CMPT 225: Data Structures & Programming – Unit 22 – Dictionaries

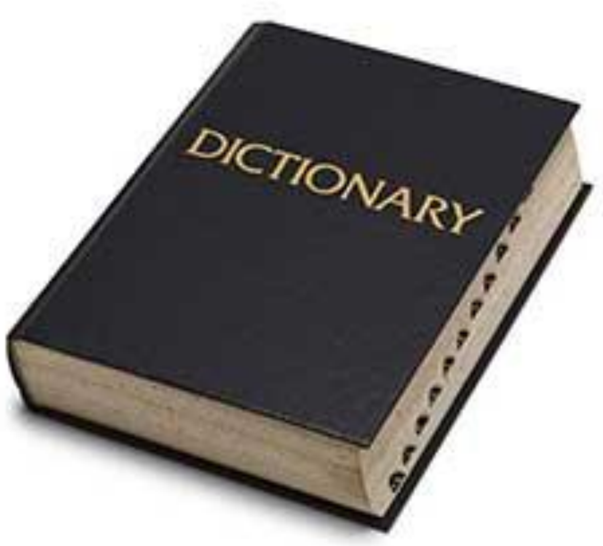
Dr. Jack Thomas
Simon Fraser University

Spring 2021

Today’s Topics

- What is a Dictionary?
- When to use a Dictionary
- Dictionary ADT
- Dictionaries in Java
- Implementing a Dictionary
- Dictionaries: What’s Up With Them Anyway?

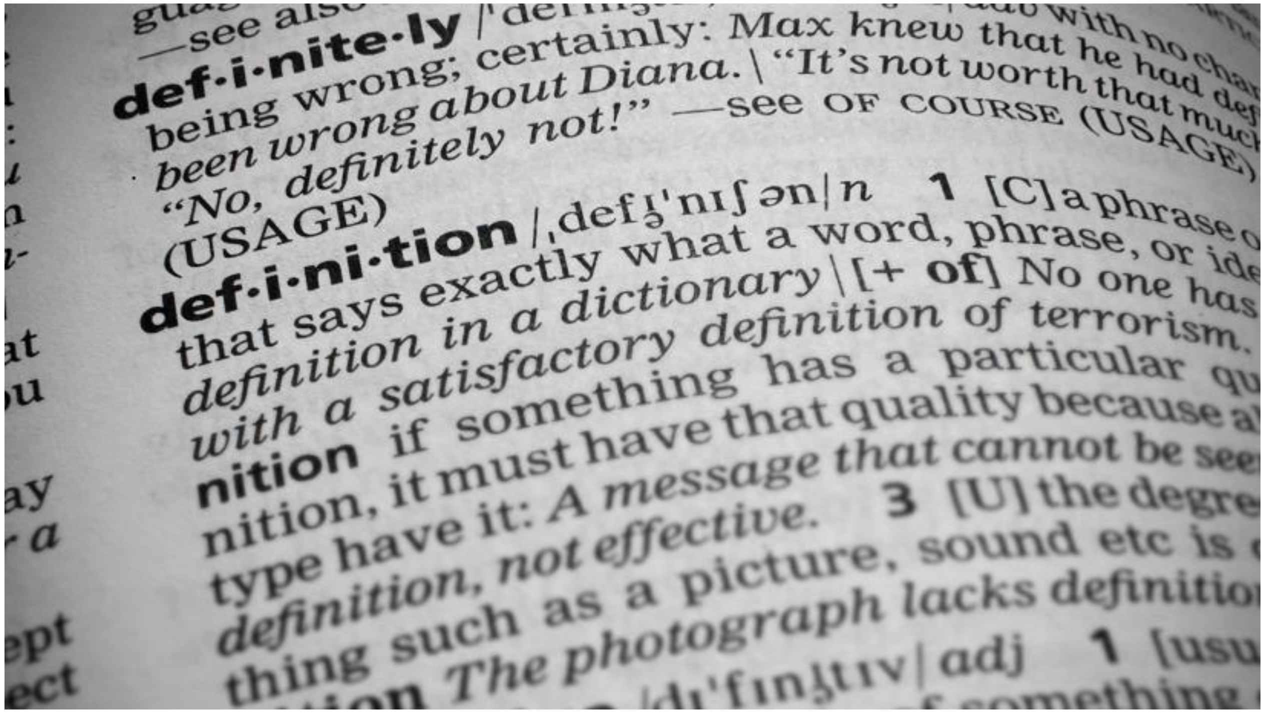
Remember These?



