CS151 Intro to Data Structures

Midterm Review

Outline

- HW comments
- Review

Announcements

Midterm Wednesday 11/01

Closed note, closed books

HW04 due next Friday 11/10

Releasing it Wednesday the latest

No lab this Wednesday

Homework Comments

Read instructions!

If we ask for specific things, include them

Start them early

Topics

- OOP
  - class design
  - inheritance
  - polymorphism
    - overriding
    - overloading
  - super/sub typing
- Interface
- Exceptions
- Generics

- Array
- ArrayList/Expandable Array
- Linked List
- Stack
- Queue
- Big-O analysis

Exceptions

- Exceptions are objects – create by extending `Exception` or `RuntimeException`
- Thrown/raised when error/unexpected occurs – `throw new MyException()`
- Exception handling is when you catch a raised exception (error recovery) – `try{} catch(myException e){}`

Exceptions

- Subclass of `Exception` are checked exceptions – must be handled
  - either write code to catch
  - or declare with `throws` - `public static void main(String[] args) throws FileNotFoundException`

- Review lab1

Big-O Analysis

A polynomial in terms of input size n

- No loop
  - $O(1)$
- Only loops contribute, and only nonconstant loops
- Each nested term is multiplied
- Each sequential term is summed

Simplify the polynomial

- Identify dominant term – highest degree polynomial
- Polynomials beat polylogs
- Exponentials beat polynomials
- Discard constants

Questions

True / False

Reading code

- what does this do/print?

Analyzing code

what is the big-O?

Writing code

What does this do?

```
public static int mystery(int[] nums) {
  int max = 0; int count = 1;
  for (int i=0; i<nums.length-1; i++) {
    if (nums[i] == nums[i+1]){
      count++;
    }
    else {
      if (count > max) {
        max = count; count = 1;
      }
    }
  }
  if (count>max) max = count;
  return max;
}
```

ArrayList

`insertHalf(ArrayList<Integer> aList)` that finds each even number `num` in `aList` and inserts its half (`num/2`) immediately before `num`.

For example, if the `aList` contains {2,5,7,6,8} initially, it will contain {1,2,5,7,3,6,4,8} after `insertHalf` runs.

Designing Data Structures

- Design a `Course` class that stores a name, department, number and a list of students enrolled. Write its constructor and a method `void enroll(Student s);`

- Design a `CappedCourse` that has a cap on the number of students who can enroll, which inherits from `Course`. Write the associated constructor and override `enroll`. Raise an exception if cap is exceeded.

Linked List

Implement `removeLast()` for `SinglyLinkedList`

Implement `insertAfter(E e, Node<E> n)`

Implement `reverse()` for `SinglyLinkedList`

Make sure you are comfortable with generics

Stack and Queue

- Review array-based stack and queue implementations

- Implement a stack/queue using a linked list
  - `SinglyLinkedList`
  - `DoublyLinkedList`