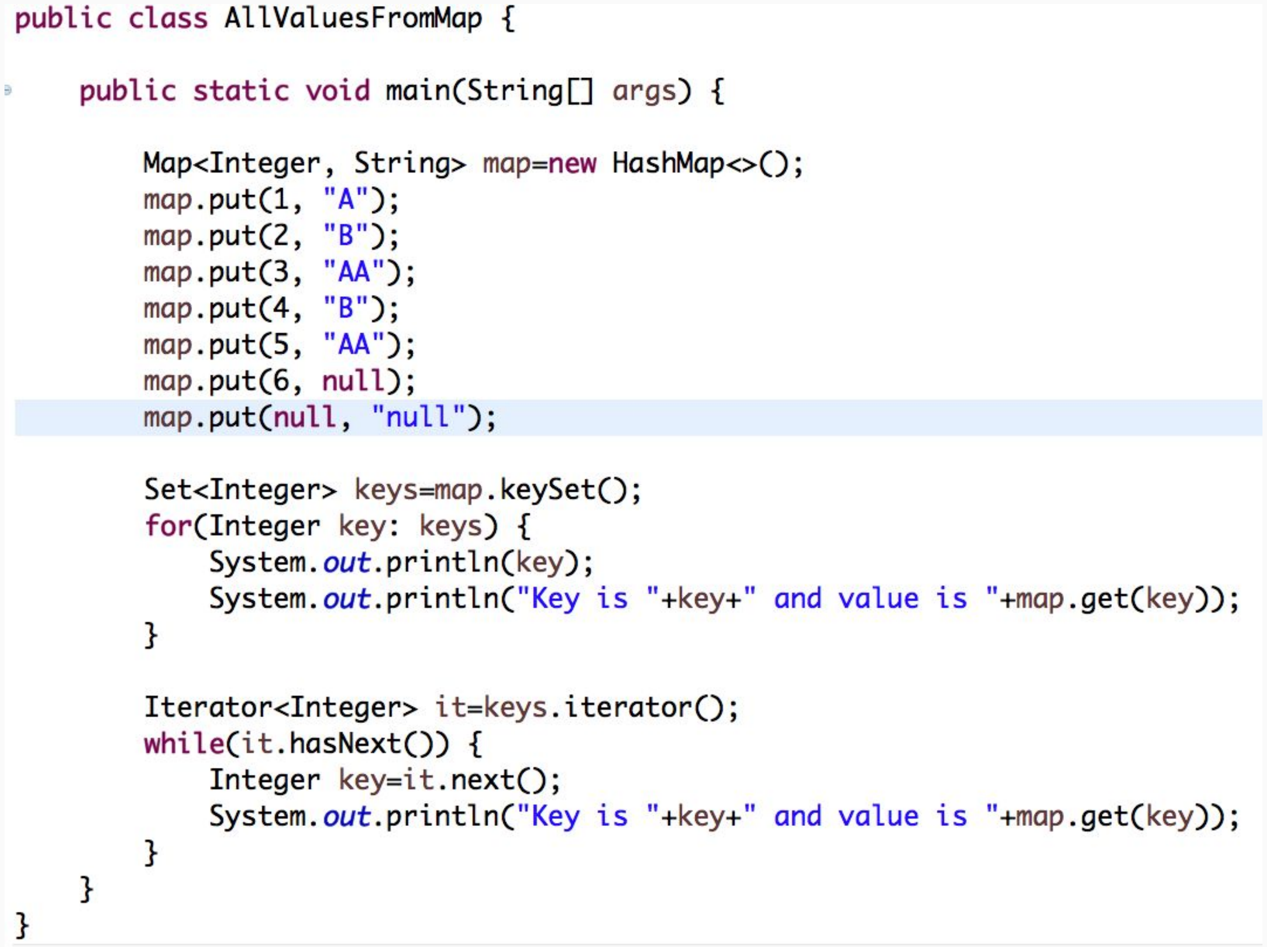
```
Map<Integer, String> studentMap=new HashMap<>();

studentMap.put(101, "John");
studentMap.put(102, "Jason");
studentMap.put(103, "Jordan");
studentMap.put(104, "Jenny");

//to check if specific key or value exist
System.out.println(studentMap.containsKey(101));
System.out.println(studentMap.containsValue("Jordan"));
//to access 1 value
System.out.println(studentMap.get(102));
//replace value
studentMap.replace(104, "Nikky");
System.out.println(studentMap);
//remove object
studentMap.remove(103);
System.out.println(studentMap);
```

