Maps

Click to Continue

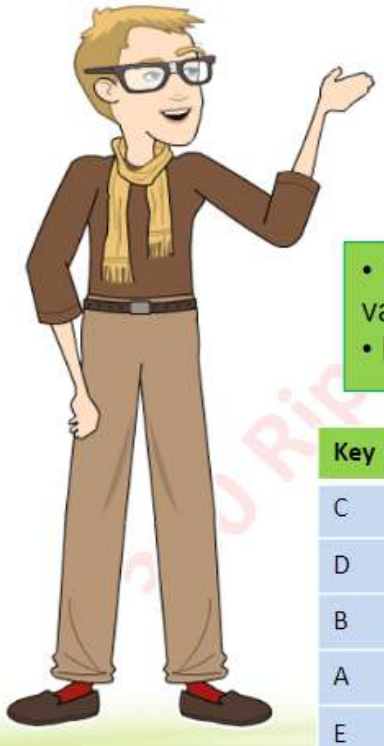
Map stores Elements as key – value pairs.

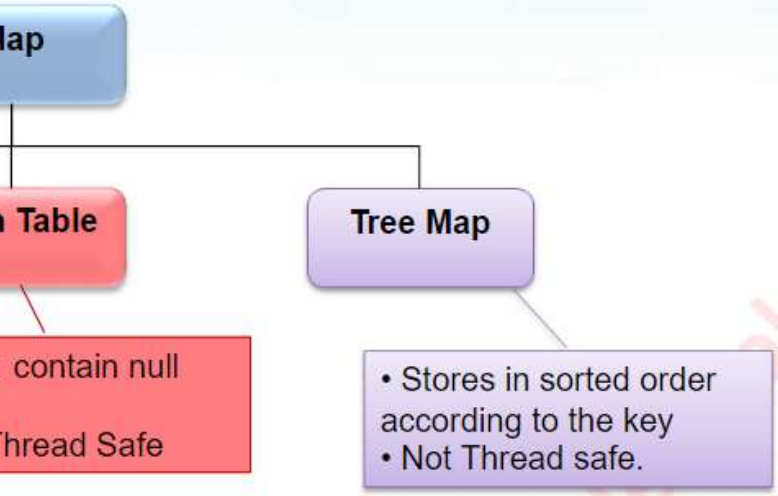
- Each key maps to one value stored in the map.
- The values can be accessed by passing the key.
- The key should be unique used to identify the value.
- The key and value can only be objects.

Key	Value
1	India
2	Australia
3	England
4	South Africa
5	New Zealand

Map Interfaces

Click to Continue





Key	Value
C	Cat
D	Dog
B	Null
A	Null
E	Bird

Key	Value
C	Cat
D	Dog
B	pigeon
A	cow
E	Bird

Key	Value
A	Null
B	Null
C	Cat
D	Dog
E	Bird

Map Interfaces

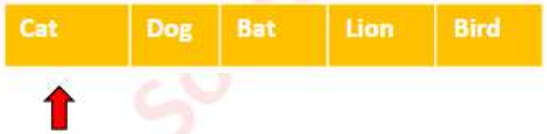
Click to Continue

- ***Map*** interface is the root interface in the map hierarchy
- All map implementations should implement this interface.

Collection Vs Map

Click to Continue

Collection



How Lion look up happens?
Click next for the animation.

Map

Key	Value
C	Cat
D	Dog
B	Bat
L	Lion
Bi	Bird

Collection Vs Map

Click to Continue

Collection

Map

Key	Value
C	Cat
D	Dog
B	Bat
L	Lion
Bi	Bird

In the case of collection we can see that the search traverses across the entire collection to get the required object. But in the case of map it is directly accessed using the key.

HashMap

Click to Continue

- Implementation of the ***Map*** interface.
- Permits null values for key(**one null key**) and value (**multiple null values**).
- Not Thread safe.
- Does not maintain the order of elements in the Map.

Syntax : Map<Type1,Type2> mapName=new HashMap<Type1,Type2>();

Creates a empty map with generics defined key of the type Type1 and value of the type Type2.

