JAVA SETS AND MAPS
EXPERIMENTAL ANALYSIS

• We should run an experiment to compare our various implementations

• I have collected data for adding \( n \) doubles (as keys) to maps and sorted maps to help analyze
  • This is not comprehensive, just one view of our options
  • Added random data that did not "overlap", again a limited view

• What conclusions do you draw?
EXPERIMENTAL ANALYSIS
MAP ADT IMPLEMENTATIONS
EXPERIMENTAL ANALYSIS
SORTED MAP ADT IMPLEMENTATIONS

![Graph showing time taken for different sizes of SortedTableMap, TreeMap, and JavaTreeMap.](image)
SUMMARY OF CLASSES (SETS)

- **EnumSet**<E> - Specialized set for enumerated types. Implemented as bit-vector.

- **HashSet**<E> - Set implemented with chained hash table – no guaranteed iteration order
  - **LinkedHashSet**<E> - Same, but with guaranteed iteration order. No complexity overhead
  - *E should override both hashCode() and equals()!* Default hashCode() hashes memory address of object (typically, not always) and default equals() compares memory address

- **TreeSet**<E> - Set implemented with red-black tree. Needs Comparator<E> or E should implement Comparable<E>

- There are no multisets in the Java library. Some external libraries have them.
- To find how to use them, go to the Java API!
EXAMPLE OF USING SET<E>

1. `Scanner s = new Scanner(new File("numbers.txt"));`
2. `HashSet<Integer> numbers = new HashSet<>();`
3. `while (s.hasNextInt())`  
4. `numbers.add(s.nextInt());`
5. `...elsewhere...`
6. `int sumOfUnique = 0;`
7. `for (Integer i : numbers)`  
8. `sum += i;`
EXAMPLE OF OVERRIDEING HASCODE()

1. public class Student {
2.     String id;  //unique!
3.     String name;
4.     List<Courses> courses;
5.     ...stuff...
6.     public boolean equals(Object obj) {
7.         if(obj != null && obj instanceof Student)
8.             return id.equals(((Student)obj).id);
9.         return false;
10.     }
11.     public int hashCode() {
12.         return id.hashCode();
13.     }
14. }

Really great stackoverflow answer about creating hash codes.
SUMMARY OF CLASSES (MAPS)

- **EnumMap<K, V>** - Specialized set for enumerated types. Implemented as bit-vector.
- **HashMap<K, V>** - Set implemented with chained hash table – no guaranteed iteration order
  - **LinkedHashMap<K, V>** - Same, but with guaranteed iteration order. No complexity overhead
  - **K should override both hashCode() and equals()! Default hashCode() hashes memory address of object (typically, not always) and default equals() compares memory address**
- **TreeMap<K, V>** - Set implemented with red-black tree. Needs Comparator<K> or K should implement Comparable<K>
- **IdentityHashMap<K, V>** - hash map specifically for objects that use reference equality instead of object equality
- **WeakHashMap<K, V>** - Like a hash map, but does not prevent garbage collector from removing keys
- There are no multisets in the Java library. Can fake with `Map<K, List<V>>`
- To find how to use them, go to the Java API!
EXAMPLE OF USING MAP<K,V>

1. `Scanner s = new Scanner(new File("numbers.txt"));`
2. `HashMap<Integer,String> numbers = new HashMap<>();`
3. `while (s.hasNextInt())`
4. `numbers.put(s.nextInt(), s.next());`
5. `...elsewhere...`
6. `System.out.println(numbers.get(5)); //get a value`
PROBLEM

• Design a barcode scanner for a super market
  • You are given an inventory (item barcode, per unit cost, and quantity per item) of the store (file online)
  • Let a customer come with a shopping list (you design file format)
    • Look up in the inventory database for each item's availability
    • If possible to purchase subtract the quantity from the inventory
    • If the item is purchased, add the price to the total
    • Print a receipt for the purchase

• What will be the underlying container for your database?