## Task

1. Create a map of a building. Store floor number and it is associated company name. (Example: 1= Google, 2=Syntax etc..). Insert 7 entries with duplicate keys and values.

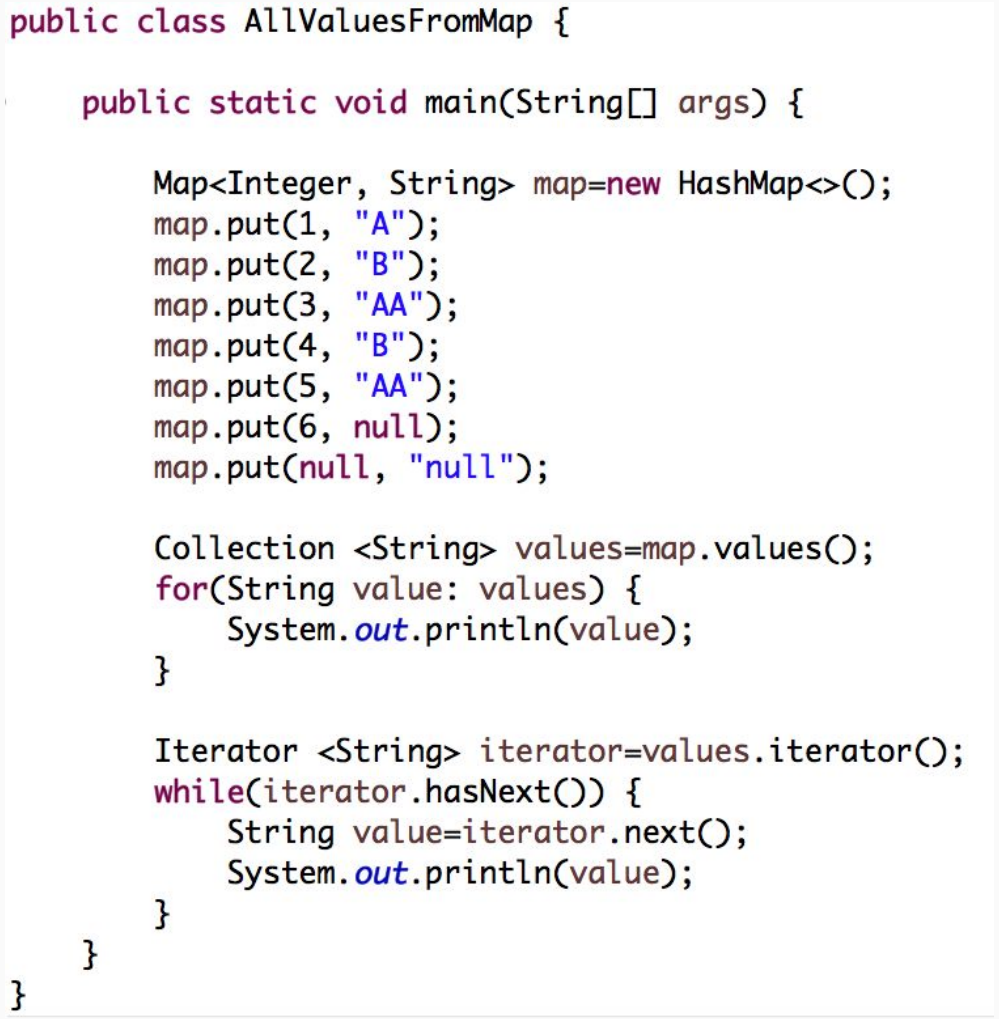
- Check how many entries you have?
- Update company on a 4th floor
- Remove company on the 7th floor
- Print your map