Example: Map<Integer,String> myMap=new HashMap<Integer,String>();

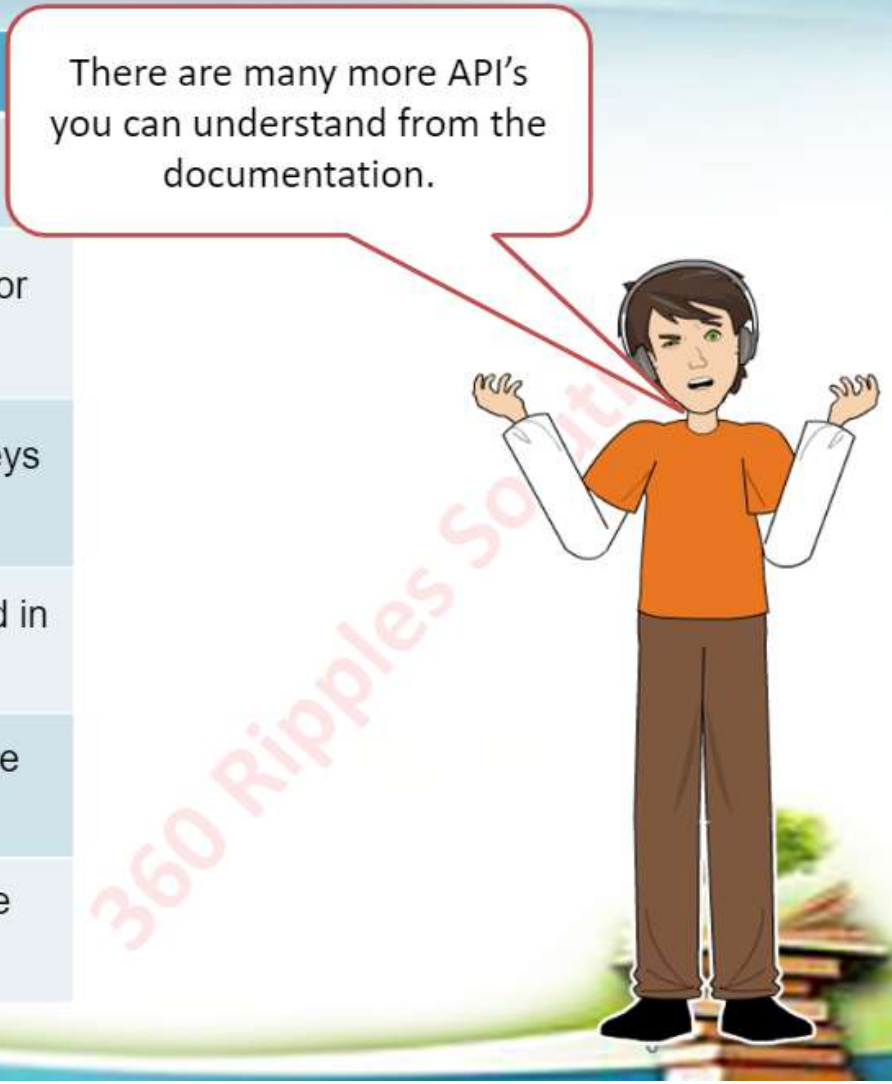
Creates map object storing Integer key and String value.

Always remember to declare the Map using the appropriate interface. In the example it is *Map*.

Map Interface

Click to Continue

Methods	Description
void clear()	Removes all mappings from this map.
boolean containsKey(Object key)	Returns true if this map contains a mapping for the specified key.
boolean containsValue(Object value)	Returns true if this map maps one or more keys to the specified value.
Set entrySet()	Returns a set view of the mappings contained in this map.
Object get(Objectkey)	Returns the value to which this map maps the specified key.
boolean isEmpty()	Returns true if this map contains no key-value mappings.



Add elements in a HashMap

Click to Continue

Elements can be added to a map as key - value pairs using the **put()** method.

Syntax : map.put(key,value);

Example: map.put(1,"India");

This adds an country ***India*** as value with a key 1.

Retrieve elements from a HashMap

Click to Continue

Elements can be retrieved from a map by passing the key using the **get()** method.

Syntax : variable =map.get(key);

Example: String country=map.get(1);

This retrieves the country specific to key 1 which is "***India***".

keySet() method

Click to Continue

keySet() method is used to returns the set of keys for the map.

- Using the keys obtained one can iterate the map.

```
Set<Integer> keys=map.keySet();
Iterator<Integer> iterator=keys.iterator();
while(iterator.hasNext())
{
  Integer key = iterator.next();
 System.out.println(map.get(key));
}
```

Get the keys of the Map.

Get the iterator of the Set.

Iterating the set and printing the key values.

You might be confused. Don't worry! As you practice you will become confident.

Try it Out - HashMap

Click to Continue

Objective: Let us learn to store values in a map, iterate through the map and print the values.

Problem Statement 1: Create a method that returns a Hash map containing the details of students. Roll no is the key and student name is the value.

Problem Statement 2 : Create a main method that reads the student map and prints the roll number and name of all the students by iterating through the keys.

Try it out Solution – Problem Statement 1

Click to Continue

Create a class named MapEx. It should loads a map with student registration number as key and student name as value and returns the student map.

```
public Map<String,String> getStudentDetails(){
    Map<String,String> studentMap=new HashMap<String, String>();
    studentMap.put("2","Arun");
    studentMap.put("3","Prithvi");
    studentMap.put("1","Tom");
    studentMap.put("4","Irfan");
    return studentMap;
}
}
```

Always remember to declare the Map using the appropriate interface. In the example it is ***Map***.

