CompSci 100
Prog Design and Analysis II

\[ \sum_{i=1}^{n} i \]

Sept 9, 2010
Prof. Rodger
Announcements

• RSG assignment due tonight by midnight
• APTs due Tuesday
• Recitation this week – before going look at
  – Frequencies.java
  – APTS – CircleCountry and SoccerLeagues
Top 10 ways to survive in CompSci 100

10. Read the book
9. Keep Randy’s Pizza number handy
8. Learn how to spell Rodger
7. Always attend class, even if you have a program due that day
6. Keep working until it is correct
Top 10 ways to survive in CompSci 100

5. Turn it off, Pay attention
4. Visit Professor, TA, UTAs in office/consulting hours
3. Read the CompSci 100 Bulletin Board
2. Seek help when stuck (Abide by “1 hour rule”)
1. Start programming assignments early
Turn it off, Pay attention

- Consider blogging, networking and tweeting while listening to a speaker
- “The most important reason to stop multitasking so much isn’t to make me feel respected, but to make you exist. If you listen first, and write later, then whatever you write will have had time to filter through your brain, and you’ll be in what you say. This is what makes you exist. If you are only a reflector of information, are you really there?”

Jaron Lanier is a computer scientist, composer, visual artist, and author.
Topdown v Bottomup

• Programming is changing our world
  – Empowering, liberating, equalizing,…

• Everything is a bit: all 0’s and 1’s
  – From jpg to mp3 to …

• It’s about problems! It’s about details!
  – Should we think about problems to get to the details?
  – Should we master details before grand thinking?

• See Wikipedia on topdown v bottomup design
Plan

• OO-Overview
  – Conventions and idioms for object-oriented Compsci 100
  – What is MVC as it applies to code you write and use?

• Prelude to analysis of execution time
  – What is “efficiency”? How to measure it?
  – Engineering, math, and science combined

• How to read code and influence people
  – When you need to understand and when you don’t
  – How to modify code, make additions, inter-operate
Conventions in Compsci 100 projects

• We want you to concentrate on algorithms and data structures
  – Not on rendering fonts, interacting with users
  – This is important! But not what this course is about

• We try to build GUIs or views that facilitate projects
  – You write the brains, we build the skull/manikin
  – Our GUI communicates with your code
    • Requires following conventions in interacting code

• GUI libraries are similar across languages, but...
  – Deeper Java specific details than HashMap
KWIC: Key word in Context

Arise, fair sun, and kill the envious moon, Who
I. Yet I should kill thee with much cherishing.
shortly, for one would kill the other. Thou! why,
those twenty could but kill one life. I beg
wherefore, villain, didst thou kill my cousin? That villain
mean, But 'banished' to kill me—'banished'? O friar,
thou happy. Tybalt would kill thee, But thou slewest
cell there would she kill herself. Then gave I
heaven finds means to kill your joys with love!

• Read file, find word and it’s context, print
  – Can find all words, but how do we get context?
  – Loop and look: create context line for each occurrence
  – See ContextModel.java
Use KWIC example to motivate study

• Dissect and inspect KWIC code to understand conventions
  – Understand Model and View interaction
  – Facilitates doing RSG and Markov text-generation

• Review some basic coding idioms and ideas
  – Avoiding recomputing same value, readability, modifiability, ...

• Errors: possible for a method to fail but still work?
  – See KWIC code when same key occurs twice on a line!
MVC Example, key-word-in-context

• User loads file
  – Where? Communicate to?
  – What changes in model?
  – What happens in view?

• User chooses word
  – Process in Model
  – Alternatives?
  – Generate context, display
  – How to show in any view?
Key Word in Context Explained

• For every different word, store where it occurs
  – *love* is the 1\(^{st}\), 3\(^{rd}\), 50\(^{th}\), and 1237\(^{th}\) word in the file

• This data is kept in a map, key is word, value is ??
  – How do we generate the data in the map?

• Given a word, how do we find its context? How do we format?
  – All words are in an array, in order
  – Memory concerns?
  – Original KWIC paper by Parnas as comparison
Code Interlude

• Examine ContextModel.process
  – Called when user enters word, parameter is the word
  – If file already read, we don’t need map, where is this?
  – Error checking? When and when happens
  – How does Model communicate back to View?

