Welcome!

Program Design and Analysis II for Engineers
CompSci 100E
Perkins 107
M, W 4:25-5:40

Professor: Jeffrey Forbes

http://www.cs.duke.edu/courses/fall09/cps100e

What is Computer Science?

- What does a computer scientist do?
- What does a programmer do?
- What are the various subfields of computer science? What binds them together?
- What do you want to do?
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/pg/04186/341012.htm
Efficient *design, programs, code*

Using the language: Java (or C++, or Matlab, 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, Labs, Quizzes, Programs**
  - Labs based on questions given out in previous week
    - Hands-on practice with programming
    - 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/note
  - Final: open book/note

---
Tradeoffs

Programming, design, algorithmic, data-structural

Simple, elegant, quick, efficient: what are our goals in programming?
Don’t worry about getting it right the first time.

Fast programs, small programs,

Runtime, space, your time, CPU time...

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

Time vs. space

Languages

Instead of imagining that our main task is to instruct a computer what to do, let us concentrate rather on explaining to human beings what we want a computer to do. - Donald Knuth

- Machine languages.
- Natural languages.
- High-level programming languages.

- Kids Make Nutritious Snacks.
- Red Tape Holds Up New Bridge.
- Police Squad Helps Dog Bite Victim.
- Local High School Dropouts Cut in Half.

[ real newspaper headlines, compiled by Rich Pattis ]

Pictures
- Natural language (English)
- Pseudo-code
- Specific high-level programming language
- Machine Language

More precisely expressed

© Sedgewick & Wayne
Why Java?

- **Java features.**
  - Widely used.
  - Widely available.
  - Embraces full set of modern abstractions.
  - Variety of automatic checks for mistakes in programs.

- **Java economy.**
  - Mars rover.
  - Cell phones.
  - Blu-ray Disc.
  - Web servers.
  - Medical devices.
  - Supercomputing.
  - ...

---

I'm one of the few crazies... who believes it’s very possible the Internet has been underhyped instead of overhyped... I predict over the next 90 days Java is going to be like a drug you rub over venture capitalists and they go crazy. - John Doerr

There are only two kinds of programming languages: those people always [gripe] about and those nobody uses. - Bjarne Stroustrup

---

© Sedgewick & Wayne
Why Java?

- **Java features.**
  - Widely used.
  - Widely available.
  - Embraces full set of modern abstractions.
  - Variety of automatic checks for mistakes in programs.

- **Caveat.** No perfect language.

- **Our approach.**
  - Minimal subset of Java.
  - Develop general programming skills that are applicable to:
    C, C++, C#, Perl, Python, Ruby, Matlab, Fortran, Fortress, ...

---

A Rich Subset of the Java Language

<table>
<thead>
<tr>
<th>Built-In Types</th>
<th>System</th>
</tr>
</thead>
<tbody>
<tr>
<td>int, double</td>
<td>System.out.println()</td>
</tr>
<tr>
<td>long, String</td>
<td>System.out.println()</td>
</tr>
<tr>
<td>char, boolean</td>
<td>System.out.println()</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Math Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>Math.sin()</td>
</tr>
<tr>
<td>Math.cos()</td>
</tr>
<tr>
<td>Math.log()</td>
</tr>
<tr>
<td>Math.exp()</td>
</tr>
<tr>
<td>Math.sqrt()</td>
</tr>
<tr>
<td>Math.pow()</td>
</tr>
<tr>
<td>Math.min()</td>
</tr>
<tr>
<td>Math.max()</td>
</tr>
<tr>
<td>Math.abs()</td>
</tr>
<tr>
<td>Math.PI</td>
</tr>
<tr>
<td>Math.abs()</td>
</tr>
<tr>
<td>Math.exp()</td>
</tr>
<tr>
<td>Math.cos()</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Flow Control</th>
</tr>
</thead>
<tbody>
<tr>
<td>if</td>
</tr>
<tr>
<td>for</td>
</tr>
<tr>
<td>else</td>
</tr>
<tr>
<td>while</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parsing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer.parseInt()</td>
</tr>
<tr>
<td>Double.parseDouble()</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Punctuation</th>
</tr>
</thead>
<tbody>
<tr>
<td>( )</td>
</tr>
<tr>
<td>,</td>
</tr>
<tr>
<td>;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>String</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
</tr>
<tr>
<td>&quot; &quot;</td>
</tr>
<tr>
<td>length()</td>
</tr>
<tr>
<td>comparesTo()</td>
</tr>
<tr>
<td>charAt()</td>
</tr>
<tr>
<td>matches()</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Arrays</th>
</tr>
</thead>
<tbody>
<tr>
<td>a[i]</td>
</tr>
<tr>
<td>new</td>
</tr>
<tr>
<td>a.length</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Objects</th>
</tr>
</thead>
<tbody>
<tr>
<td>class</td>
</tr>
<tr>
<td>static</td>
</tr>
<tr>
<td>public</td>
</tr>
<tr>
<td>private</td>
</tr>
<tr>
<td>toString()</td>
</tr>
<tr>
<td>equals()</td>
</tr>
<tr>
<td>new</td>
</tr>
<tr>
<td>main()</td>
</tr>
</tbody>
</table>
Programming in Java

