COMPSCI 100, Fall 2006
Owen Astrachan

http://www.cs.duke.edu/courses/cps100/fall06
http://www.cs.duke.edu/~ola

What is Computer Science?

What is it that distinguishes it from the separate subjects with which it is related? What is the linking thread which gathers these disparate branches into a single discipline? My answer to these questions is simple --- *it is the art of programming a computer*. It is the art of designing efficient and elegant methods of getting a computer to solve problems, theoretical or practical, small or large, simple or complex.

C.A.R. (Tony)Hoare

Programming != Computer Science

- What is the nature of intelligence? How can one predict the performance of a complex system? What is the nature of human cognition? Does the natural world 'compute'?
- It is the interplay between such fundamental challenges and the human condition that makes computer science so interesting. The results from even the most esoteric computer science research programs often have widespread practical impact. Computer security depends upon the innovations in mathematics. Your Google search for a friend depends on state-of-the-art distributed computing systems, algorithms, and artificial intelligence.

http://www.post-gazette.com/pg/pp/04186/341012.stm

Efficient *design, programs, code*

Using the language: Java (or C++, or Python, or …), its idioms, its idiosyncracies

Object-oriented design and patterns. Software design principles transcend language, but …

Know data structures and algorithms. Trees, hashing, binary search, sorting, priority queues, greedy methods, graphs …

Engineer, scientist: what toolkits do you bring to programming? Mathematics, design patterns, libraries --- standard and others…

Course Overview

- Active Lectures, Recitations???, Quizzes, Programs
  - Recitation based on questions given out in previous week
    - Discuss answers, answer new questions, small quiz
    - More opportunities for questions to be answered.
  - Active Lectures based on readings, questions, programs
    - Online quizzes used to motivate/ensure reading
    - In-class questions used to ensure understanding
  - Programs
    - Theory and practice of data structures and OO programming
    - Fun, practical, tiring, …
    - Weekly APT programs and longer programs
- Exams/Tests
  - Semester: open book
  - Final: open book

Questions

If you gotta ask, you'll never know
Louis Armstrong: "What's Jazz?"

If you gotta ask, you ain't got it
Fats Waller: "What's rhythm?"

What questions did you ask today?
Arno Penzias

Tradeoffs

Programming, design, algorithmic, data-structural

Simple, elegant, quick, efficient: what are our goals in programming? What does XP say about simplicity? Einstein?

Fast programs, small programs, run anywhere-at-all programs. Runtime, space, your time, CPU time…

How do we decide what tradeoffs are important? Tension between generality, simplicity, elegance, …

Problem Solving and Programming

- How many words are in a file?
  - What's a word?
  - What's a file?
  - How do we solve this: simply, quickly, …?
    - What's the best we can do? Constraints?
- How many different words are in a file?
  - How is this similar? Different?
- How many words do two files have in common?
  - Spell-checking, did you mean ..?

OO design in code/wordcount

- Count number of different words in an array, how can we accommodate more than one approach?
  - Why do we say array rather than file?

```
public interface IUniqueCounter {
    public int uniqueCount(String[] list);
}
```

- Three (or more) approaches:
  -
  -
  -

Fast, cheap, out-of-control?

- This is valid and correct Java code, questions?
  - What about `HashSet`?

```
import java.util.*;

public class SetUniqueCounter
             implements IUniqueCounter {

  public int uniqueCount(String[] list) {
     TreeSet<String> set = new TreeSet<String>();
     set.addAll(Arrays.asList(list));
     return set.size();
  }
}
```


How fast is fast? How cheap is cheap?

- How do we measure how fast the code/design is?
  - Can we implement this design in C++?
  - Can we implement this in Python?
- We want a measure that's independent of language?
  - What are we measuring? Express answer?
  - Units? Best case? Average? Worst?
- What is answer using recognized terminology?
  -
  -

What is Computer Science?

- Computer science is no more about computers than astronomy is about telescopes.

Edsger Dijkstra

- Computer science is not as old as physics; it lags by a couple of hundred years. However, this does not mean that there is significantly less on the computer scientist's plate than on the physicist's: younger it may be, but it has had a far more intense upbringing!

Richard Feyneman

http://www.wordiq.com

How do we use `SetUniqueCounter`?

