Sets

CS 121: Data Structures

Symbol Tables and Sets

- Symbol Tables: store key-value pairs. No duplicate keys.
- Sets: store keys. No duplicates.

```
> cat example.txt
E X A M P L E
```

```
> java-algs4 Count < example.txt
A: 1
E: 2
L: 1
M: 1
P: 1
X: 1
```

```
> java-algs4 Unique < example.txt
A
E
L
M
P
X
```

Count.java

```
ST<String, Integer> st = new ST<String, Integer>();
while(!StdIn.isEmpty()) {
    String s = StdIn.readString();
    if (st.contains(s)) st.put(s, st.get(s) + 1);
    else st.put(s, 1);
}
for (String s: st.keys()) {
    StdOut.println(s + ": " + st.get(s));
}
```

```
> java-algs4 Count < example.txt
A: 1
E: 2
L: 1
M: 1
P: 1
X: 1
```

Unique.java, using Symbol Tables

```
ST<String, Integer> st = new ST<String, Integer>();
while(!StdIn.isEmpty()) {
    st.put(StdIn.readString(), 1);
}
for (String s: st.keys()) {
    StdOut.println(s);
}
```

A bit wasteful, since we aren't using the values.

```
> java-algs4 Unique < example.txt
A
E
L
M
P
X
```

Unique.java, using Sets

```
SET<String> set = new SET<String>();
while(!StdIn.isEmpty()) {
    set.add(StdIn.readString());
}
for (String s: set) {
    StdOut.println(s);
}
```

Code is clearer, and doesn't waste space for unused values.

```
> java-algs4 Unique < example.txt
A
E
L
M
P
X
```

SET API

public class SET<Key extends Comparable<Key>>

SET()	*create an empty set*
void add(Key key)	*add key to the set, if not already present*
boolean contains(Key key)	*true if the set contains the key*
void delete(Key key)	*remove key from the set, if it was present*
boolean isEmpty()	*is the set empty?*
int size()	*number keys in the set*
Iterator<Key> iterator()	*iterator over all keys in the set*

Set Implementations

- Sets are simplified symbol tables (i.e., just keys, no values), so sets can be implemented using any of the approaches we've seen: separate chaining or linear probing hash tables, binary search trees, etc.
- The textbook's SET class requires that keys be comparable, to support ordered operations
  - Internally, SET uses java.util.TreeSet to store the keys
- Other set implementations, such as java.util.HashSet, don't have this requirement