Introduction to Computer Science

Introduction

Ryan Stansifer

Florida Institute of Technology
Melbourne, Florida USA  32901

http://www.cs.fit.edu/~ryan/

11 April 2021
Overview of Course

- Introduction and Context. What is CS?
- Java review. Data, control constructs, static methods
- Classes. Incorporation, instantiation, inheritance
- Generics. Code reuse
- Program analysis. Steps the program takes
- Data structures. Lists, stacks, queue
Course Goals

• Programming
  • exciting to translate ideas into reality
  • basics are simple, yet programming well is difficult; do not underestimate the challenge
  • delivery high-quality programs on time; be able to express control flow and design data in Java
  • problem solving is hard and difficult to teach

• Computer Science
  • Computer Science is not just programming
  • It is easy to lose sight of the big picture, so we have a general introduction
  • Other (non-programming) topics from time to time: architecture, Monte Carlo methods, $O(N)$, invariants, and so on
Outline of Introduction

There are couple of topics that put programming in context and that are helpful if pointed out in advance and getting mired in the details.

- What is Computer Science? Areas of study: AI, OS, ...
- What is a computer? Architecture, CPU, memory hierarchy
- Interface layers: hardware, operating system, application
- The Java platform
  - JVM and a million other pieces
  - Java history, pragmatics
- Programming languages — not just Java
- Program development; debuggers and so on
- Program style. A program is a text file
- I/O, streams

The single most important skill in programming, computer science, and science in general is abstraction. Yet I think that belaboring the idea may be too philosophical at this time. If one is observant, one will see abstraction at work in all the topics above.
What is Computer Science?

Computer Science is the study of information, protocols and algorithms for idealized and real automata.

- **Automaton:** "self moving" – in our context, self "deciding" or autonomous mechanism with bounded resources (time and space).
- **Information:** knowledge represented in a form suitable for transmission, manipulation, etc.
- **Protocol:** rules for exchanging information without problems.
- **Algorithm:** an unambiguous, finite description in simple steps or actions.

Computer Science is not the study of computers, nor is it the practice of their use.
Computer science. *The study of information, protocols and algorithms for idealized and real automata.*
What is Computer Science?

**computer science.** *The study of information, protocols and algorithms for idealized and real automata.*

- automaton:
What is Computer Science?

**computer science.** *The study of information, protocols and algorithms for idealized and real automata.*

- automaton: “self moving” – in our context, self “deciding” or autonomous mechanism with bounded resources (time and space)
What is Computer Science?

**computer science.** The study of information, protocols and algorithms for idealized and real automata.

- automatons: “self moving” – in our context, self “deciding” or autonomous mechanism with bounded resources (time and space)
- information:
What is Computer Science?

computer science. The study of information, protocols and algorithms for idealized and real automata.

- automaton: “self moving” – in our context, self “deciding” or autonomous mechanism with bounded resources (time and space)
- information: knowledge represented in a form suitable for transmission, manipulation, etc.
What is Computer Science?

**computer science.** *The study of information, protocols and algorithms for idealized and real automata.*

- **automaton:** “self moving” – in our context, self “deciding” or autonomous mechanism with bounded resources (time and space)
- **information:** knowledge represented in a form suitable for transmission, manipulation, etc.
- **protocol:**
What is Computer Science?

**computer science.** *The study of information, protocols and algorithms for idealized and real automata.*

- **automaton:** “self moving” – in our context, self “deciding” or autonomous mechanism with bounded resources (time and space)
- **information:** knowledge represented in a form suitable for transmission, manipulation, etc.
- **protocol:** rules for exchanging information without problems
computer science. The study of information, protocols and algorithms for idealized and real automata.

- automaton: “self moving” – in our context, self “deciding” or autonomous mechanism with bounded resources (time and space)
- information: knowledge represented in a form suitable for transmission, manipulation, etc.
- protocol: rules for exchanging information without problems
- algorithm:
What is Computer Science?

computer science. The study of information, protocols and algorithms for idealized and real automata.

- automaton: “self moving” – in our context, self “deciding” or autonomous mechanism with bounded resources (time and space)
- information: knowledge represented in a form suitable for transmission, manipulation, etc.
- protocol: rules for exchanging information without problems
- algorithm: an unambiguous, finite description in simple steps or actions
What is Computer Science?

**computer science.** *The study of information, protocols and algorithms for idealized and real automata.*

- **automaton:** “self moving” – in our context, self “deciding” or autonomous mechanism with bounded resources (time and space)
- **information:** knowledge represented in a form suitable for transmission, manipulation, etc.
- **protocol:** rules for exchanging information without problems
- **algorithm:** an unambiguous, finite description in simple steps or actions

Computer Science is not the study of computers, nor is it the practice of their use.
• Arabic: خوارزمية
• Chinese (simplified): 算法
• Dutch: algoritme
• Finnish: algoritmi
• French: algorithme
• German: Algorithmus
• Georgian: ალგორითმი
• Hindi: कलन विधि
• Icelandic: reiknirit
• Japanese: アルゴリズム
• Latin: algorithmus
• Spanish: algoritmo
• Swedish: algoritm
• Turkish: algoritma
How does this class (studying Java) fit into the study of Computer Science?
How does this class (studying Java) fit into the study of Computer Science?