• Examine ContextModel.justify
  – What is method doing
  – What is parameter, where was it constructed, issues?
  – What about ‘magic numbers’, e.g., 30?
  – What about comments, should we add some?
public class ContextMain {
    public static void main(String[] args) {
        ContextModel model = new ContextModel();
        SimpleViewer view =
            new SimpleViewer("Compsci 100 KWIC", "word");
        view.setModel(model);
    }
}

• What changes in above, e.g., for RSG assignment?
  – How can view communicate with any model?
  – View doesn’t change, model does!
    • Requires using a Java interface to capture commonality
Model View Controller, MVC

• Gui is the View and often the controller
  – Separate user-interaction from updates to data
• User loads file, chooses word, ...
  – Model notified, computes, updates view
• Model has all the state and knows when it changes
  – Communicates changes to views (via controller)
  – Must be initialized, updated, etc.

• Very common Design Pattern
  – Capture common solutions to problems in a context
  – Iterator, Composite, Decorator seen in Compsci 100
Convention Summary

• Classes start with capital letter and then we have:
  – They’re public, except nested class? Protected means …
  – camelCaseForMethods and ForClasses
  – Ivars, fields, instance variables, mySize, myMap, …
  – Constants (public static) are ALL_CAPS

• Interfaces are IModel, IView, and so on
  – Not true for standard Java classes, yes for Compsci 100
  – Don’t need to label methods as abstract, but can

• Supply AbstractDefault implements IThing
  – Constructor, some state, some common behavior: extend!
Eugene (Gene) Myers

• Lead computer scientist/software engineer at Celera Genomics, then at Berkely, now at Janelia Farms Research Institute (HHMI)

• BLAST and WG-Shotgun

"What really astounds me is the architecture of life. The system is extremely complex. It's like it was designed." ... "There's a huge intelligence there."
Methods, Interfaces, Inheritance

• A method by any other name would smell as sweet
  – Method in OO languages, functions, procedures in others
  – Parameters and return value: communication
    • Do objects or methods communicate?: OO v procedural

• Static: Math.sqrt, Character.toUpperCase, ...
  – Don’t belong to an object, invoked via class (clue above?)
  – Java API helpful here

• Interface: implement class with required, related methods
  – HashMap, TreeMap
  – ArrayList, LinkedList, Vector
Interfaces continued

• In the beginning
  – Make code work, don’t worry about generalizing
  – But, if you write code using `Map` rather than `TreeMap`
    • Can swap in a new implementation, coded generally!

• Don’t know how to optimize: space or time
  – Facilitate change: use interface rather than concrete class
  – My DVD connects to my TV, regardless of brand, why?
  – How do you turn on a Nokia cell phone? Motorola? But!

• Interfaces facilitate code refactoring
  – Don’t add functionality, change speed or memory or …
What does Object-Oriented mean?

• Very common method of organizing code
  – Design classes, which encapsulate state and behavior
  – Some classes can be similar to, but different from their parent class: inheritance
    • Super class, subclass
  – Inherit behavior, use as is or modify and use *or both*

• Complex to design a hierarchy of classes, but important
  – More of this in Compsci 108 or on-the-job training
  – We’re solving simple problems, not designing re-usable libraries

• Simple does not mean straight-forward, not Vista!
Inheritance and Interfaces

• Interfaces provide method names and parameters
  – The method signature we can expect and thus use!
  – What can we do to an ArrayList? To a LinkedList?
  – What can we do to a Map or Set or PriorityQueue?
  – `java.util.Collection` is an interface

• Abstract classes can implement core, duplicated code
  – If we can add one object to a [set, map, list], can we add an entire list of objects?
    `java.util.AbstractCollection`
  – If we can iterate can we remove? Convert to array? Obtain size?
Random Java stuff

• What happens when you call new, when do you?
  – Creates object, assign reference/pointer somewhere (or?)
  – Two ‘labels/variables’ share same object, consequences?
    • Why isn’t this a concern with String objects/labels?

• What about int, double, byte, char, long
  – Related, but different. What’s the same, what’s different?
  – Range of values for each?
  – Arithmetic with each?
  – Casting vs promotion (pass int to Math.sqrt?)
Jaron Lanier is a computer scientist, composer, visual artist, and author. He coined the term ‘Virtual Reality’ … he co-developed the first implementations of virtual reality applications in surgical simulation, vehicle interior prototyping, virtual sets for television production, and assorted other areas.

In 2010 writes: “You are Not a Gadget”

"What's the difference between a bug and a variation or an imperfection? If you think about it, if you make a small change to a program, it can result in an enormous change in what the program does. If nature worked that way, the universe would crash all the time."
Analysis: Algorithms and Data Structures

• We need a vocabulary to discuss performance
  – Reason about alternative algorithms and implementations
  – It’s faster! It’s more elegant! It’s safer! It’s less filling!

• Need empirical tests, analytical/mathematical tools
  – Given two methods, which is better? Run them to check.
    • 30 seconds vs. 3 seconds, easy. 5 hours vs. 2 minutes, harder
    • What if it takes two weeks to implement the methods?
  – Use mathematics to analyze the *algorithm*,
  – The implementation is another matter, cache, compiler optimizations, OS, memory,...
How fast does the code run?

