Essential Data Structures-

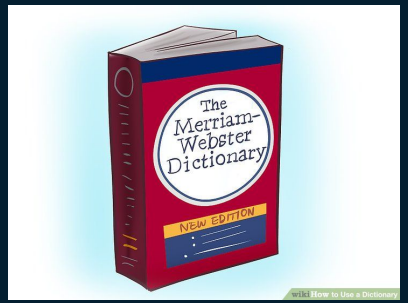
Maps, Sets, and Hashing

Week 4 - AI Inspire 2019

Quick Review

Before we start let’s do a quick review of prev. lecture (iterators, recursion, etc.) since many of the concepts moving forward will build off.

Maps/Symbol Tables/Dictionaries



Alternate Phrases

- Maps
- Symbol Tables
- Dictionaries
- Associative Arrays

- They all mean the SAME thing

Key-Value Pair

- Keys are associated with values
- Key-Value Pair
- 2 linked data types
  - With key → find a value associated with that key
  - Key = unique identifier of data AND value = data that is identified OR pointer to data
- NO DUPLICATE KEYS

key	value
firstName	Bugs
lastName	Bunny
location	Earth

Examples of Key-Value Pairs

Find page numbers from book that have a term
Key = term
Value = list of page numbers

Purpose = find definition from dictionary

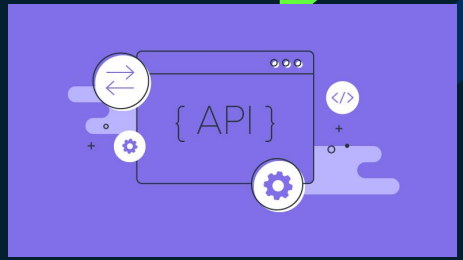
Key = word
Value = definition

Purpose = find web pages

Key = keyword/search query
Value = list of page numbers

API

- put (Key key, Value val)
  - Insert key-value pair
  - Will overwrite val if key already has a value
- get (Key key)
  - Obtain value paired with key
- contains (Key key)
  - Does there exist a value for given key?
- Iterable<Key> keys()
  - List out all keys
- size ()
  - Give size of map

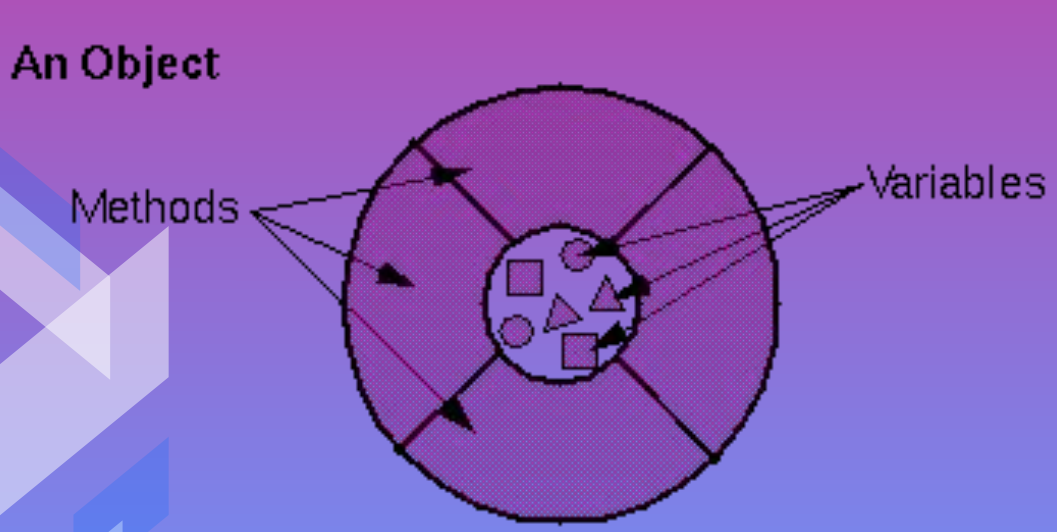


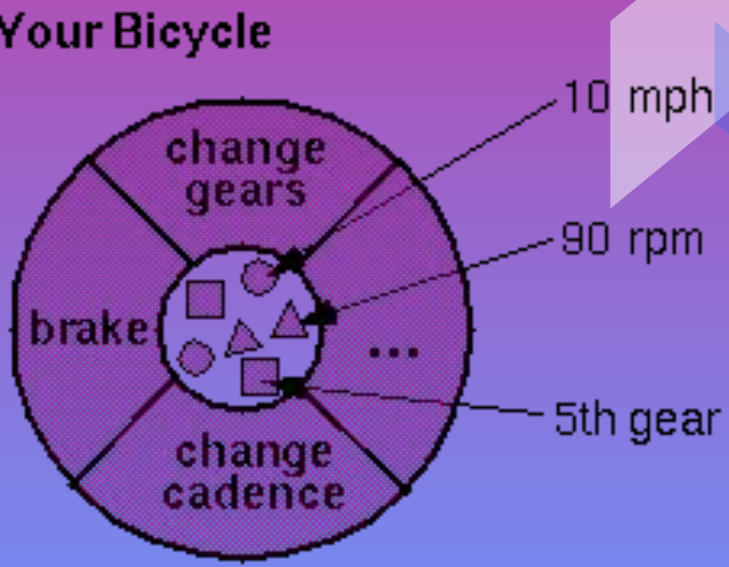
Quick Java Crash Course

Objects & Instantiation

- Call constructor of object which creates instance or object
- Object
  - Bundle/Combination of variables + methods
  - Java is OOP ⟶ revolves around objects
- Instantiation
  - Allocates some memory for object & returns reference a reference