Learn some algorithms, some real and idealized machines, learn something about information. Mostly learn some mechanisms which can express computation.
Mathematics, science, or engineering?

**Mathematics.** *The science of numbers, interrelations, and abstractions.*

**Science.** *Systematic knowledge or practice.* Acquiring knowledge through the scientific method of natural phenomena (natural sciences) or human or social behavior (social sciences).

**Engineering.** *The applied science of acquiring and applying knowledge to design, or construct works for practical purposes.*
What is CS?

- Engineering? Application of science?
- Natural science? Observable phenomena?
- Mathematics? Invisible abstractions?
- Social science? Functioning of human society?
What is CS?

- Engineering? Application of science?
- Natural science? Observable phenomena?
- Mathematics? Invisible abstractions?
- Social science? Functioning of human society?

CS is exciting and difficult as it is all these things.
What is CS?

- To do anything requires programming
- To do something useful requires domain knowledge

Do you become skillful at programming, or an expert in some small area?
Existential Angst

*The Scream* by the Norwegian artist Edvard Munch, painted in 1893.
We are at the dawn of new era. The, as yet unfinished, language of computation is the language of science and engineering and is overtaking mathematics as the Queen of Science.

*Philosophy is written in this grand book, the universe which stands continually open to our gaze. But the book cannot be understood unless one first learns to comprehend the language and read the letters in which it is composed. It is written in the language of mathematics, and its characters are triangles, circles and other geometric figures without which it is humanly impossible to understand a single word of it; without these, one wanders about in a dark labyrinth.*

Galileo Galilee in *The Assayer*
What Does A Computer Scientist Do?

Similar to mathematics, most everyone in modern society uses computing. So getting a computer science degree prepares you for everything and nothing.

The most visible activity is commanding computers to do our bidding, i.e., programming.

What do you want to do?
Fields

- Computer architecture
- Operating systems
- Programming languages and compilers
- Algorithms, data structures, complexity
- Computability theory
- Numerical analysis
- Networking and distributed computing
- Parallel computing
- Information Management/Database systems
- Software development (aka Software Engineering)
- Human-computer communication/interaction
- Graphics and Visual Computing
- Intelligent Systems (aka Artificial Intelligence)
Basic five-stage pipeline in a RISC machine: instruction fetch, instruction decode, execute, memory access, register write back.

The IBM PowerPC G5 has 21 pipeline stages; the Intel Pentium 4E has 31 stages.
Operating Systems — paging

[Diagram showing paging concept with CPU, logical address, page table, physical address, and memory blocks.]

physical memory

physical address

f0000 … 0000

f1111 … 1111
Programming Languages and Compilers

[Front end of compiler]
Algorithms and Data Structures — Sorting

Sorting animation
The argument that the power of mechanical computations is limited is not surprising. Intuitively we know that many vague and speculative questions require special insight and reasoning well beyond the capacity of any computer that we can now construct or even foresee. What is more interesting to computer scientists is that there are questions than can be clearly and simple stated, with an apparent possibility of an algorithmic solution, but which are know to be unsolvable by any computer.

Linz 6th, Section 12.1, page 310
An example of such a problem is if a grammar is ambiguous or not. (This can formalized and is an interesting issue in constructing compilers.)
The proof that there are specific problems that cannot be solved, if not remarkably simple.
Theory of Computation — halting problem

Start

Will this program halt?

yes

no

Halt
Numerical Analysis

A report from the United States General Accounting Office begins “On February 25, 1991, a Patriot missile defense system operating at Dhahran, Saudi Arabia, during Operation Desert Storm failed to track and intercept an incoming Scud. This Scud subsequently hit an Army barracks, killing 28 Americans.” The report finds the failure to track the Scud missile was caused by a precision problem in the software.


http://www.ima.umn.edu/~arnold/disasters/disaster.html
In the sliding window protocol the window size is the amount of data a sender is allowed to have sent into the network without having yet received an acknowledgment for it.

In Internet routers, active queue management (AQM) is a technique that consists in dropping packets before a router’s queue is full. Historically, queues use a *drop-tail* discipline: a packet is put onto the queue if the queue is shorter than its maximum size. Drop-tails queue have a tendency to penalize bursty flows.

