

# 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/fall109/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/pp/04186/341012.stm>

## 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...**

**Time vs. space**

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

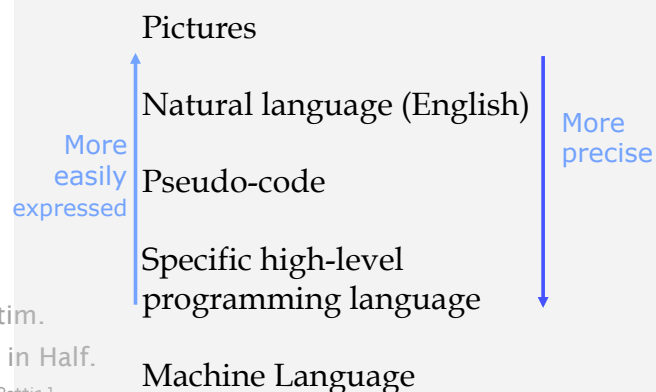
## Languages

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.**

- 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 ]



- **High-level programming languages.**

## 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.
  - ...



James Gosling  
<http://java.net/jag>

## Why Java?

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

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

- **Caveats.**



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

# 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

Built-In Types		System		Math Library	
int	double	System.out.println()		Math.sin()	Math.cos()
long	String	System.out.print()		Math.log()	Math.exp()
char	boolean	System.out.printf()		Math.sqrt()	Math.pow()
				Math.min()	Math.max()
				Math.abs()	Math.PI

Flow Control		Parsing	
if	else	Integer.parseInt()	
for	while	Double.parseDouble()	

Boolean		Punctuation	
true	false	{	}
	&&	(	)
!		,	;

Primitive Numeric Types		
+	-	*
/	%	++
--	>	<
<=	>=	==
!=		

String		Arrays	Objects	
+	""	a[i]	class	static
length()	compareTo()	new	public	private
charAt()	matches()	a.length	toString()	equals()
			new	main()

# Programming in Java

## ● Programming in Java.

- **Create** the program by typing it into a text editor, and save it as `HelloWorld.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`

# Programming in Java

## ● Programming in Java.

- Create the program by typing it into a text editor, and save it as `HelloWorld.java`
- **Compile** it using Eclipse or by typing at the command-line:  
`javac HelloWorld.java`

command-line →

```
javac HelloWorld.java
```

(or click the Save button in Eclipse)

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

# Java Bytecode

HelloWorld.class

```
0000000 312 376 272 276 \0 \0 \0 . \0 035 \n \0 006 \0 017 \t
0000020 \0 020 \0 021 \b \0 022 \n \0 023 \0 024 007 \0 025 007
0000040 \0 026 001 \0 006 < i n i t > 001 \0 003 ( )
0000060 v 001 \0 004 C o d e 001 \0 017 L i n e N
0000100 u m b e r T a b l e 001 \0 004 m a i
0000120 n 001 \0 026 ( [ L j a v a / l a n g
0000140 / S t r i n g ; ) v 001 \0 \n S o u
0000160 r c e F i l e 001 \0 017 H e l l o W
0000200 o r l d . j a v a \f \0 007 \0 \b 007 \0
0000220 027 \f \0 030 \0 031 001 \0 \f H e l l o ,
0000240 w o r l d 007 \0 032 \f \0 033 \0 034 001 \0 \n
0000260 H e l l o W o r l d 001 \0 020 j a v
0000300 a / l a n g / O b j e c t 001 \0 020
0000320 j a v a / l a n g / S y s t e m
0000340 001 \0 003 o u t 001 \0 025 L j a v a / i
0000360 o / P r i n t S t r e a m ; 001 \0
0000400 023 j a v a / i o / P r i n t S t
0000420 r e a m 001 \0 007 p r i n t l n 001 \0
0000440 025 ( L j a v a / l a n g / S t r
0000460 i n g ; ) v \0 ! \0 005 \0 006 \0 \0 \0 \0
0000500 \0 002 \0 001 \0 007 \0 \b \0 001 \0 \t \0 \0 \0 035
0000520 \0 001 \0 001 \0 \0 \0 005 * 267 \0 001 261 \0 \0 \0
0000540 001 \0 \n \0 \0 \0 006 \0 001 \0 \0 \0 \f \0 \t \0
0000560 013 \0 \f \0 001 \0 \t \0 \0 \0 % \0 002 \0 001 \0
0000600 \0 \0 \t 262 \0 002 022 003 266 \0 004 261 \0 \0 \0 001
0000620 \0 \n \0 \0 \0 \n \0 002 \0 \0 \0 017 \0 \b \0 020
0000640 \0 001 \0 \r \0 \0 \0 002 \0 016
0000652
```

## 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 ( $\sim 10^7$  bytes) file
- Print the words together with counts of how often they occur
- Need more specification?
  
- How do you do it?
  
  
- What if we only wanted the top  $k$  (say 20) words?

## Possible solutions

1. Use heavy duty data structures (Knuth) →

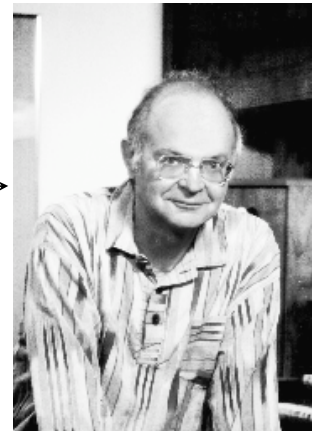
- Hash tries implementation
- Randomized placement
- Lots o' pointers
- Several pages

2. UNIX shell script (Doug McIlroy)

```
tr -cs "[:alpha:]" "\\n*" < FILE | \  
sort | \  
uniq -c | \  
sort -n -r -k 1,1
```

3. See `SimpleWordCount.java`

- Which is better?
  - K.I.S.?