Try it out solution – Problem Statement 2

Click to Continue

Let us now iterate through the map and print the name and the registration number.

```
public class MapExMain {
    public static void main(String args[]) {
        MapEx ex1 = new MapEx();
        Map<String, String> studentMap = ex1.getStudentDetails();
        System.out.println(studentMap);
        Set<String> keys = studentMap.keySet();
        Iterator<String> iterator = keys.iterator();
        System.out.println("Students Details");
        while (iterator.hasNext()) {
            String rollNo = iterator.next();
            String name = studentMap.get(rollNo);
            System.out.println("Roll No : " + rollNo + " Name : " + name);
        }
    }
}
```

Prints the map

Gets the key Set for the map

Gets the iterator for the keyset

Iterates over the key set and reads the value for each key.

Try it out solution – Problem Statement 2

Click to Continue

Let us now iterate through the map and print the name and the registration number.

```
public class MapExMain {
    public static void main(String args[]) {
        MapEx ex1 = new MapEx();
        Map<String, String> studentMap = ex1.getStudentDetails();
        System.out.println(studentMap);
        Set<String> keys = studentMap.keySet();
        Iterator<String> iterator = keys.iterator();
        System.out.println("Students Details");
        while (iterator.hasNext()) {
            String rollNo = iterator.next();
            String name = studentMap.get(rollNo);
            System.out.println("Roll No : " + rollNo + " Name : " + name);
        }
    }
}
```

Prints the map

Gets the key Set for the map

Gets the iterator for the keyset

Iterates over the key set and reads the value for each key.

Iteration could be confusing. You will become confident as you practice many programs.

Try it out - Output

Click to Continue

Try it out - Output

Click to Continue

What can be done to automatically sort the map according to the registration number ?

The solution is ***TreeMap***, since Tree implements the SortedMap interface, it automatically stores the elements in a sorted order based on the key.

Try it out – TreeMap Demo

Click to Continue

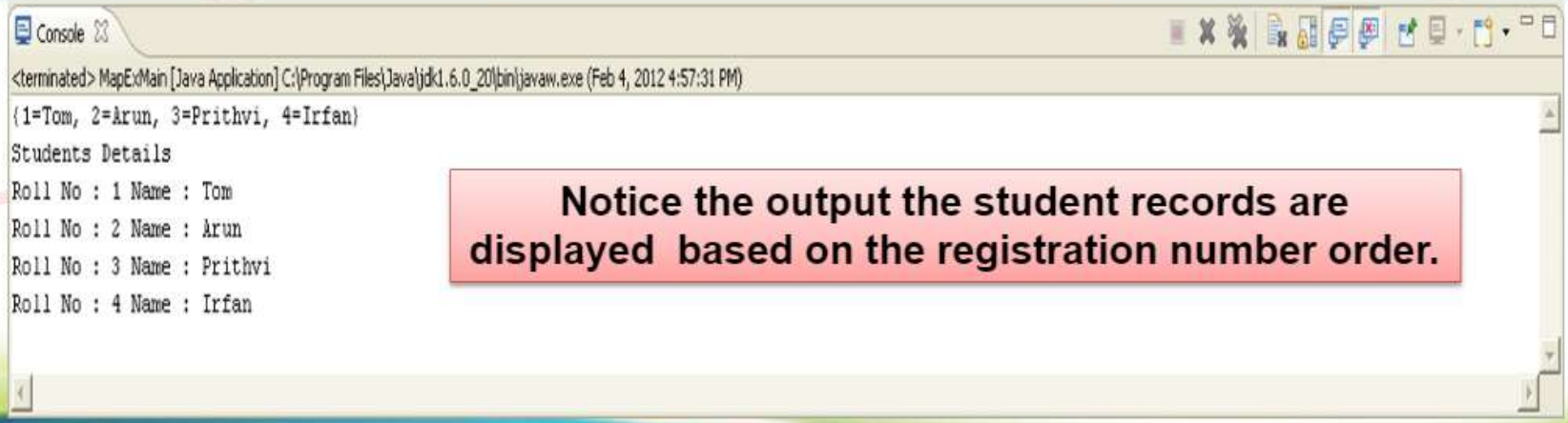
In this hands on, we will see how TreeMap can be used to store elements sorted by the key. We will tweak the previous program to use TreeMap rather than HashMap implementation.

Solution: Change the HashMap to TreeMap in the getStudentDetails() method in MapEx class.

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

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

Note : See how easily we changed the implementation from ***HashMap*** to ***TreeMap*** by making just one change. This is the advantage of declaring collections with the interface ***Map***.



Notice the output the student records are displayed based on the registration number order.