Active queue disciplines drop packets before the queue is full based on probabilities. Active queue disciplines are able to maintain a shorter queue length than the drop-tail queue which reduces network latency.
Distributed Computing — barber shop problem
Parallel Computing

sing inst  mult inst

single data  SISD  MISD
multiple data  SIMD  MIMD

Flynn’s taxonomy

MIMD Instruction Pool

Data Pool

PU  PU  PU  PU
PU  PU  PU  PU
PU  PU  PU  PU
PU  PU  PU  PU
The join of two relational tables

<table>
<thead>
<tr>
<th>Name</th>
<th>Have</th>
<th>Result of Join</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td>Dot</td>
<td>cat</td>
<td>Dot</td>
</tr>
<tr>
<td>Sue</td>
<td>dog</td>
<td>Sue</td>
</tr>
<tr>
<td>Zan</td>
<td>cat</td>
<td>Zan</td>
</tr>
<tr>
<td>Bip</td>
<td>bird</td>
<td>Bip</td>
</tr>
<tr>
<td>A</td>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td>Dot</td>
<td>cat</td>
<td>shots</td>
</tr>
<tr>
<td>Sue</td>
<td>dog</td>
<td>shots</td>
</tr>
<tr>
<td>Zan</td>
<td>cat</td>
<td>shots</td>
</tr>
</tbody>
</table>

The join of two relational tables
Software Engineering — waterfall model

1. Requirements definition
2. System and software design
3. Implementation and unit testing
4. Integration and system testing
5. Operation and maintenance
Human-Computer Communication/Interaction

The Miracle Worker scene from Star Trek 4: The Voyage Home on YouTube.
Frozen Fire

- 37 hours to render
- POV ray uses a C-like programming language
Intelligent Systems

C3PO and R2D2 are fantasy robots from the movie Star Wars, while Kiva’s industrial robots can efficiently and intelligently move shelves in a warehouse.

Computer Science (Introduction)
What is CS? (Overview of Fields)
Computer Science (Introduction)

What is CS? (Overview of Fields)
End of the overview of different fields of study in computer science
Layers, Scale, Interfaces
The solution to vast complexity is layers. Good layers enable high-quality specialization, bad layers just increase the complexity.

Software development—programming—is often about creating layers, even in small programming projects.
Computing is complex. There are many layers of interesting stuff between the person and the automaton.
Scale of the Universe 2
by Cary and Michael Huang
The vastness and minuteness of time and space is a challenge to comprehend.

From Wikipedia, the free encyclopedia

<table>
<thead>
<tr>
<th>Section</th>
<th>Range (m)</th>
<th>Unit</th>
<th>Example Items</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>≥ 10⁻¹⁵</td>
<td>am</td>
<td>electron, quark, string</td>
</tr>
<tr>
<td>Subatomic</td>
<td>10⁻¹² - 10⁻⁹</td>
<td>nm</td>
<td>proton, neutron, wavelength of gamma rays and X-rays, hydrogen atom</td>
</tr>
<tr>
<td></td>
<td>10⁻⁹ - 10⁻⁶</td>
<td>pm</td>
<td>DNA helix, virus, wavelength of optical spectrum</td>
</tr>
<tr>
<td>Atomic to Cellular</td>
<td>10⁻³ - 10⁻⁰</td>
<td>µm</td>
<td>bacterium, fog water droplet, human hair[1]</td>
</tr>
<tr>
<td></td>
<td>10⁻² - 10⁻¹</td>
<td>mm</td>
<td>mosquito, golf ball, soccer ball</td>
</tr>
<tr>
<td></td>
<td>10⁻¹ - 10⁰</td>
<td>m</td>
<td>human being, American football field, Eiffel Tower</td>
</tr>
<tr>
<td>Human Scale</td>
<td>10⁰ - 10⁶</td>
<td>km</td>
<td>Mount Everest, length of Panama Canal, asteroid</td>
</tr>
<tr>
<td></td>
<td>10⁶ - 10⁸</td>
<td>Mm</td>
<td>the Moon, Earth, one light-second</td>
</tr>
<tr>
<td>Astronomical</td>
<td>10⁹ - 10¹²</td>
<td>Gm</td>
<td>Sun, one light-minute, Earth's orbit</td>
</tr>
<tr>
<td></td>
<td>10¹² - 10¹⁵</td>
<td>Tm</td>
<td>orbits of outer planets, Solar System,</td>
</tr>
<tr>
<td></td>
<td>10¹⁵ - 10¹⁸</td>
<td>Pm</td>
<td>one light-year, distance to Proxima Centaur</td>
</tr>
<tr>
<td></td>
<td>10¹⁸ - 10²¹</td>
<td>Em</td>
<td>galactic arm</td>
</tr>
<tr>
<td></td>
<td>10²¹ - 10²⁴</td>
<td>Zm</td>
<td>Milky Way, distance to Andromeda Galaxy</td>
</tr>
<tr>
<td></td>
<td>10²⁴ - ∞</td>
<td>Ym</td>
<td>visible universe</td>
</tr>
</tbody>
</table>
A study of people in Nicaragua who were born deaf and never learned Spanish or a formal sign language has concluded that humans need language in order to understand large numbers. “Up to three, they’re fine,” says Elizabet Spaepen, a researcher at the University of Chicago and an author of the study. “But past three, they start to fall apart.”

