

# Sets and Maps

---



**Nelson Padua-Perez**

**Bill Pugh**

**Department of Computer Science**

**University of Maryland, College Park**

# Sets

- **A Set is simply a set**
  - **a collection of elements without duplicates**
- **No front or back**
- **The order in which elements are added to a set doesn't matter**
- **But a set (generally) offers the ability to find or remove an element quickly**
  - **without having to search through all the elements**

# Set operations

- **contains** - does this set contain a value?
- **remove** - remove a value from the set (return true if it was found)
- **add** - add value to the set

# How do sets work?

- **Finding a matching element is based on the equals method**
  - **If you want a collection of your own classes, you will need to define your own equals(Object) method**
- **HashSet is most common Set implementation**
  - **needs a hashCode() function**

# hashCode()

- **Function must return some int such that**
  - if `x.equals(y)`,
  - then we are guaranteed that
  - `x.hashCode() == y.hashCode()`
- **Hashcode already defined for many classes**
  - `String`, `Integer`, etc.
- **For a Point class:**
  - ```
class Point {  
    final int x,y;  
    public int hashCode() { return x + 17*y; }  
}
```

# Good hashCode() functions

- A good hashCode function is one that tries to make it so that two different objects are likely to have different hashCodes
  - can't make promises; for any hashCode function, there are unequal objects that have equal hashCodes
- But any valid hashCode function is better than an invalid one
  - `public int hashCode() { return 42; }`

# We'll come back to hashCode

- We'll come back to hashCode and hash tables later
- Designing a good one, and implementing a HashSet is a little bit tricky
- But an important and valuable thing to understand and something that will be useful to you

# SortedSet

- **A Set of ordered objects**
  - For example, Integers or Strings
  - Allows you to ask for things such as the smallest value, or largest value
    - can also ask for a headSet or tailSet
      - a headSet is the set of all values less than a supplied value
  - Can use objects that implement the Comparable interface
  - Or provide a Comparator object as a argument to the constructor

# Map

- **A dictionary**
  - A set of keys
  - for each key, an associated value
- You can think of it as an array that can be indexed by any value
- **Methods on Map**
  - `get(key)`
  - `put(key, value)`
  - `containsKey(key)`
  - `remove(key)`
  - `keySet()`

# Example Map usage

```
public static void main(String args[]) {  
    SortedMap<String, Integer> map  
        = new TreeMap<String, Integer>();  
    for(String s : args) {  
        if (map.containsKey(s))  
            map.put(s, map.get(s)+1);  
        else map.put(s,1);  
    }  
    for(String s : map.keySet())  
        System.out.println(map.get(s) + "\t" + s);  
}
```

# Map notes

- If you call `get` and the key isn't present, `null` is returned
- Map is not a collection
  - but you can get the `keySet()`
  - for more advanced usage, might also want `entrySet()` or `values()`
- `HashMap` is most common implementation
  - also a `TreeMap` (which is a `SortedMap`)