Advanced Topics: Collections

Colorado State University
Department of Computer Science

Announcements

- NEXT Week (week before finals)
  - Monday – Review
  - Wednesday – No Lecture, use time to catch up and work on finalizing your projects
  - Thursday – Time to work finish your labs and practical project work
  - Friday – Early Take Option, Final Exam
  - HELP DESK CLOSES - Friday May 3, plan accordantly!
- Finals Week
  - Tuesday May 7 7:30am-9:30am at CS110 – 002 section

Help Desk

Day	Time : Room
Monday	2 PM - 5 PM : CSB 120
Tuesday	6 PM - 8 PM : Teams
Wednesday	3 PM - 5 PM : CSB 120
Thursday	6 PM - 8 PM : Teams
Friday	3 PM - 5 PM : CSB 120
Saturday	12 PM - 4 PM : Teams
Sunday	12 PM - 4 PM : Teams

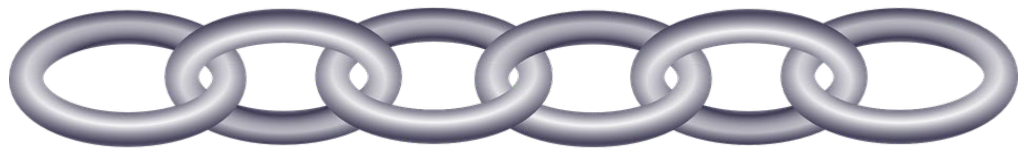
ArrayLists

- ArrayLists
  - Part of Java Collections Library
  - Assumes default naming conventions
    - done through interfaces and abstract classes!
  - .add(Type)
  - .remove(location)
  - .size()
- Is ArrayList always best to use?
  - What happens if it is *very* large?
  - Hard to find continuous memory in order!
  - Causes actions to slow down
- Introducing Data Structures (CS 165)

LinkedList

Additional Reading: https://www.geeksforgeeks.org/linked-list-in-java/
Simulator: Visualgo: https://visualgo.net/en/list
Blockchain: https://www.ibm.com/topics/what-is-blockchain

- Think about a chain
- Each link connects to the **next**
- Linked Lists
  - Connect objects to the next
  - But don't worry about it all being order in memory
- If you know the next, they can be anywhere
- pros
  - memory efficient
- cons
  - what if a link is broken?
  - Can you easily jump to the middle? – no!
- Also a foundation of blockchain!



```java
List<String> list = new LinkedList<>();
list.add("A");
list.add("B");
for(String str : list)
System.out.println(str);
```

LinkedList basic methods and Practice

get()	`get(index)` Returns element at specified index.
set()	`set(index, newElement)` Replaces element at specified index with newElement. Returns element previously at specified index.

add()	`add(newElement)`
	
	Adds newElement to the end of the List. List's size is increased by one.
	
	`add(index, newElement)`
	
	Adds newElement to the List at the specified index. Indices of the elements previously at that specified index and higher are increased by one. List's size is increased by one.

size()	`size()` Returns the number of elements in the List.

remove()	`remove(index)`
	
	Removes element at specified index. Indices for elements from higher positions are decreased by one. List size is decreased by one. Returns reference to element removed from List.
	
	`remove(existingElement)`
	
	Removes the first occurrence of an element which is equal to existingElement. Indices for elements from higher positions are decreased by one. List size is decreased by one. Returns true if specified element was found and removed.

```
import java.util.LinkedList;
Import java.util.List;
import java.util.ListIterator;
public class LLTest {
  public static void main(String args[]){
    List<Integer> list = new LinkedList<>();
    list.add(2);
    list.add(4);
    list.add(6);
    ListIterator<Integer> intIterator = list.listIterator();
    //what is printed?
    while(intIterator.hasNext()){
      System.out.print(intIterator.next());
    }
    System.out.println("");
    //how to add number 5 between numbers 4 and 6?
    //how to add number 1 before number 2?
    //how to remove number 2?
    //how to remove the second element in the list?
    //what is printed?
    intIterator = list.listIterator();
    while(intIterator.hasNext()){
      System.out.print(intIterator.next());
    }
  }
}
```

LinkedList versus ArrayList

- A LinkedList typically provides faster element insertion and removal at the list's ends (and middle if using ListIterator)
  - LinkedList methods with index parameters, such as get() or set(), cause the list to be traversed from the first element to the specified element each time the method is called. Thus, using the LinkedLists' get() or set() methods within a loop that iterates through all list elements is inefficient.

- ArrayList offers faster positional access with indices
  - it maintains index based system for its elements as it uses array data structure implicitly which makes it faster for searching an element in the list.

