PAL Week 9

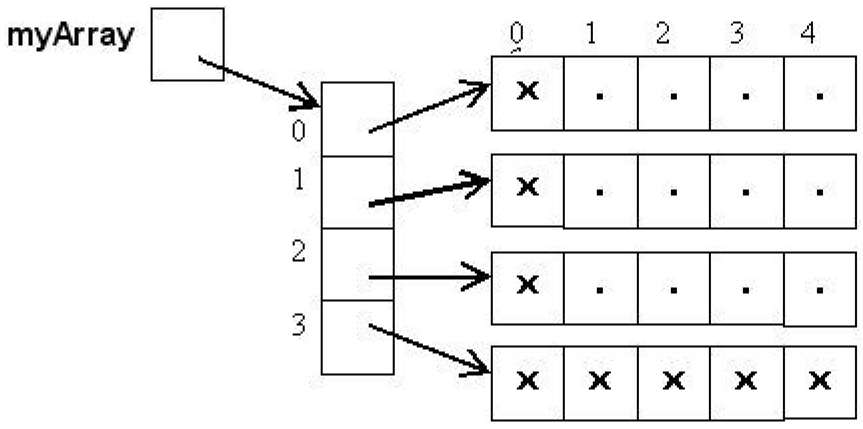
COMP125

Memory Diagrams

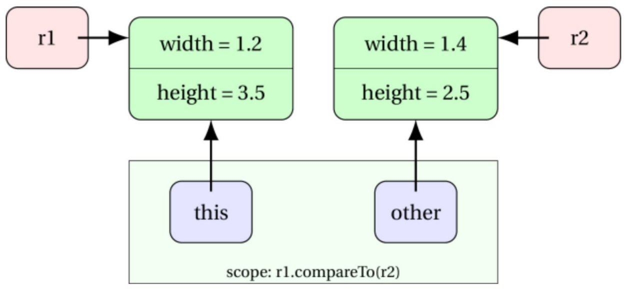
Final Exam Prep

- Arrays / Arrays and Functions
  - https://github.com/comp125mq/comp115_to_comp125_transition_workshops/
  - Shallow and Deep Copy
  - Operating on an array passed to a function
  - Modifying contents of array passed
  - Returning an array from a function

- 2D Arrays



- Objects
  - http://ishare.mq.edu.au/prod/file/d070ad41-c730-4369-a123-f3cf87b375da/1/comp125lectureNotes.zip/03-classes-and-objects/05-copying-objects.html



e.g. of compareTo method

- Class containing array(s)
    - http://ishare.mq.edu.au/prod/file/d070ad41-c730-4369-a123-f3cf87b375da/1/comp125lectureNotes.zip/03-classes-and-objects/07-class-containing-array.html

- Array of objects
    - http://ishare.mq.edu.au/prod/file/d070ad41-c730-4369-a123-f3cf87b375da/1/comp125lectureNotes.zip/03-classes-and-objects/08-array-of-objects.html

Time Complexities

- O(1)
    - Constant Time
    - This means that whatever you are trying to do will not take longer/slower depending on the size of the array (as an example) you are working on.
    - Examples:
        - accessing any single element in an array
        - finding the minimal value in an array sorted in ascending order; it is the first element
    - However, finding the minimal value in an unordered array is not a constant time operation as scanning over each element in the array is needed in order to determine the minimal value (it's O(n))
- O(n)
    - Linear Time
    - This means that the running time increases at most linearly with the size of the input (so, the bigger the input, the more time your method is going to take to run)
    - Examples:
        - Simple for/while loop that iterates over array and

            -
              ```
              for(int i = 0; i < a.length; i++) {
                      //do something with constant time (e.g. count++)
              }
              ```
            - while(i < a.length;)

- $O(\log_2 n)$
  - Logarithmic Time
  - Binary search halves the size of the reasonable portion upon every incorrect guess
  - Best example is **binary search** - binary search uses a divide and conquer strategy (more information here)
    - In binary search, you check the middle item first, then you **halve** the number of items you are working with to try and locate this number you're finding. You keep doing this over and over again until you find the item

- $O(n^2)$
  - Quadratic Time
  - Best way to think about this one is a nested loop, where both of these loops iterate through the entire array

  -
    ```
    for(int i = 0; i < arr.length; i++) {
            for(int j = 0; j < arr.length; j++) {
                    //some constant time operation
                    //e.g. if(arr[i] == arr[j]) count++;
            }
    }
    ```

- Combined Time Complexities
  - You can also have stuff like O(n logn)
    - Not 100% sure this will come up in the exam because you didn't really touch on sorting this semester
    - If you're interested, read up on Quick Sort