Sets

- Set is an unordered list of items
  - Items are unique! Only one copy of each item in set!
- We will use two different implementations of sets
- TreeSet
  - A TreeSet backed up by a tree structure (future topic)
  - Keeps items sorted (+)
  - Slower than HashSets (-)
- HashSet
  - A TreeSet backed up by a hashing scheme (future topic)
  - Items not sorted - should seem to be in random order (-)
  - Faster than TreeSets (-)

Basic Operations:

- Create a set
- Add an item to a set
- Check if item is in a set
- Is set empty?
- Remove item from set

Look at API!

- What other goodies are there?

Example – Create and add to Set

```java
TreeSet<String> firstnames = new TreeSet<String>();
firstnames.add("John");
firstnames.add("Emily");
firstnames.add("Alex");
firstnames.add("Mike");
firstnames.add("Mike");
```

```
John Emily Mike
```

Example – Is object in set?

```java
if (firstnames.contains("Zed"))
    System.out.println("Zed is in the set.");
else
    System.out.println("Zed is not in the set.");
if (firstnames.contains("Mike"))
    System.out.println("Mike is in the set.");
else
    System.out.println("Mike is not in the set.");
```
Iterator – Look at each element in a Set

- Can get an iterator to look at each element in the set (as covered recently)
- May not know the order of the elements
- Guaranteed to give you all the elements in the set – one at a time

```java
// You must first get an iterator for set
Iterator<String> nameIter = firstnames.iterator();

for (String name : firstnames) {
    System.out.println(name);
}
```

Iterate over elements in Set firstnames

- With for-each loop, iterator is automatically invoked for you!

```java
for (String name : firstnames) {
    System.out.println(name);
}
```

Other Basic Operations on Sets

- `size()` – returns size of set
  ```java
  System.out.println("Size of set is "+firstnames.size());
  ```
- `remove(object)` – remove object from set if there
- `isEmpty()` – return true if set is empty

- See “Sets” and “Iterator” on Java API page
  - Note that Sets i an interface, not a class

Set Operations

- Union of two sets
  - all the elements from both sets
- Intersection of two sets
  - the elements that are in both sets
- Difference of two sets (A – B)
  - the elements in A that are not in B
How do we?

- Implement set operations for two sets
  - Union, intersection, difference
- Implement set operations for array of sets
  - Union, intersection

Sets

- You should have discovered by looking up the Set API:
  - Almost everything has been done for you!
- Union of two sets – Use:
  - boolean `addAll(Collection c)`
- Intersection of two sets – Use:
  - boolean `retainAll(Collection c)`
- Difference of two sets (A – B) – Use:
  - boolean `removeAll(Collection c)`

Collections: ArrayList vs Set

- ArrayList
  - directly access an item
  - keep items ordered (as entered)
  - Can have duplicates of items
  - What are operations on an ArrayList?
- Sets
  - Keeps items unordered
    - (Unless using TreeSet: sorted)
  - No duplicate items
  - Easily remove duplicates
  - What are operations on a Set?

Using Both ArrayList and Sets

- You may want to use a set to get rid of duplicates, then put the items in an ArrayList and sort them!
- Problem:
  - Often data comes in the form of an array
  - How do we go from array to ArrayList or TreeSet?
- Problem:
  - Often we are required to return an array
  - How do we go from a Collection such as an ArrayList or TreeSet to an array?
- Can do it the “hard” way with loops or iterators:
  - one item at a time
- OR:
Converting from array to Collection

- For arrays of objects (such as Strings) use the `asList` method in the `Arrays` class.
  - This returns a fixed-size list backed by the specified array
  - Pass this into the constructor of your ArrayList or set

- Example
  ```java
  String[] words = String[N];
  ...
  TreeSet<String> wordset = new TreeSet<String>(Arrays.asList(words));
  ```

Converting from Collection to array

- Collections such as ArrayLists and TreeSet have a `toArray` method
  - This returns an array
  - Syntax a bit awkward

- Example
  ```java
  TreeSet<String> wordset = new TreeSet<String>();
  ...
  String[] words = (String[]) wordset.toArray(new String[0]);
  or
  return (String[]) wordset.toArray(new String[0]);
  ```