Linked List – Practical Examples

- Image viewer – Previous and next images are linked and can be accessed by the next and previous buttons.
- Previous and next page in a web browser – We can access the previous and next URL searched in a web browser by pressing the back and next buttons since they are linked as a linked list.
- Music Player – Songs in the music player are linked to the previous and next songs. So you can play songs either from starting or ending of the list.
  - https://coderspacket.com/to-create-a-simple-song-playlist-using-linked-list-in-java

Map

Additional Reading: https://www.geeksforgeeks.org/java-util-hashmap-in-java-with-examples/

- What if you had key value pairs?
- Example: Your address points to your house
  - Does a book of addresses, store all the information about your house?
  - Or simply the address, that can get the info?
- Introducing **Maps**
  - Pairs keys to values
  - Keys needs to be unique
- Some uses: database indexing, network routing

```
HashMap<String, String> contacts = new HashMap<>();
contacts.put("awonder", "awonder@wonderland.colostate.edu");
contacts.put("queen", "redqueen@wonderland.colostate.edu");
contacts.put("hatter", "madhatter@wonderland.colostate.edu");
System.out.println(contacts.get("queen")); // prints redqueen@wonderland.colostate.edu
```

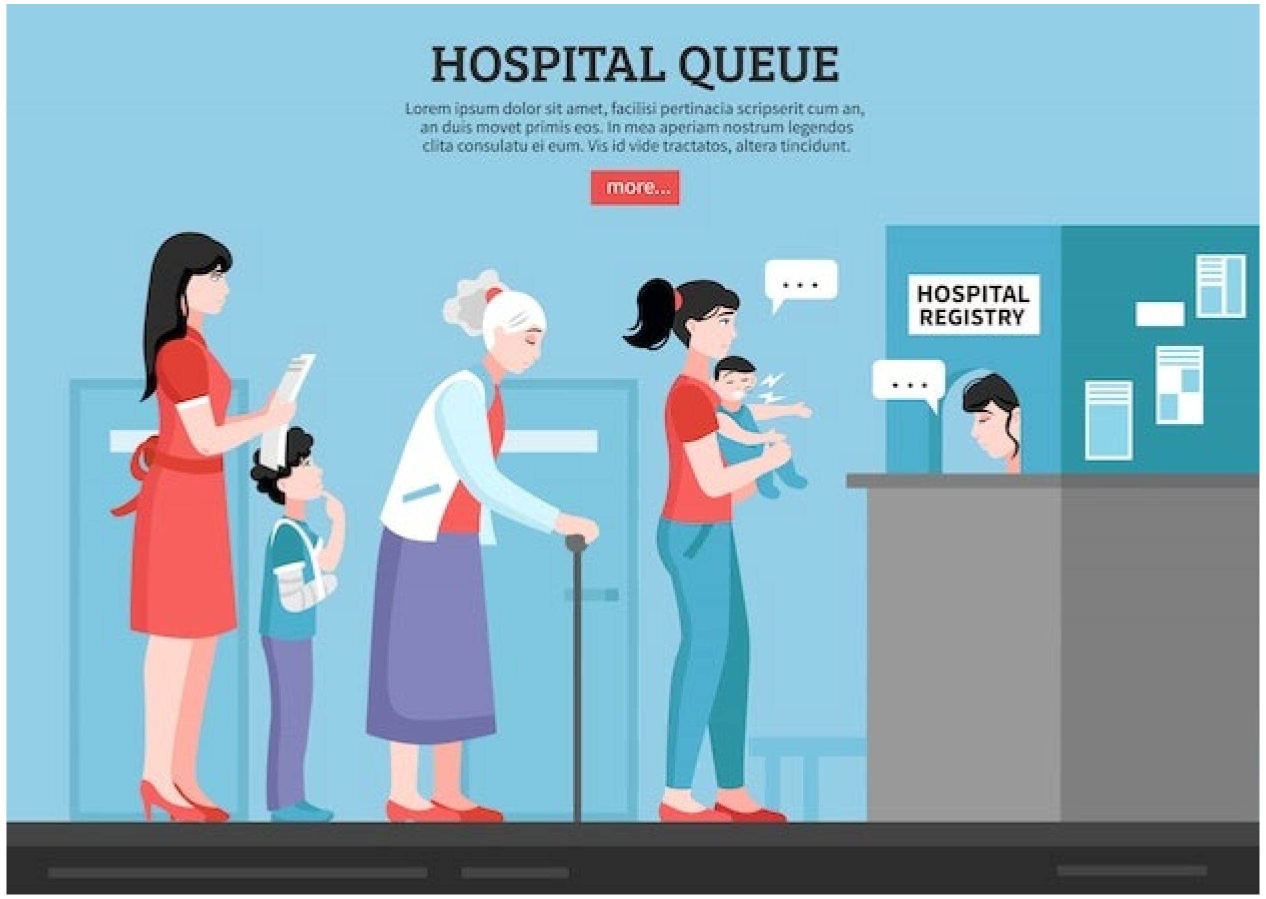
HashMap basic methods and Practice

put()	`put(key, value)` Associates key with specified value. If key already exists, replaces previous value with specified value.
putIfAbsent()	`putIfAbsent(key, value)` Associates key with specified value if the key does not already exist or is mapped to null.
get()	`get(key)` Returns the value associated with key. If key does not exist, return null.
containsKey()	`containsKey(key)` Returns true if key exists, otherwise returns false.

containsValue()	`containsValue(value)` Returns true if at least one key is associated with the specified value, otherwise returns false.
remove()	`remove(key)` Removes the map entry for the specified key if the key exists.
