

【DS】 Day15

☰ Tags	
📅 Date	@June 9, 2022
☰ Summary	Symbol Table API

【Week 4】 Symbol Table

4.4 Symbol Table API

Key-value pair abstraction:

- Insert a value with specified key
- Given a key, search for the corresponding value.

API

```
public class ST<Key, Value> {
    ST() // Create a symbol table.
    void put(Key key, Value value) // Put key-value pair into the table(remove key from table if value is null)
    Value get(Key key) // Get value paired with key(null if key doesn't exist)
    void delete(Key key) // Remove key(and its value) from table
    boolean contains(Key key)
    boolean isEmpty()
    int size()
    Iterable<Key> keys()
}
```

Conventions:

- Values are not null
- Method `get()` returns null if key not present
- Method `put()` overwrites old value with new value.

Intended Consequences:

- Easy to implement `contains()`

```
public boolean contains(Key key) {
    return get(key) != null;
}
```

- Can implement lazy version of `delete()`

```
public void delete(Key key) {
    put(key, null);
}
```

Key type:

- Assume keys are `Comparable`
- Assume keys are any generic type, use `equals()` to test equality.

Test Client:

```
public static void main(String[] args) {
    ST<String, Integer> st = new ST<String, Integer>();
    for (int i = 0; !StdIn.isEmpty(); ++i) {
        String key = StdIn.readString();
        st.put(key, i);
    }
    for (String s : st.keys())
        StdOut.println(s + " " + st.get(s));
}
```

```
public class FrequencyCounter
{
    public static void main(String[] args)
    {
        int minlen = Integer.parseInt(args[0]);
        ST<String, Integer> st = new ST<String, Integer>();
        while (!StdIn.isEmpty())
        {
            String word = StdIn.readString();
            if (word.length() < minlen) continue;
            if (!st.contains(word)) st.put(word, 1);
            else
                st.put(word, st.get(word) + 1);
        }
        String max = "";
        st.put(max, 0);
        for (String word : st.keys())
            if (st.get(word) > st.get(max))
                max = word;
        StdOut.println(max + " " + st.get(max));
    }
}
```

Annotations:

- ← create ST (points to `ST<String, Integer> st = new ST<String, Integer>();`)
- ← ignore short strings (points to `if (word.length() < minlen) continue;`)
- ← read string and update frequency (points to `st.put(word, 1);` and `st.put(word, st.get(word) + 1);`)
- ← print a string with max freq (points to `StdOut.println(max + " " + st.get(max));`)