- Programming in Java.
  - Create the program by typing it into a text editor, and save it as HelloWorld.java

```java
/****************************
 * Prints "Hello, World"
 * Everyone's first Java program.
 ****************************/

public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, World");
    }
}
```

HelloWorld.java

Compile it using Eclipse or by typing at the command-line:
```
javac HelloWorld.java
```
(or click the Save button in Eclipse)

- This creates a Java bytecode file named: HelloWorld.class
Java Bytecode

```
000000 312 376 272 276 \0 0 035 \n 0 06 0 017 \
000020 \0 0 20 \0 0 21 \b 0 0 22 \n 0 023 \0 0 24 007 \0 025 007
000040 \0 0 26 001 \0 0 06 < i n i t > 001 \0 0 03 { ( )
000060 \0 001 \0 0 04 C o d e 001 \0 017 L i n e N
000100 u n m e r T a b l e 001 \0 0 04 m a i
000120 n 001 \0 0 26 { [ L ] a v a / l a n g
000140 / S t r i n g ; } V 001 \0 0 \n S o u
000160 r c e F i l e 001 \0 017 H e l l o W
000200 o r l d . j a v a / f \0 0 07 \0 \b 0 0 07 \0
000220 0 27 \f \0 0 30 \0 0 31 001 \0 \f H e l l o o .
000240 W o r l d 007 \0 0 32 \f \0 0 33 \0 0 34 001 \0 \n
000260 H e l l o W o r l d 001 \0 020 j a v
000300 a / l a n g / O b j e c t 001 \0 0 20
000320 j a v a / l a n g / S y s t e m
000340 0 03 0 0 03 0 025 L j a v a / i
000360 o / P r i n t S t r e a m ; 001 \0
000400 0 23 j a v a / i o / P r i n t S t
000420 r e a m 001 \0 0 07 p r i n t l n 001 \0
000440 0 25 { L j a v a / l a n g / S t r
000460 i n g ; } V \0 l \0 0 05 \0 0 06 \0 \0 \0 \0
000500 \0 0 02 \0 0 01 \0 0 07 \b 0 0 01 \0 \t \0 \0 \0 035
000520 \0 0 01 \0 0 01 \0 \0 \0 0 06 * 267 \0 0 01 261 \0 \0 \0
000540 \0 0 01 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0
000560 0 13 \0 \f \0 0 01 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0
000600 \0 \0 \0 262 \0 0 02 023 266 \0 0 04 261 \0 \0 \0 \0 \0
000620 \0 0 \0 \0 \0 0 02 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0
000640 \0 0 01 \0 0 0 \0 0 \0 0 002 \0 0 016
000652
```

© Sedgewick & Wayne

A real problem?

- **Text clouds**: A simple yet powerful idea
  - Visualization of most frequently occurring words within some body of text
  - Color or font size indicates word frequency
  - Close cousin: Tag clouds

- **What is involved with generating text clouds?**
  - Steps? Issues?
  - See `SimpleCloudMaker.java`
Data processing

- Scan a large (~ $10^7$ bytes) file
- Print the words together with counts of how often they occur
- Need more specification?

- How do you do it?

- What is we only wanted the top $k$ (say 20) words?

Possible solutions

1. Use heavy duty data structures (Knuth)
   - Hash tries implementation
   - Randomized placement
   - Lots o’ pointers
   - Several pages

2. UNIX shell script (Doug McIroy)
   ```
   tr -cs "[:alpha:]" "[\n*]" < FILE | \
   sort | \
   uniq -c | \
   sort -n -r -k 1,1
   ```

3. See SimpleWordCount.java
   - Which is better?
     - K.I.S.?
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 ..?

Basics

- Definitions.

- Trace.
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 5/6 facilitates int to/from Integer conversion
  - Accessing elements can require a *downcast*
  - 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
Built-In Data Types

- **Data type.** A set of values and operations defined on those values.

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
<th>Literals</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>char</td>
<td>characters</td>
<td>'A' 'G'</td>
<td>compare</td>
</tr>
<tr>
<td>String</td>
<td>sequences of characters</td>
<td>&quot;Hello World&quot; &quot;CS is fun&quot;</td>
<td>concatenate</td>
</tr>
<tr>
<td>int</td>
<td>integers</td>
<td>17 12345</td>
<td>add, subtract, multiply, divide</td>
</tr>
<tr>
<td>double</td>
<td>floating point numbers</td>
<td>3.1415 6.022e23</td>
<td>add, subtract, multiply, divide</td>
</tr>
<tr>
<td>boolean</td>
<td>truth values</td>
<td>true false</td>
<td>and, or, not</td>
</tr>
</tbody>
</table>

Integers

- **Integers:** the `int` data type.
  - Values: integers between $-2^{31}$ and $2^{31} - 1$.
  - Operations: add, subtract, multiply, divide, remainder.
  - Useful for expressing algorithms.