http://www.npr.org/2011/02/09/
<table>
<thead>
<tr>
<th>Prefix</th>
<th>Symbol</th>
<th>Conversion</th>
<th>Value</th>
<th>Number Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>peta</td>
<td>P</td>
<td>quadrillion</td>
<td>$10^{15}$</td>
<td>1 000 000 000 000 000</td>
</tr>
<tr>
<td>tera</td>
<td>T</td>
<td>trillion</td>
<td>$10^{12}$</td>
<td>1 000 000 000 000</td>
</tr>
<tr>
<td>giga</td>
<td>G</td>
<td>billion</td>
<td>$10^{9}$</td>
<td>1 000 000 000</td>
</tr>
<tr>
<td>mega</td>
<td>M</td>
<td>million</td>
<td>$10^{6}$</td>
<td>1 000 000</td>
</tr>
<tr>
<td>kilo</td>
<td>k</td>
<td>thousand</td>
<td>$10^{3}$</td>
<td>1 000</td>
</tr>
<tr>
<td>hecto</td>
<td>h</td>
<td>hundred</td>
<td>$10^{2}$</td>
<td>100</td>
</tr>
<tr>
<td>deca</td>
<td>da</td>
<td>ten</td>
<td>$10^{1}$</td>
<td>10</td>
</tr>
<tr>
<td>(none)</td>
<td></td>
<td>one</td>
<td>$10^{0}$</td>
<td>1</td>
</tr>
<tr>
<td>deci</td>
<td>d</td>
<td>tenth</td>
<td>$10^{-1}$</td>
<td>0.1</td>
</tr>
<tr>
<td>centi</td>
<td>c</td>
<td>hundredth</td>
<td>$10^{-2}$</td>
<td>0.01</td>
</tr>
<tr>
<td>milli</td>
<td>m</td>
<td>thousandth</td>
<td>$10^{-3}$</td>
<td>0.001</td>
</tr>
<tr>
<td>micro</td>
<td>µ</td>
<td>millionth</td>
<td>$10^{-6}$</td>
<td>0.000 001</td>
</tr>
<tr>
<td>nano</td>
<td>n</td>
<td>billionth</td>
<td>$10^{-9}$</td>
<td>0.000 000 001</td>
</tr>
<tr>
<td>pico</td>
<td>p</td>
<td>trillionth</td>
<td>$10^{-12}$</td>
<td>0.000 000 000 001</td>
</tr>
<tr>
<td>femto</td>
<td>f</td>
<td>quadrillionth</td>
<td>$10^{-15}$</td>
<td>0.000 000 000 000 001</td>
</tr>
</tbody>
</table>
**SCIENCE MNEMONICS**

**ORDER OF OPERATIONS**
parentheses, exponents, division, multiplication, addition, subtraction

**TAXONOMY**
kings, phyla, class, order, family, genus, species

**SI PREFIXES**
Kilo, mega, giga, tera, peta, exa, zetta, yotta

**GEOLOGIC PERIODS**
Paleozoic, Mesozoic, Cenozoic

**PLANETS**
Mercury, Venus, Earth, Mars, Jupiter, Saturn, Uranus, Neptune

**RESISTOR COLOR CODES**
black, brown, red, orange, yellow, green, blue, violet, gray, white

**HERE'S MY DAD A SPARK**
Oh, brainy, expect more brains and sex

**I'M NOT SURE WHY OTHERS THINK I'M AN IDIOT**
On the contrary, Grady-Olly, apply same system

**BIG BROTHE TO TOPOLANS OVERLORDS,**
Yelled Glenn, "BRING YOURS VIA GROUND WATER!!"

**Big Memes Give The Prez A Raw Like Cowboy**
Microsoft made no profit from anyone's zones

**KINGDOM, PHYLUM, CLASS, ORDER, FAMILY, GENUS, SPECIES**
In 1978, funky decks were sold for God's glory

**PLANET**
Mercury, Venus, Earth, Mars, Jupiter, Saturn, Uranus, Neptune

**ORDER OF OPERATIONS**
parentheses, exponents, division, multiplication, addition, subtraction
A Memory Aid for the SI Prefixes

[deca, hecto], kilo, mega, giga, tera, peta, exa, zetta
Karl Marx gave the proletariat eleven zeppelins.

[deci, centi], milli, micro, nano, pico, femto, atto, zepto
Microsoft made no profit from anyone’s zunes.

Planets:
My very excellent mother just served us nachos.
Mary’s “Virgin” explanation made Joseph suspect upstairs neighbor.
Man very early made jars serve useful needs [period].
A computer is a remarkable tool and easily works at all scale. Now some early history of technology and scientific computing...

