Maps

Interface **Map**

A Map is a collection of *key/value* pairs.

The *Interface* contains the method definitions, such as:

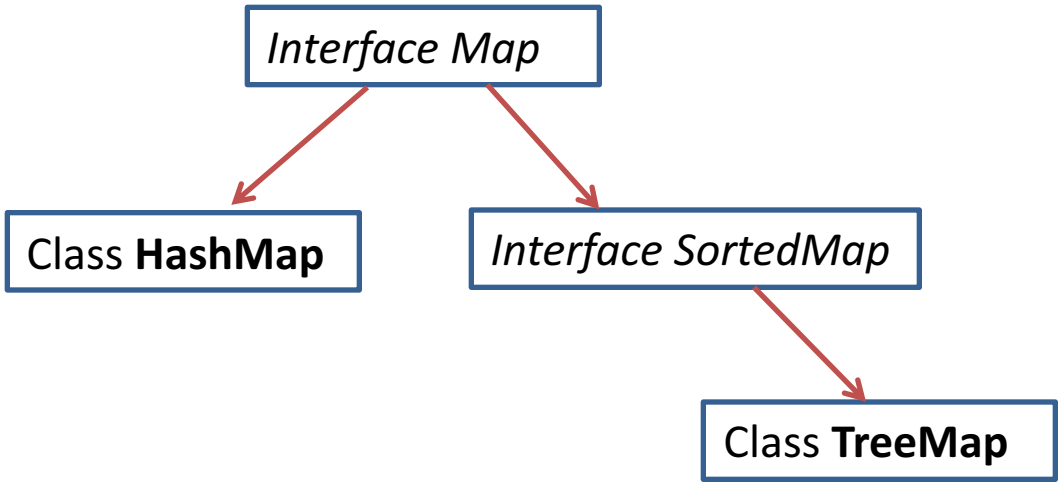
put (“143995066”, new Student(…));

get (“089741885”);

remove (“113478915”);

containsKey (“999999999”);

Key	Value
127468897	*Student Object*
089741885	*Student Object*
894557812	*Student Object*
113478915	*Student Object*


Interface Map
Class **HashMap**
Interface SortedMap
Class **TreeMap**

Class **HashMap**

```
HashMap french = new HashMap();
french.put("cat","chat");
french.put("water","eau");
french.put("moon","lune");
```

A "hash function" maps **key** to **index**

h("cat") →2
h("water")→ 0
h("moon")→ 4

Issues: Search in time O(c)
Collisions
Growth

index	Key	Value
0	water	eau
1		
2	cat	chat
3		
4	moon	lune
…		

Working with the HashMap

```
import java.util.HashMap;
import java.util.Iterator;

public class HashMapExample{
       public static void main(String args[]){
          HashMap hashMap = new HashMap();
          hashMap.put("One", new Integer(1));
          hashMap.put("Two", new Integer(2));
          hashMap.put("Three", new Integer(3));

        Integer myInt = hashMap.get("Two");

        if(hashMap.containsValue(new Integer(1)))
            System.out.println("HashMap contains 1 as value");
         else{
            System.out.println("HashMap does not contain 1 as value");

        if( hashMap.containsKey("One") )
            System.out.println("HashMap contains One as key");
        else
            System.out.println("HashMap does not contain One as value");


}
```

Getting the keys and values out of the HashMap

```
    Iterator itr;
    System.out.println("Retrieving all keys from the HashMap");

    itr = hashMap.keySet().iterator();
    while(itr. hasNext()){
       System.out.println(itr.next());
     }


     System.out.println("Retrieving all values from the HashMap");

     itr = hashMap.entrySet().iterator();
     while(itr. hasNext()){
        System.out.println(itr.next());
     }
}
```

The order items went in is not the same as how they come out!