• “As soon as an Analytical Engine exists, it will necessarily guide the future course of the science. Whenever any result is sought by its aid, the question will then arise — by what course of calculation can these results be arrived at by the machine in the shortest time?”
  
  — Babbage, Analytical Engine 1864

• Question: What’s the fastest way to sort a million 32-bit integers?
• Obama’s answer: I don’t think the bubble sort is the way to go
Quantitative Measurements of Code

• Typically measure running time (memory?)
  – Other things to measure?
  – What about wall-clock v CPU time? Java: wall-clock

• Typically change size of input/problem to validate runtime hypotheses
  – Not the data itself, but the number of data items
  – Size of string vs. number of strings in array?

• Doubling hypothesis: What effect does doubling input size have on running time?
  – Linear: time doubles, quadratic: factor of four, ...
Different measures of complexity

• Worst case
  – Gives a good upper-bound on behavior
  – Never get worse than this
  – Drawbacks?

• Average case
  – What does average mean?
  – Averaged over all inputs? Assuming uniformly distributed random data?
  – Drawbacks?

• Best case
  – Linear search, useful?
Notations for measuring complexity

• O-notation or big-Oh: $O(n^2)$ is used in most algorithmic analysis, e.g., Compsci 130 at Duke. It’s an upper bound in the limit
  – Correct to say that linear algorithm is $O(n^2)$, but useful?

• Theta-notation or $\Theta(n^2)$ is a tight bound, solid guarantee that algorithmic analysis is exact, both upper and lower bound

• Omega is lower bound: $\Omega(n \log n)$ is a lower bound for comparison based sorts
  – Can’t do better than that, very hard to prove

• Sedgewick/Wayne uses tilde notation $\sim n^2$ means leading term is $n$ squared
  – We’ll use this, but abuse big-Oh since we want “best” answer

CompSci 100, Fall 2010
Big-Oh, O-notation: concepts & caveats

• Count how many times “simple” statements execute
  – In the body of a loop, what matters? (e.g., another loop?)
  – Assume simple statements take a second, cost a penny,…
    • What’s good, what’s bad about this assumption?
    • Alternatives?

• In a recursive method what do we do? Recurrence relations
  – Like a loop, but easier! (and trickier) to analyze

• In real life: cache behavior, memory behavior, swapping behavior, library gotchas, things we don’t understand,…
Multiplying and adding big-Oh

• Suppose we do a linear search then do another one
  – What is the complexity?
  – If we do 100 linear searches?
  – If we do n searches on a vector of size n?

• Binary search followed by linear search?
  – What are big-Oh complexities? Sum?
  – What about 50 binary searches? What about n searches?

• What is the number of elements in the list (1,2,2,3,3,3);
  (1,2,2,3,3,3,4,4,4,4)?
  – What about (1,2,2, ..., n,n,...,n)?
Helpful formulae

• We always mean base 2 unless otherwise stated
  – What is \( \log(1024) \)?
  – \( \log(xy) = \log(x) + \log(y) \)
  – \( \log(x^y) = y \log(x) \)
  – \( \log(2^n) = n \log(2) \)
  – \( 2^{\log n} = n \)

• Sums (also, use sigma notation when possible)
  – \( 1 + 2 + 4 + 8 + \ldots + 2^k = 2^{k+1} - 1 = \sum_{i=0}^{k} 2^i \)
  – \( 1 + 2 + 3 + \ldots + n = \frac{n(n+1)}{2} = \sum_{i=1}^{n} i \)
  – \( a + ar + ar^2 + \ldots + ar^{n-1} = \frac{a(r^n - 1)}{(r-1)} = \sum_{i=0}^{n-1} ar^i \)
Complexity: ideas and measurements

• In word-counting program, avoiding rescan of text?
  – Store and lookup in map, avoid counting more than once
  – Changed from UT to T*\log T to T + U (or T + U*\log U)

• In Markov assignment we’ll do something similar
  – It’s easy to rescan text, store entire text and no more
  – Avoid rescanning requires storing more information
  – If we only do something once, should we optimize?

• Before coding analysis: what approach will work?
  – Why is this a good idea?
Sergey Brin

• Simple ideas sometimes can change the world [wikipedia]
  – Works because of scale
  http://www.youtube.com/watch?v=Ka9IwHNykfU

• Co-created pagerank (Larry Page), which evaluates links to a page and the importance of those links, based on the importance of the page from which the links come which ...

http://upload.wikimedia.org/wikipedia/commons/0/00/Sergey_Brin_2008.jpg