The MANIAC (Mathematical Analyzer Numerical Integrator And Computer Model I) was an early computer built at the Los Alamos Scientific Laboratory. It ran from 1952–1958.

- Klára Dán von Neumann - wrote the first programs for MANIAC.
- Dana Scott - programmed the MANIAC to enumerate all solutions to a pentomino puzzle by backtracking in 1958.
By mid-1953, five distinct sets of problems were running on the MANIAC, characterized by different scale of time: (1) nuclear explosions, over in microseconds; (2) shock and blast waves, ranging from microseconds to minutes; (3) meteorology, ranging from minutes to years; (4) biological evolution, ranging from years to millions of years; and (5) stellar evolution, ranging from millions to billions of years. All this in 5 kilobytes—enough memory for about one-half second of audio, at the rate we now compress music into MP3s.

Powers of Two

Because computers represent information in binary form, it is important to know how many pieces of information can be represented in \( n \) (binary) bits. \( 2^n \) pieces of information can be stored in \( n \) bits, and so is it necessary to be familiar with powers of two.

It is obvious that \( \lceil \log_2 n \rceil \) bits are required to represent \( n \) things. Some bit patterns might be unused.

\[
\begin{array}{c|c}
2^n & \text{value} \\
\hline
2^0 & 1 \\
2^1 & 2 \\
2^2 & 4 \\
2^3 & 8 \\
2^4 & 16 \\
2^5 & 32 \\
\end{array}
\]

(What does it mean that one thing can be represented with zero bits? There is no need to represent one thing as there is nothing else which can be confused with it.)
In Java, the expression

$$32 - \text{Math.numberOfLeadingZeros}(n-1)$$

will compute the number of bits necessary to represent $n$ things.

The number of bits necessary to represent a natural number $n$ is a closely related, but different notion. $\lfloor \log_2 n \rfloor + 1$ is not the same thing as $\lceil \log_2 n \rceil$.

<table>
<thead>
<tr>
<th>$n$</th>
<th>$\log_2 n$</th>
<th>$\lfloor \log_2 n \rfloor$</th>
<th>$\lceil \log_2 n \rceil$ + 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.0</td>
<td>0.0</td>
<td>1.0</td>
</tr>
<tr>
<td>2</td>
<td>1.0</td>
<td>1.0</td>
<td>2.0</td>
</tr>
<tr>
<td>3</td>
<td>1.6</td>
<td>2.0</td>
<td>2.0</td>
</tr>
<tr>
<td>4</td>
<td>2.0</td>
<td>2.0</td>
<td>3.0</td>
</tr>
<tr>
<td>5</td>
<td>2.3</td>
<td>3.0</td>
<td>3.0</td>
</tr>
</tbody>
</table>

See jgloss.
Because information is represented in ones and twos, doubling (exponential growth) is an important concept in computing.
To grasp it better, we use an ancient Indian chess legend.

A picture of Krishna playing Chess from the National Museum, New Delhi
Legend of the Ambalappuzha Paal Payasam

According to the legend, Lord Krishna once appeared in the form of a sage in the court of the king who ruled the region and challenged him for a game of chess (or chaturanga).

The sage told the king that he would play for grains of rice—one grain of rice in the first square, two grains in the second square, four in the third square, eight in the fourth square, and so on. Every square would have double the number of grains of its predecessor. The king lost the game and soon realized that even if he provided all the rice in his kingdom, he would never be able to fulfill the promised reward. The sage appeared to the king in his true form, that of Lord Krishna, and said he could serve paal-payasam (sweet pudding made of milk and rice) in the temple freely to the pilgrims every day until the debt was paid off.
How many squares in a checkerboard? How big is $2^{64}$?

- $10^3$ kilo $k$
- $10^6$ mega $M$
- $10^9$ giga $G$
- $10^{12}$ tera $T$
- $10^{15}$ peta $P$
- $10^{18}$ exa $E$
Maybe we can comprehend it better as weight or time. How much does a gain of rice weigh? One grain of rice weighs about 0.029 grams. And, approximately 470 million tons of milled rice is harvested annually in the world today.
<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>0.029g</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>0.058g</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>0.116g</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>0.232g</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>0.464g</td>
</tr>
<tr>
<td>5</td>
<td>32</td>
<td>0.928g</td>
</tr>
<tr>
<td>6</td>
<td>64</td>
<td>1.856g</td>
</tr>
<tr>
<td>7</td>
<td>128</td>
<td>3.712g</td>
</tr>
<tr>
<td>8</td>
<td>256</td>
<td>7.424g</td>
</tr>
<tr>
<td>9</td>
<td>512</td>
<td>14.848g</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>10</td>
<td>1 024</td>
<td>29.696g</td>
</tr>
<tr>
<td>11</td>
<td>2 048</td>
<td>59.392g</td>
</tr>
<tr>
<td>12</td>
<td>4 096</td>
<td>118.784g</td>
</tr>
<tr>
<td>13</td>
<td>8 192</td>
<td>237.568g</td>
</tr>
<tr>
<td>14</td>
<td>16 384</td>
<td>475.136g</td>
</tr>
</tbody>
</table>
|   |   |   |...
|25 |33 554 432 |973.079kg |about a ton |
|   |   |   |...
|40 |1 099 511 627 776 |35 thousand tons |
|50 |1 125 899 906 842 624 |36 million tons |
|60 |1 152 921 504 606 846 976 |36 billion tons |
|64 |18,446,744,073,709,551,616 |589 billion tons |