- If you don’t know what it is, Google it.
- What if you could have the power of a **Dictionary**… but **as a data structure**?

Image credit: https://www.collinsdictionary.com/dictionary/english/dictionary

Dictionaries Contain Definitions



—see also
def·i·nite·ly /ˈde
being wrong; certainly: *Max knew that he had no chan*
been wrong about Diana. | *"It's not worth that much*
"No, definitely not!" —see OF COURSE (USAGE)
(USAGE)
def·i·ni·tion /ˌdefəˈnɪʃən/ *n* **1** [C] a phrase o
that says exactly what a word, phrase, or ide
definition in a dictionary | [+ **of**] *No one has*
with a satisfactory definition of terrorism.
nition if something has a particular qu
nition, it must have that quality because al
type have it: *A message that cannot be see*
definition, not effective. **3** [U] the degre
thing such as a picture, sound etc is
The photograph lacks definitio
/dɪˈfɪnətɪv/ *adj* **1** [usu

- A dictionary has a **definition** for each word – and sometimes **more than one**!
- Can you even imagine a **data structure with multiple definitions**?

Image credit: https://news.sky.com/story/listen-up-fam-1-400-new-words-added-to-dictionary-in-controversial-updation-11517902

It’s A Map Where the Keys Don’t Have To Be Unique

- I mean **that’s basically it**.
- Made up of entries with key-value pairs, like a map.
- It’s kind of like how one word in a regular dictionary can have multiple definitions.
- It’s got some new applications at least.

Like What?

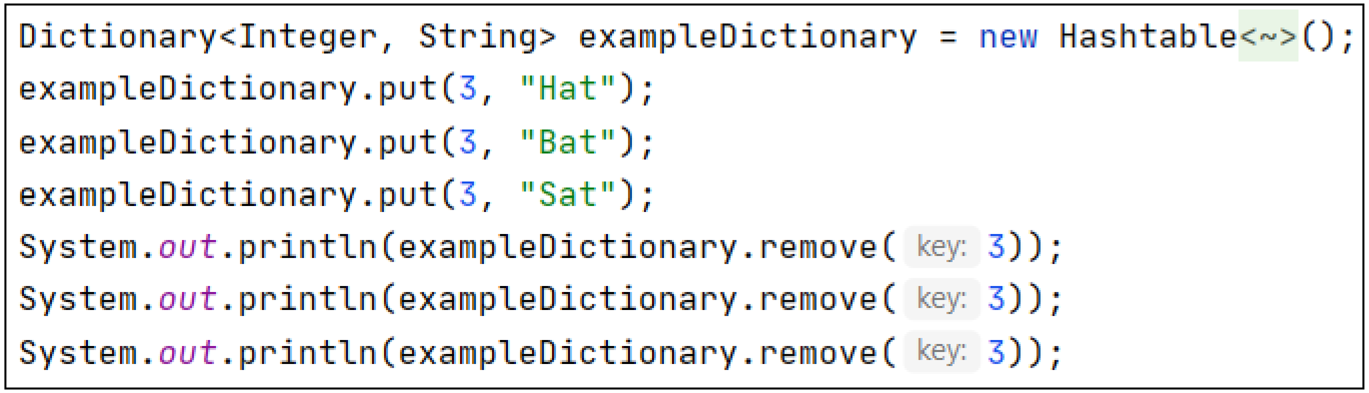
- Like… a **Dictionary**?
- I guess some stuff with **DNS protocols** and **credit card transactions**…
- To be real, though, **you don’t often need a Map that lets you store non-unique keys**.

