

# Maps and Sets

- Are Interfaces
- Map<K, V> interface - basically a dictionary
- K and V can be of any type
- Can be removed and added (if the same key is added twice the second one replaces the existing pair)

```
public V put(K key, V val);           // store a given key,value pair           java
public V get(K key);                  // get the value associated with given key
public V remove(K key);               // remove key,value pair for given key
public boolean containsKey(K key);    // determine whether key exists in Map
public Set<K> keySet();               // return the set of keys
```

HashMap and TreeMap implement the map interface, Hash is preferred

Map Syntax

```
Map<String, String> noCallMap = new HashMap<String, String>(); // <param, param> should match java
                           with constructor

// add pairs to map
mapName.put("key", "value");
noCallMap.put("Roger M", "090-997-2918");
// get pairs from map
mapName.get("key");
noCallMap.get("Jane Q"));
// remove pairs from map
noCallMap.remove("Jane Q");
// checks if map contains key
System.out.print(noCallMap.containsKey("Jane Q"));
// loop over map
for (String name : noCallMap.keySet())
{
    System.out.println("name: " + name
        + ", phone: " + noCallMap.get(name));
}
```

The Set interface

Treeset, Hashset

- sets are the same no matter the order

```
Set<PhoneRecord> noCall = new TreeSet<PhoneRecord>();           java
boolean add(E element);    // add an element to the set
noCall.add(new PhoneRecord("Stacy K", "090-997-9188"));
boolean contains(Object o); // does the set contain given object?
boolean inside = noCall.contains(roger);
boolean remove(Object o);  // remove given object from the set
```