$2^{64}$ grains of rice is the total rice harvest for over 1200 years.
Powers of Two (Time in Seconds)

Suppose we double 1 second and double the amount of time again. And, we do this again and again.

0  1   1 second
1  2   2 seconds
2  4   4 seconds
3  8   8 seconds
4 16   16 seconds
5 32   32 seconds
6 64   about a minute
7 128  about 2 minutes
8 256  about 4 minutes
9 512  about 8 minutes
Powers of Two

10  1 024  17 minutes
...  ...
19  524 288  one week
20  1 048 576  two weeks
...  ...
25  33 554 432  a year
...  ...
30  1 073 741 824  34 years
40  1 099 511 627 776  37 millennia
50  125 899 906 842 624
60  1 152 921 504 606 846 976  age of universe
Powers of Two

Notice that $2^{10} \approx 10^3$, so these powers have significance:

<table>
<thead>
<tr>
<th>Power</th>
<th>Value</th>
<th>Approximation</th>
<th>Prefix</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>$1024$</td>
<td>$10^3$</td>
<td>kilo</td>
</tr>
<tr>
<td>20</td>
<td>$1048576$</td>
<td>$10^6$</td>
<td>mega</td>
</tr>
<tr>
<td>30</td>
<td>$1073741824$</td>
<td>$10^9$</td>
<td>giga</td>
</tr>
<tr>
<td>40</td>
<td>$1099511627776$</td>
<td>$10^{12}$</td>
<td>tera</td>
</tr>
<tr>
<td>50</td>
<td>$1125899906842624$</td>
<td>$10^{15}$</td>
<td>peta</td>
</tr>
<tr>
<td>60</td>
<td>$1152921504606846976$</td>
<td>$10^{18}$</td>
<td>exa</td>
</tr>
<tr>
<td>70</td>
<td>$1180591620717411303424$</td>
<td>$10^{21}$</td>
<td>zetta</td>
</tr>
</tbody>
</table>

SI prefixes are not supposed to be used for powers of 2 (just powers of 10). Sadly, abuse of SI prefixes in computer technology has led to confusion. Whereas 1GHz usually means $10^9$ instructions per second, 1GB usually means $2^9$ bytes.
Some other powers have special significance in computing.

<table>
<thead>
<tr>
<th>exponent</th>
<th>bit patterns</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>128</td>
<td>size of ASCII char set</td>
</tr>
<tr>
<td>8</td>
<td>256</td>
<td>size of Latin-1 char set</td>
</tr>
<tr>
<td>16</td>
<td>65 536</td>
<td>size of Java <strong>short</strong></td>
</tr>
<tr>
<td>31</td>
<td>2 147 483 648</td>
<td>no. of neg <strong>int</strong></td>
</tr>
<tr>
<td>32</td>
<td>4 294 967 296</td>
<td>size of Java <strong>int</strong></td>
</tr>
<tr>
<td>63</td>
<td>9 233 372 036 854 775 808</td>
<td>no. of neg <strong>long</strong></td>
</tr>
<tr>
<td>64</td>
<td>18 446 744 073 709 551 616</td>
<td>size of Java <strong>long</strong></td>
</tr>
</tbody>
</table>
One challenge in computer science is the vast scale of computing devices. A computer may have a terabyte ($10^{12}$ bytes) worth of storage. A computer may execute ten instructions every nanosecond ($10^{-9}$ seconds).

How does one deal with such complexity?
Layers
Car dashboard in an interface.
The details of the engine are immaterial to the user (but not to the mechanic).
Interface Layers

User

Application

Operating System

Hardware
Computing is complex. There are many layers of interesting stuff between the person and the automaton.

- person (user)
- user-interface (mouse, etc)
- application (program)
- high-level programming language
- machine language
- operating system (OS)
- hardware
- logical devices
- physics
Definitions

- **interface** — An *interface* defines the communication boundary between two entities, such as a piece of software, a hardware device, or a user. It generally refers to an abstraction that an entity provides of itself to the outside.

- **API** — An *application programming interface (API)* is a set of procedures that an operating system, library, or service provides to support requests made by computer programs. For example, the Java API.