Class **TreeMap**

```
TreeMap <String, String> french  =
                  new TreeMap<String, String> ( );

french.put("cat","chat");
french.put("water","eau");
french.put("moon","lune");

String frenchWord = french.get("water");
```

frenchWord gets "eau"

A **TreeMap** arranges the data keys so they come out in order when using the iterator.
All the words come out in order of the keys

```
Set set = french.entrySet();
Iterator i = set.iterator();
Map.Entry <String,String> me;

while(i.hasNext()) {
    me = (Map.Entry)i.next();
    System.out.print(me.getKey() + ": ");
    System.out.println(me.getValue());
}
```

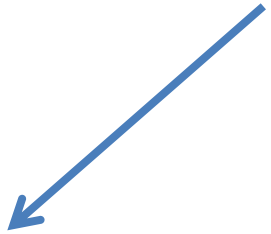
entrySet() returns a collection of key/value pairs

interface Map.Entry is a key/value pair.

Items will come out in the order: cat, moon, water

How is the order determined??

Order can be assume if the class implements *Comparable*

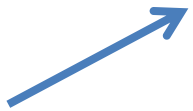


```
TreeMap <String, String> french  =
                  new TreeMap<String, String> ( );
```

For user-defined objects the TreeMap needs to know how to order the keys

<key, value>

```
TreeMap <SSN, Integer> treeMap  =
                    new TreeMap (new SSNComparator());
```



Comparator
a class that implements Comparator
has a method int compare(Object, Object)

The *Comparator* tells the *TreeMap* how the keys are ordered.

```
import java.util.Comparator;

public class SSNComparator implements Comparator <SSN>  {
   public int compare(SSN num1, SSN num2) {
      return num1.compareTo(num2);
   }
```

Put items in the TreeMap

```
TreeMap <SSN, Integer> treeMap  =
          new TreeMap (new SSNComparator() );
SSN numbers[] = {   new SSN("123456789"),
                    new SSN("945621345"),
                    new SSN("765499999")};

treeMap.put(numbers[0], new Integer(1));
treeMap.put(numbers[1], new Integer(2));
treeMap.put(numbers[2], new Integer(3));
```

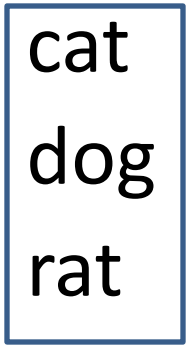
Get items out of the TreeMap

```
TreeMap <SSN, Integer> treeMap  =
          new TreeMap (new SSNComparator());
SSN numbers[] = {   new SSN("123456789"),
                    new SSN("945621345"),
                    new SSN("765499999")};
Integer myData;
myData = treeMap.get(numbers[0]);
```

Why is the TreeMap so efficient?

Keeping items in order:

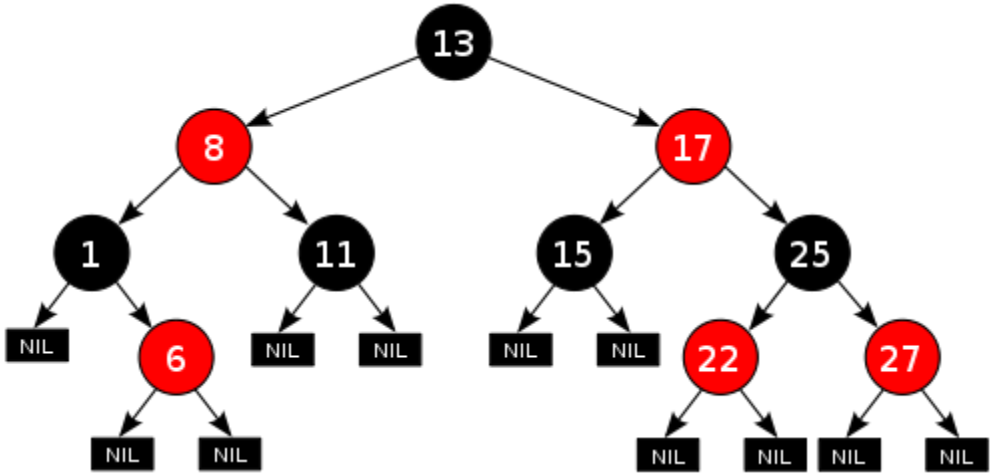
Array



Linked List

add "fat" *O*(n)

The TreeMap is based on the Red-Black Tree




```
public Data search (key, root) {
    if (root == null) return null;
    if (root.key == key) return root.data;
    if (key < root.key) return search (key, root.left);
    if (key > root.key) return search (key, root.right);
}
```