<table>
<thead>
<tr>
<th>expression</th>
<th>value</th>
<th>comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 + 3</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>5 - 3</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>5 * 3</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>5 / 3</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>5 % 3</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>1 / 0</td>
<td></td>
<td>runtime error</td>
</tr>
<tr>
<td>3 * 5 - 2</td>
<td>13</td>
<td>* has precedence</td>
</tr>
<tr>
<td>3 + 5 / 2</td>
<td>5</td>
<td>/ has precedence</td>
</tr>
<tr>
<td>3 - 5 - 2</td>
<td>-4</td>
<td>left associative</td>
</tr>
<tr>
<td>(3 - 5) - 2</td>
<td>-4</td>
<td>better style</td>
</tr>
<tr>
<td>3 - (5 - 2)</td>
<td>0</td>
<td>unambiguous</td>
</tr>
</tbody>
</table>

Typical `int` expressions
# Integer Operations

```java
public class IntOps {
  public static void main(String[] args) {
    int a = Integer.parseInt(args[0]);
    int b = Integer.parseInt(args[1]);
    int sum = a + b;
    int prod = a * b;
    int quot = a / b;
    int rem = a % b;
    System.out.println(a + " + " + b + " = " + sum);
    System.out.println(a + " * " + b + " = " + prod);
    System.out.println(a + " / " + b + " = " + quot);
    System.out.println(a + " % " + b + " = " + rem);
  }
}
```

- `javac IntOps.java`
- `java IntOps 1234 99`

```
1234 + 99 = 1333
1234 * 99 = 122166
1234 / 99 = 12
1234 % 99 = 46
```

# Leap Year

**Q.** Is a given year a leap year?

**A.** Yes if either (i) divisible by 400 or (ii) divisible by 4 but not 100.

```java
public class LeapYear {
  public static void main(String[] args) {
    int year = Integer.parseInt(args[0]);
    boolean isLeapYear;

    // divisible by 4 but not 100
    isLeapYear = (year % 4 == 0) && (year % 100 != 0);

    // or divisible by 400
    isLeapYear = isLeapYear || (year % 400 == 0);

    System.out.println(isLeapYear);
  }
}
```

- `javac LeapYear.java`
- `java LeapYear 2004`
- `java LeapYear 1900`
- `java LeapYear 2000`

```
true
false
true
```
Type Conversion

- **Type conversion.** Convert from one type of data to another.
  - Automatic: no loss of precision; or with strings.
  - Explicit: cast; or method.

```java
public class RandomInt {
    public static void main(String[] args) {
        int N = Integer.parseInt(args[0]);
        double r = Math.random();
        int n = (int) (r * N);  // double to int (cast)
        System.out.println("random integer is "+n);
    }
}
```

<table>
<thead>
<tr>
<th>String to int (method)</th>
<th>double to int (cast)</th>
<th>int to double (automatic)</th>
</tr>
</thead>
<tbody>
<tr>
<td>java RandomInt 6</td>
<td>random integer is 3</td>
<td></td>
</tr>
<tr>
<td>java RandomInt 6</td>
<td>random integer is 0</td>
<td></td>
</tr>
<tr>
<td>java RandomInt 10000</td>
<td>random integer is 3184</td>
<td></td>
</tr>
</tbody>
</table>

Control Flow

- **Control flow.**
  - Sequence of statements that are actually executed in a program.
  - Conditionals and loops: enable us to choreograph control flow.

```
statement 1
     |
     v
statement 2
     |
     v
statement 3
    /|
   / \
statement 4
```

```
boolean 1
     |
true  | false
     |
statement 1
     |
     v
boolean 2
     |
true  | false
     |
statement 1
     |
     v
statement 3
```

- straight-line control flow
- control flow with conditionals and loops
If-Else Statement

- **The if-else statement.** A common branching structure.
  - Check boolean condition.
  - If true, execute some statements.
  - Otherwise, execute other statements.

```java
if (boolean expression) {
    statement T;
} else {
    statement F;
}
```

If-Else: Leap Year

- **If-else.** Take different action depending on value of variable.
  - If `isLeapYear` is true, then print "is a".
  - Otherwise, print "isn't a ".

```java
System.out.print(year + " ");
if (isLeapYear) {
    System.out.print("is a");
} else {
    System.out.print("isn't a");
} System.out.println(" leap year");
```
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 Feynman

http://www.wordiq.com

Computer Science in a Nutshell?