Objects & Instantiation



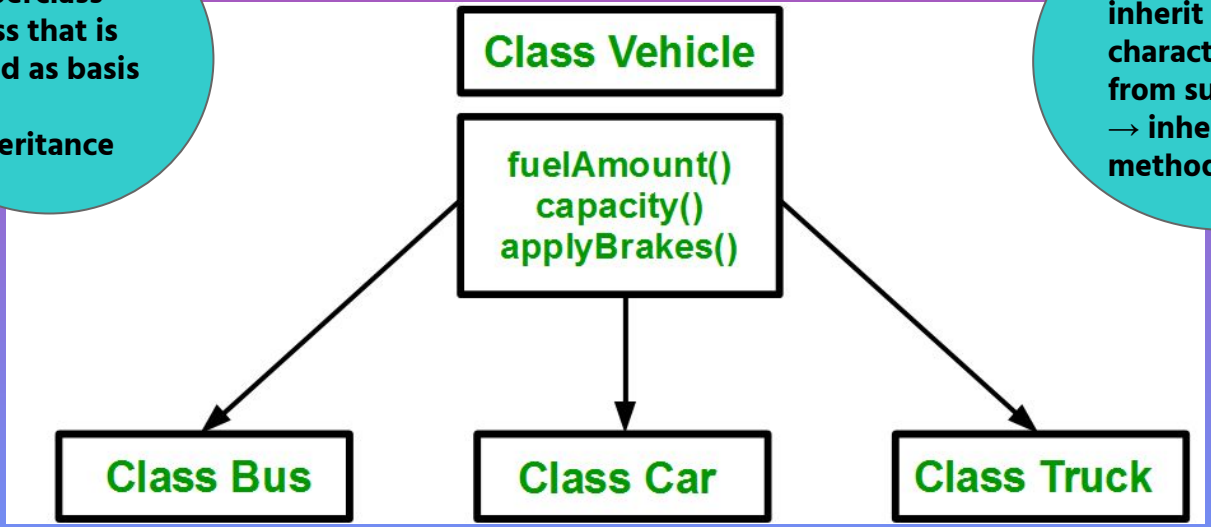


But wait - how do we create an instance of the class?

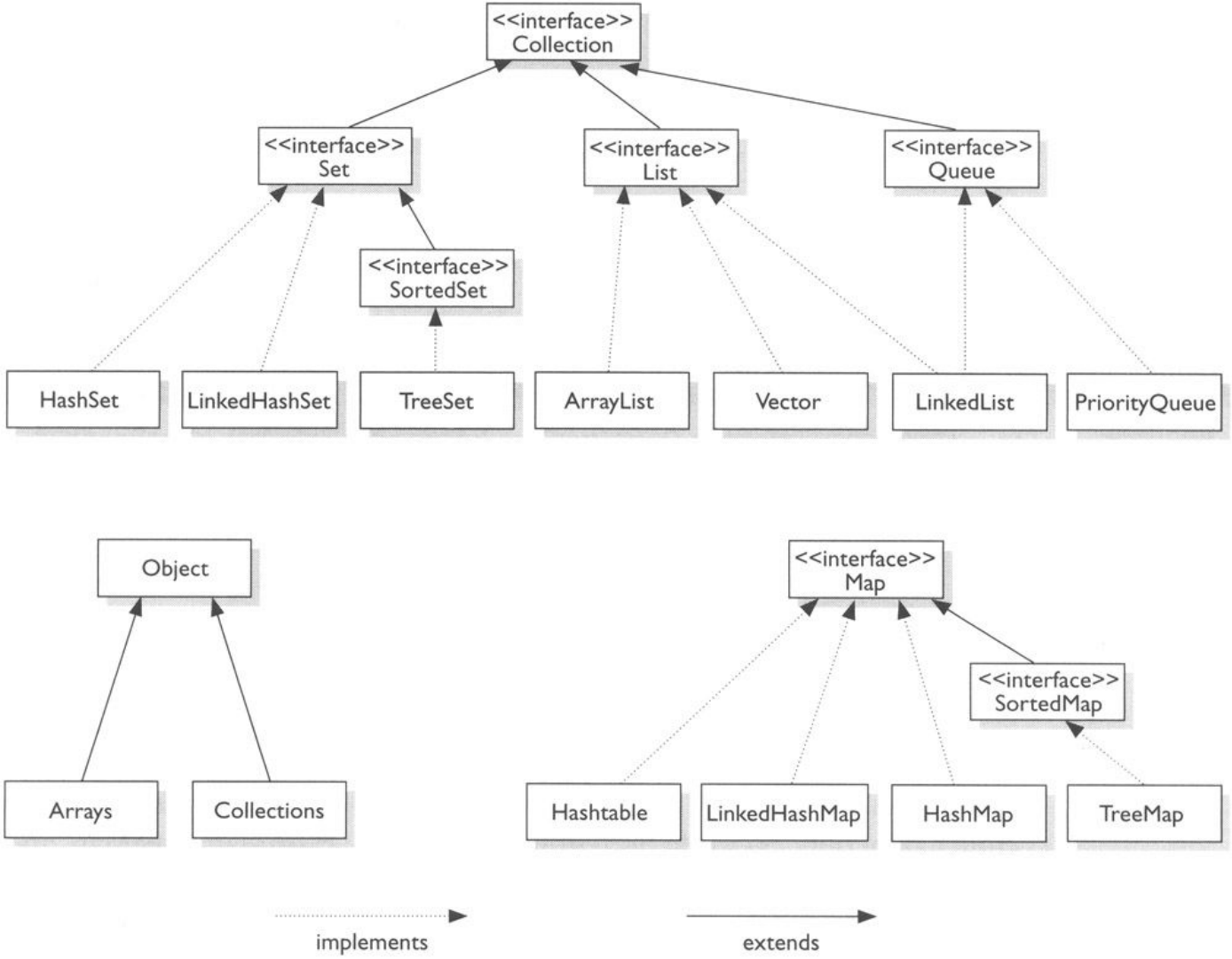
- Maps = Interface
  - Interfaces - Only define methods, NONE of them defines a method
- Classes extend classes
  - Class inherits info from another class
- Classes implement interfaces
  - Clases defines ALL methods from interface
- Only classes can instantiate a method and NOT interfaces