clear()	`clear()` Removes all map entries.
keySet()	`keySet()` Returns a Set containing all keys within the map.
values()	`values()` Returns a Collection containing all values within the map.

```java
import java.util.Collection;
import java.util.HashMap;
import java.util.Set;
public class CountryMap {
   public static void main(String args []){
      HashMap<String, Double> mapCountry = new
HashMap<String, Double>();
      mapCountry.put("Brazil",3288000.0);
      mapCountry.put("USA", 3797000.0);
      mapCountry.put("Portugal", 35603.0);
      mapCountry.put("India", 1269000.0);
      System.out.println(mapCountry);
      //how to return a list of keys?
      System.out.println(keys);
      //how to return a list of values?
      System.out.println(values);
   }
}
```

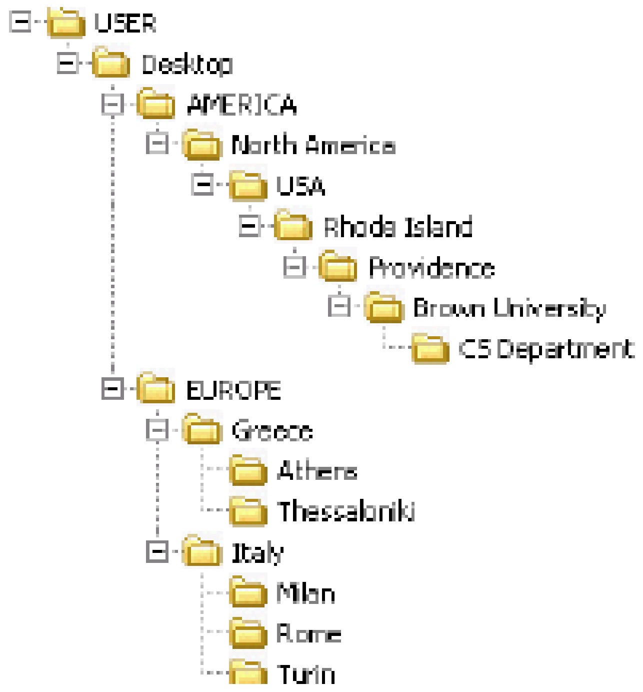
Other Types of Data Structures


HOSPITAL QUEUE
Lorem ipsum dolor sit amet, facilisi pertinacia scripserit cum an, an duis movet primis eos. In mea aperiam nostrum legendos clita consulatu ei eum. Vis id vide tractatos, altera tincidunt.
more...
HOSPITAL REGISTRY
...
...

https://docs.oracle.com/javase/8/docs/api/java/util/Queue.html
https://docs.oracle.com/javase/7/docs/api/java/util/PriorityQueue.html

Other Types of Data Structures

Stack
https://docs.oracle.com/javase/7/docs/api/java/util/Stack.html



Tree
https://docs.oracle.com/javase/7/docs/api/java/util/TreeSet.html

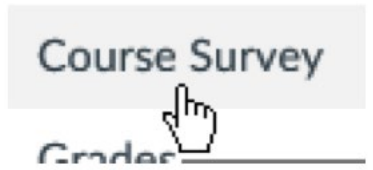
Graph – can be implemented
in Java using a HashMap

Why does it matter?

- Different data structures
  - affects speed, memory and storage
  - important for all fields
    - biology – large datasets
    - graphics – speed is needed
    - cybersecurity - processing serialized information over networks
- If you interview at Google, Amazon, etc
  - they often give you a tech quiz
  - Most of what is on that quiz – you learn in CS 165
- Take CS 165, it provides major programming foundation!

Last but not least

- Thank you!!!
- Please fill out course survey



- Keep coding, keep learning, and have a great time at CS 165!