The Dictionary: The ADT

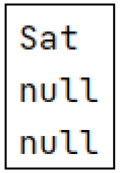
- A **non-unique-key-based data structure** for storing entries made of key-value pairs.
- Includes the following standard methods:
  - **Get**: Returns an entry with a given key.
  - **getAll**: Returns a collection of all entries with agiven key.
  - **Put**: Creates and adds a new entry with a given key and value into the dictionary.
  - **Remove**: Removes a given entry from the dictionary, and returns it as proof.
  - **entrySet**: Returns a collection of all entries.
  - **isEmpty**: Return whether the dictionary is empty.
  - **Size**: Return the number of entries.

Dictionaries in Java

- **Java** has a **Dictionary abstract class**, which you'll recall isn't an interface but can't be instantiated either.
- But you can **instantiate classes that extend it as a Dictionary**, and then use them as though they were one.



```java
Dictionary<Integer, String> exampleDictionary = new Hashtable<~>();
exampleDictionary.put(3, "Hat");
exampleDictionary.put(3, "Bat");
exampleDictionary.put(3, "Sat");
System.out.println(exampleDictionary.remove( key: 3));
System.out.println(exampleDictionary.remove( key: 3));
System.out.println(exampleDictionary.remove( key: 3));
```



```
Sat
null
null
```

Wait That’s Not Right

- Yep – **Dictionaries in Java aren’t like the ADT Dictionary**, they’re a part of Java’s data structure architecture that’s like a simple Map.
- Plus, **they’re obsolete**. Hashtable is extended from it, true, but most similar data structures just inherit the Map interface these days.
- Sooooo if you want a Dictionary **you’re going to have to make one**.

Implementing a Dictionary

- A lot **like implementing a regular Map**, where you choose a good underlying data structure and use it to fulfill each method of the ADT.
- Some options:
  - **Unordered List**: Search the list from head to tail for the first matching key.
  - **Ordered Search Table**: Nothing in its implementation requires the keys to be unique, they'll just be stored adjacent to each other.
  - **Hash Table using Separate Chaining**: All entries with the same key will just be stored in the same bucket.
  - **Skip List**: Same as above.

Unsorted List Dictionary

- Using an unsorted list allows insertions in O(1), but finds matching keys in O(n).

```
Algorithm find(k):
        for each p in (S.begin(), S.end()) do
                if p.key() = k then
                        return p
```

```
Algorithm put(k,v):
        Create a new entry e = (k,v)
        p = S.insertBack(e)
        return p
```

```
Algorithm erase(k):
        for each p in [S.begin(), S.end()] do
                if p.key() = k then
                        S.erase(p)
```

What About Sorting?

- The sorted, array-based implementation is called a **Search Table**.
- It improves finding to O(log n) thanks to binary search, but now adding and removing takes O(n) as we may have to move everything else in the array around.
- Effective only for small dictionaries or ones where we mostly search rather than add.

What’s Going On With This Data Structure?

- Dictionaries are a prime example of the difference between **theoretical completeness for Abstract Data Types** and the **practical needs of a working programming language**.
- **For theory reasons**, it’s useful to have a name and a definition for a certain kind of imaginable data structure as a distinct idea.
- **For a programmer**, there’s not much need day to day for a Dictionary class, and what it offers can mostly be covered by existing classes with a little tweaking. Otherwise, it’s just unnecessary bloat.

Recap – A Word That Means Summarizing the Preceding

- A **Dictionary** is a kind of key-based data structure, like a Map, where the keys don't have to be unique.
- It has an **ADT**, but the **Java abstract class** isn't the same sort of Dictionary.
- There's a variety of **straightforward options for implementing** it based on what we've done so far.
- In practice, **there isn't much programming need for it**, but it fills a space in our theory of data structures.