- Code below doesn't compile, what's missing (not much)?

```
public class UniqueDemo {

  public static void main(String[] args) {

   Scanner s = new Scanner(new File("kjv10.txt"));
   ArrayList<String> list = new ArrayList<String>();
   while (s.hasNext()) {
       list.add(s.next());
   }
   String[] arr = list.toArray(new String[0]);

   IUniqueCounter uc = new SetUniqueCounter();
   int count = uc.uniqueCount(arr);
   System.out.println("unique count: "+count);
  }
}
```


Some Java Vocabulary and Concepts

- Java has a huge standard library
  - Organized in *packages*: java.lang, java.util, javax.swing, …
  - API browseable online, but Eclipse IDE helps a lot
- Java *methods* have different kinds of access inter/intra class
  - Public methods …
  - Private methods …
  - Protected and Package methods …
- *Primitive* types (int, char, double, boolean) are not objects but everything else is literally an *instance* of class *Object*
  - `foo.callMe();`

Basic data structures and algorithms

- Arrays are typed and fixed in size when created
  - Don't have to fill the array, but cannot expand it
  - Can store `int`, `double`, `String`, …
- ArrayList (and related class Vector and interface List) grows
  - Stores objects, not primitives
    - Autoboxing in Java 5 facilitates `int` to/from `Integer` conversion
  - Accessing elements can require a *downcast*
    - *This has changed in Java 5 if ArrayList is typed*
  - ArrayList objects grow themselves intelligently
- `java.util` package has lots of data structures and algorithms
  - Use rather than re-implement, but know how do to do both

Tracking different/unique words

- We want to know how many times 'the' occurs
  - Do search engines do this? Does the number of occurrences of "basketball" on a page raise the priority of a webpage in some search engines?
    - Downside of this approach for search engines?
- Constraints on solving this problem
  - We must read every word in the file (or web page)
  - Search for the word? Avoid counting twice? Store?
  - Are there fundamental limits on any of these operations? *Where should we look for data structure and algorithmic improvements?*

How stuff works: invariant? Problems?

```
public class SlowUniqueCounter implements IUniqueCounter{

    public int uniqueCount(String[] list) {
        int count = 0;
        int diffSize = list.length;
  // invariant: strings in list[0..k] are unique
        for(int k=0; k < diffSize; k++){
            String word = list[k];
            count++;
            for(int j=k+1; j < diffSize; j++){
                if (list[j].equals(word)){
                    list[j] = list[diffSize-1];
                    diffSize--;
                }
            }
        }
        return count;
    }
}
```


Search: measuring performance

- How fast is fast enough?

```
/**
 * @return true if key in a, else return false
 */
boolean search(String[] a, String key){
    for(int k=0; k < a.length; k++)
        if (a[k].equals(key)) return true;
    return false;
}
```

- Java details: parameters? Return values? ArrayLists?
  - See next page for alternate code
- How do we measure performance of code? Of algorithm?
  - Does processor make a difference? G5? Pentium? 64-bit?

Six of one and …

```
boolean search(String[] a, String key){
    for(int k=0; k < a.length; k++)
        if (a[k].equals(key)) return true;
    return false;
}
boolean search(String[] a, String key){
    for(String s : a)
        if (s.equals(key)) return true;
    return false;
}
boolean search(String[] a, String key){
    return Arrays.asList(a).indexOf(key) != -1;
}
```

- Which is better? By what metric?
  - Iterable object: array, ArrayList, Set, Map, …
  - What do we need to call/invoke a method?

Tradeoffs in processing and counting

- Read words, then sort, determine # unique words?
  - frog, frog, frog, rat, tiger, tiger, tiger, tiger
- If we look up words one-at-a-time and bump counter if we haven't seen a word, is this slower than previous idea?
  - How do we look up word, how do we add word
- Are there kinds of data that make one approach preferable?
  - What is best case, worst case, average case?

Benefits of inheritance, interfaces

- Consider new algorithm for determining unique word count

```
public static void test(IUniqueCounter uc,
                        String[] list){

 double start = System.currentTimeMillis();
 int count = uc.uniqueCount(list);
 double end = System.currentTimeMillis();
 System.out.println(count+" unique words");
 System.out.println((end-start)/1000+" seconds");
}
```

- Why can we pass different kinds of objects to `test`?
  - Why is this an advantage?
  - Inheritance and late/dynamic binding

Inheritance and interfaces

- First view: exploit common interfaces in programming
  - Iterators in Java (java.util.Iterator is an interface)
  - Implementation varies while interface stays the same
- Second view: share code, factor code into parent class
  - Code in parent class shared by subclasses
  - Subclasses can *override* inherited method
    - Subclasses can override and call
- Polymorphism/late(runtime) binding (compare: static)
  - Function actually called determined when program runs, not when program is compiled

Who is Alan Perlis?

- It is easier to write an incorrect program than to understand a correct one
- Simplicity does not precede complexity, but follows it
- If you have a procedure with ten parameters you probably missed some
- If a listener nods his head when you're explaining your program, wake him up
- Programming is an unnatural act
- Won first Turing award

http://www.cs.yale.edu/homes/perlis-alan/quotes.html

Computer Science in a Nutshell