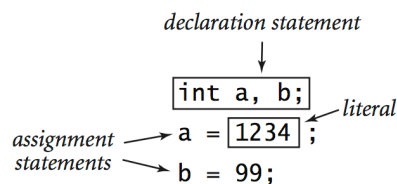


# Problem Solving and Programming

- 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.

	a	b	t
int a, b;	undefined	undefined	
a = 1234;	1234	undefined	
b = 99;	1234	99	
int t = a;	1234	99	1234
a = b;	99	99	1234
b = t;	99	1234	1234

## 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 to do both

# Built-In Data Types

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

Type	Description	Literals	Operations
char	characters	'A' '@'	compare
String	sequences of characters	"Hello World" "CS is fun"	concatenate
int	integers	17 12345	add, subtract, multiply, divide
double	floating point numbers	3.1415 6.022e23	add, subtract, multiply, divide
boolean	truth values	true false	and, or, not

# Integers

$-2^{31}$                        $2^{31} - 1$   
 ↓                                      ↓

- **Integers:** the `int` data type.
  - **Values:** integers between  $-2147483648$  and  $2147483647$ .
  - **Operations:** add, subtract, multiply, divide, remainder.
  - **Useful for expressing algorithms.**

expression	value	comment
$5 + 3$	8	
$5 - 3$	2	
$5 * 3$	15	
$5 / 3$	1	
$5 \% 3$	2	
$1 / 0$		runtime error
$3 * 5 - 2$	13	* has precedence
$3 + 5 / 2$	5	/ has precedence
$3 - 5 - 2$	-4	left associative
$(3 - 5) - 2$	-4	better style
$3 - (5 - 2)$	0	unambiguous

*Typical int expressions*

## Integer Operations

```
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);
    }
}
```

command-line  
arguments

```
% javac IntOps.java
% java IntOps 1234 99
1234 + 99 = 1333
1234 * 99 = 122166
1234 / 99 = 12
1234 % 99 = 46
```

Java automatically converts  
a, b, and rem to type String

## 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.

```
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);
    }
}
```

```
% java LeapYear 2004
true
% java LeapYear 1900
false
% java LeapYear 2000
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.

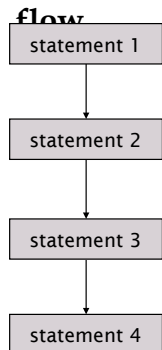
```
public class RandomInt {
    public static void main(String[] args) {
        int N = Integer.parseInt(args[0]);
        double r = Math.random();
        int n = (int) (r * N);
        System.out.println("random integer is " + n);
    }
}
```

String to int (method)  
double between 0.0 and 1.0  
double to int (cast)    int to double (automatic)  
int to String (automatic)

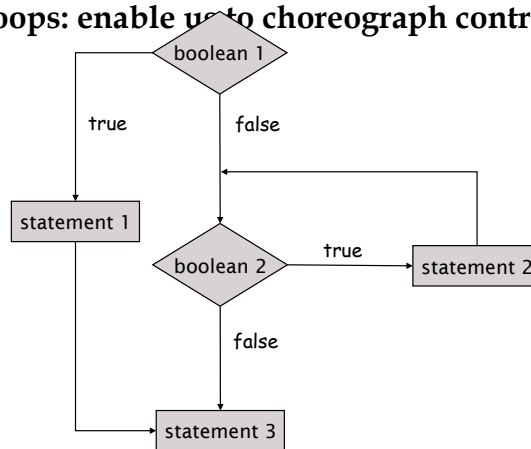
```
% java RandomInt 6
random integer is 3
% java RandomInt 6
random integer is 0
% java RandomInt 10000
random integer is 3184
```

# Control Flow

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



straight-line control flow



control flow with conditionals and loops

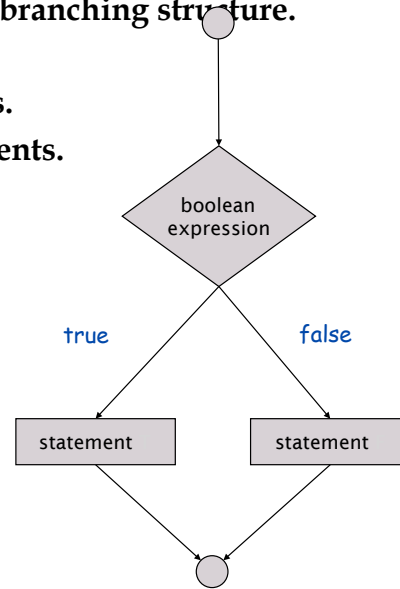
## If-Else Statement

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

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

can be any sequence  
of statements

if-else syntax



if-else flow chart

## 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".

```
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*

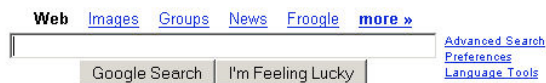
- Computer science is not as old as physics; it lags by a couple of hundred years. However, this does not mean that there is significantly less on the computer scientist's plate than on the physicist's: younger it may be, but it has had a far more intense upbringing!



*Richard Feynman*

<http://www.wordiq.com>

# Computer Science in a Nutshell?



[Advertising Programs](#) - [Business Solutions](#) - [About Google](#)

©2004 Google - Searching 8,068,044,851 web pages