Inheritance Concept

Superclass = class that is used as basis for inheritance



Subclasses inherit characteristics from superclass → inherit methods


<<interface>>
Collection
<<interface>>
Set
<<interface>>
List
<<interface>>
Queue
<<interface>>
SortedSet
HashSet
LinkedHashSet
TreeSet
ArrayList
Vector
LinkedList
PriorityQueue
Object
Arrays
Collections
<<interface>>
Map
<<interface>>
SortedMap
Hashtable
LinkedHashMap
HashMap
TreeMap
implements
extends

How do we create an instance of the class?

Map<String, Integer> MAP = new HashMap <String, Integer>

Map<String, Integer> asdf = new TreeMap<String, Integer>

HashMap<String, Integer> x = new HashMap<String, Integer>

Back to Maps! Time to start develop algorithms!

Develop Algorithm #1

After reading through list of items

Store freq. of words in a map, where key = word and value = frequency

Develop Algorithm #2

Print out string with max frequency and it's max frequency

Store freq. of words in a map, where key = word and value = frequency

Analyzing Implementations of Map

Why are we learning implementations of data structures?

- More aware and smarter when using the data structure
  - “Know what’s under the hood” → internal workings
- Sometimes need to edit/modify the data structure so we need to have the implementation
  - Can change our own code instead of copy-pasting from API and not having control of changing underlying code in package
- Helps you analyze which method is most efficient → improve analysis of runtime of algos

Implementation 1 - Unordered list

- Unordered list of key-value pairs
- Get - scan through each key
- Put - scan through all keys till match
  - Otherwise, just add to list
- **Analyze the runtime efficiency!**

Implementation 2 - Ordered List

- More efficient than unordered list for searching but NOT insertion
- Use binary search
  - Binary search speeds up searching and NOT insertion
- Need to learn binary search first
- Will come back to this implementation

Implementation 3 - Binary Search Tree

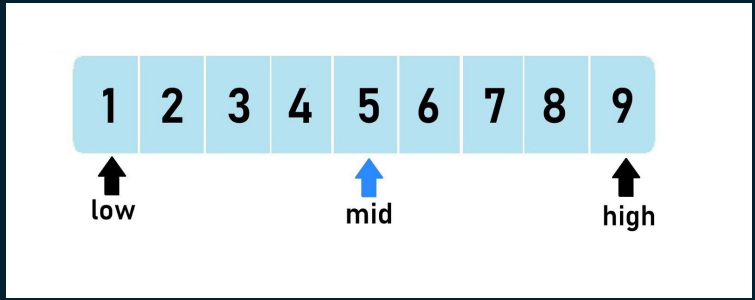
- Most efficient implementation
- Need to learn binary search first
- Will come back to this implementation

Binary Search

Binary Search

https://www.youtube.com/watch?v=P3YID7liBug

Runtime?



Suggested Homework

(Purple = Mostly Covered in class)

1. Readings + code on Maps + hashmaps
   a. Different tasks using these APIs
   b. Need to understand Java OOP for coding
   c. Logical thinking for pseudocode
   d. Will post additional reading + code + resources
2. Implementation + analysis of algos (unordered, ordered, BSTs)
   a. Analyzing how fast algos for implementations work
      i. Analyzed runtime for only unordered
   b. Need to understand how binary search works before jumping into implementation for ordered and BSTs
   c. Will post additional reading + code + resources

Hope you had fun!