- **IDE** — In computing, an *Integrated development environment (IDE)* is a software application that provides facilities to computer programmers of a source code editor, a compiler and/or interpreter, build automation tools, and usually a debugger. [The definition of IDE does not really belong here.]
The program controls the computer, yet it needs critical assistance (from the operating system) to communicate with the outside environment and even to run effectively.
Can a Java program print or display a ß (es-zet) character?
Can a Java program draw a red line on the display?
Can a Java program determined which button on the display was clicked or touched?

No, not really; yet we write Java programs to these things all the time. In reality the capabilities of the system are determined by the hardware and managed by the operating system. Nothing can happen without the support of the operating system. Java APIs abstract external communication. How easy these API's are to use is determined by the skill of the software designers.
Can a Java program print or display a ß (es-zet) character?
Can a Java program draw a red line on the display?
Can a Java program determine which button on the display was clicked or touched?
No, not really; yet we write Java programs to these things all the time. In reality the capabilities of the system are determined by the hardware and managed by the operating system.
Nothing can happen without the support of the operating system.
Java APIs abstract external communication. How easy these API’s are to use is determined by the skill of the software designers.
For a deeper appreciation of programming a computer, we should examine briefly the many layers upon which the user depends.

An important lesson in organizing these complex systems is that the boundaries should be well chosen. Rapidly changing technology, competing business interests, and new insights make it impossible to settle these boundaries once and for all. Whole college classes like computer architecture, operating systems, compiler construction, and programming languages go into the subjects more deeply.
Hardware and Operating System Platform

Application

System calls: open(), read(), mkdir(), kill()

OS:

File system
Memory management

Process management
Networking

Hardware:

CPU
Memory
Network interface

Monitor
Disk
Keyboard
Example Platforms

- Hardware: IBM PowerPC, Intel x86, Sun Ultra-SPARC II
- OS: Microsoft Windows XP, Mac OS X v10.5 “Leopard”, Linux, Solaris 10

Try:

cs> uname -io
RackMac3, 1 Darwin

olin> uname -io
X86_64 GNU/Linux

The Java programming language (and other high-level languages) try to form a high-level platform.
Good interfaces mean you don’t have to understand the lower layers. For example, you don’t have to understand electronic flip-flops in order to program. The point is:

- Many interfaces are software constructions, and software interfaces are an important design problem for programmers
- Many existing interfaces are in flux requiring an understanding of the lower layers.
Computer Hardware
Computer Hardware

Intel quad
Computer Architecture—CPU

- control
- clock
- instruction register
- program counter
- incr
- MAR
- MDR
- ALU
- condition codes
- registers
- bus
control unit is the part of the cpu that controls all the internal actions of the cpu, especially the fetch/execute cycle.

arithmetic/logic unit (ALU) is the part of the cpu that does operations: addition, multiplication, etc.

memory data register (MDR) is the register of the cpu that contains the data to be stored in the computer’s main storage, or the data after a fetch from the storage. It acts like a buffer keeping the contents of storage ready for immediate use by the cpu.
Computer Memory Hierarchy

- **Processor Registers**: Small size, small capacity. Very fast, very expensive.
- **Processor Cache**: Medium size, medium capacity. Power on immediate term. Very fast, expensive.
- **Flash/USB Memory**: Large size, very large capacity. Power off short term. Slower, cheap.
- **Hard Drives**: Power off mid term. Hard drives slow, very cheap.
- **Tape Backup**: Power off long term. Tape backup very slow, affordable.
Unfortunately, registers are always comparatively few in number, since they are among the most expensive resources in most implementations, both because of the read and interconnection complexity they require and because the number of registers affects the structure of instructions and the space available in instructions for specifying opcodes, offsets, conditions, and so on.

Muchnick, page 110
### Memory Hierarchy

<table>
<thead>
<tr>
<th>type</th>
<th>access</th>
<th>size</th>
<th>cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>registers</td>
<td>5ns</td>
<td>1e2</td>
<td></td>
</tr>
<tr>
<td>caches (SRAM)</td>
<td>10ns</td>
<td>1e6</td>
<td>100.00 $/MB</td>
</tr>
<tr>
<td>main memory (DRAM)</td>
<td>100ns</td>
<td>1e9</td>
<td>1.00 $/MB</td>
</tr>
<tr>
<td>hard disk</td>
<td>5000ns</td>
<td>1e11</td>
<td>.05 $/MB</td>
</tr>
</tbody>
</table>

As the technology improves and the costs go down over time, the typical size of each layer goes up. The ratio in access time between two layers influences the design of the computer hardware. When the ratio changes significantly a different design may achieve better performance.
Cloud Computing

A final note about computers. The computing platform today is less concerned about the individual computer and more concerned about the network of interconnected computers on the Internet.