## How to get all Keys from a HashMap



```java
public class AllValuesFromMap {

    public static void main(String[] args) {

        Map<Integer, String> map=new HashMap<>();
        map.put(1, "A");
        map.put(2, "B");
        map.put(3, "AA");
        map.put(4, "B");
        map.put(5, "AA");
        map.put(6, null);
        map.put(null, "null");

        Set<Integer> keys=map.keySet();
        for(Integer key: keys) {
            System.out.println(key);
            System.out.println("Key is "+key+" and value is "+map.get(key));
        }

        Iterator<Integer> it=keys.iterator();
        while(it.hasNext()) {
            Integer key=it.next();
            System.out.println("Key is "+key+" and value is "+map.get(key));
        }
    }
}
```

## How to get all Values from a HashMap



```java
public class AllValuesFromMap {

    public static void main(String[] args) {

        Map<Integer, String> map=new HashMap<>();
        map.put(1, "A");
        map.put(2, "B");
        map.put(3, "AA");
        map.put(4, "B");
        map.put(5, "AA");
        map.put(6, null);
        map.put(null, "null");

        Collection <String> values=map.values();
        for(String value: values) {
            System.out.println(value);
        }

        Iterator <String> iterator=values.iterator();
        while(iterator.hasNext()) {
            String value=iterator.next();
            System.out.println(value);
        }
    }
}
```

## Task

1. Create a map of countries with its capital.

- Print all keys and values from a country map using for each loop and iterator.
- Print all values from a country map using for each loop and iterator.

Break till 1;35

## Search in HashMap

```
public class CheckKeyExample {

 public static void main(String[] args) {
HashMap<Integer, String> hashmap = new
HashMap<Integer, String>();

    hashmap.put(11,"Chaitanya");
    hashmap.put(22,"Pratap");
    hashmap.put(44,"Rajesh");
    hashmap.put(55,"Kate");

    boolean flag = hashmap.containsKey(22);
    SOP("Key 22 exists in HashMap? : " + flag);

    boolean flag2 = hashmap.containsKey(55);
    SOP("Key 55 exists in HashMap? : " + flag2);

    boolean flag3 = hashmap.containsKey(99);
    SOP("Key 99 exists in HashMap? : " + flag3);
 }
}
```

```
public class CheckValueExample {

 public static void main(String[] args) {

HashMap<Integer, String> hashmap = new
HashMap<Integer, String>();

    hashmap.put(11,"Chaitanya");
    hashmap.put(22,"Pratap");
    hashmap.put(33,"Singh");
    hashmap.put(44,"Rajesh");
    hashmap.put(55,"Kate");

boolean flag =
hashmap.containsValue("Singh");
    SOP(" String Singh exists in HashMap? : " +
flag);
 }
}
```

# How to loop HashMap in java

```
Map<String, Integer> classroomMap=new LinkedHashMap<>();
        classroomMap.put("Table", 20);//Entry<Key, Value>
        classroomMap.put("Chair", 60);//Entry<String, Integer>
        classroomMap.put("Screen", 3);
        classroomMap.put("Student", 60);
        classroomMap.put("Instructor", 3);

        System.out.println(classroomMap);
        for( Map.Entry<String, Integer> entry:classroomMap.entrySet()) {
                System.out.println(entry.getKey()+"="+entry.getValue());
        }

        Iterator<Map.Entry<String, Integer>>
classRoomIterator=classroomMap.entrySet().iterator();
        while(classRoomIterator.hasNext()) {
                Map.Entry<String, Integer> entry=classRoomIterator.next();
                System.out.println(entry.getKey()+"="+entry.getValue());
        }
```

## Task

1. Create a map of Best Buy store. Place item id and item name into it. Example (7664847 = Printer, 7879885= TV etc )

- Print all keys and values from a Best Buy map using EntrySet..