_The network is the computer_
Slogan of Sun Microsystems

Cue _The Network is the Computer_
Java as a Teaching Language

The subject of this course is programming and teaching programming without any particular programming language does not seem possible. There are good reasons to learn any programming language. There is no good reason to be proficient in one programming language over the rest. There is no best language for learning the others. So a comprise of different pedagogic, societal, practical, and scientific factors govern the choose of Java.
Buzz about Java might mean more jobs, more student engagement.

- TIOBE based on search results
- RedMonk based on github/stack overflow
History of Java

• 1991: a small group led by James Gosling at Sun Microsystems rejected C/C++ as the basis for digital consumer devices.
• 1993: Failed in win the contract from Time-Warner for the interactive cable television trial in Orlando, Florida.
• 1995: WWW, browsers, Java, applets
• 2009: Oracle buys Sun
• 2010: Oracle sues Google over Java use in Android Java. "The lawsuit is one battle in a whole war of the mobile industry. Virtually every major player is locked in courtroom battles with another – many fighting on multiple fronts. Software patents, which tend to be broad and subject to multiple interpretations, make for useful tools to bludgeon competitors."
• 2016 Android witches to OpenJDK

"The lawsuit is one battle in a whole war of the mobile industry. Virtually every major player is locked in courtroom battles with another – many fighting on multiple fronts. Software patents, which tend to be broad and subject to multiple interpretations, make for useful tools to bludgeon competitors."
Versions

JDK 1.0      January 1996
JDK 1.1      February 1997
J2SE 1.2     December 1998
J2SE 1.3     May 2000
J2SE 1.4     February 2002
J2SE 5.0     September 2004
Java SE 6    December 2006
Java SE 7    July 2011
Java SE 8    March 2014
Java SE 9    September 2017
Java SE 10   March 2018
Java SE 11   September 2018
Java SE 12   March 2019
Java SE 13   September 2019
Java SE 14   March 2020
Java SE 15   September 2020
• 5 Generics, for-each loop, autoboxing, var args
• 8 Lambda
• 9 Modual system—project Jigsaw
• 10 Inferring the type of local variables from context
• 11 Local variable type inference with var
• 13 Switch expressions
• 15 Text blocks. Preview: sealed classes, record classes.
Java platform overview
• Java platform, Standard Edition (Java SE) is a computing platform for development and deployment of software. Java SE was formerly known as Java 2 Platform, Standard Edition (J2SE).

• Java Platform, Enterprise Edition (Java EE) is a related platform that includes all the classes in Java SE, plus a number that are more useful to programs that run on servers as opposed to workstations. Java Platform, Micro Edition (Java ME) is a related platform containing classes for resource-constrained devices such as cell phones.

• The Java Development Kit (JDK) includes the translator and other tools. The Java Runtime Environment (JRE) does not.

• OpenJDK (Open Java Development Kit) is a free and open-source implementation of the Java SE. The OpenJDK is the official reference implementation of Java and developed by Oracle Corporation, IBM, Apple, SAP AG, and others.
Java SE platform overview
Java SE (Standard Edition) 8 platform overview
Java SE (Standard Edition) Conceptual Diagram
Java 15 Development Kit Tools
40 in Total

- jaotc - static compiler that produces native code for compiled Java methods
Java 15 Development Kit Tools
40 in Total

• javaotc - static compiler that produces native code for compiled Java methods
• jar - create and manipulates an archive for classes and resources
Java 15 Development Kit Tools

40 in Total

- jaotc - static compiler that produces native code for compiled Java methods
- jar - create and manipulate an archive for classes and resources
- jarsigner - sign and verify Java Archive (JAR) files
Java 15 Development Kit Tools
40 in Total

- jaotc - static compiler that produces native code for compiled Java methods
- jar - create and manipulate an archive for classes and resources
- jarsigner - sign and verify Java Archive (JAR) files
- java - launch a Java application
Java 15 Development Kit Tools
40 in Total

- jaotc - static compiler that produces native code for compiled Java methods
- jar - create and manipulate an archive for classes and resources
- jarsigner - sign and verify Java Archive (JAR) files
- java - launch a Java application
- javac - read Java class definitions and compile them into bytecode code
Java 15 Development Kit Tools
40 in Total

- **jar** - create and manipulate an archive for classes and resources
- **jarsigner** - sign and verify Java Archive (JAR) files
- **java** - launch a Java application
- **javac** - read Java class definitions and compile them into bytecode code
- **javadoc** - generate HTML pages of API documentation from Java source files
Java 15 Development Kit Tools
40 in Total

- jarsigner - sign and verify Java Archive (JAR) files
- java - launch a Java application
- javac - read Java class definitions and compile them into bytecode code
- javadoc - generate HTML pages of API documentation from Java source files
- javap - disassemble one or more class files
Java 15 Development Kit Tools
40 in Total

- `java` - launch a Java application
- `javac` - read Java class definitions and compile them into bytecode code
- `javadoc` - generate HTML pages of API documentation from Java source files
- `javap` - disassemble one or more class files
- `jcmd` - send diagnostic command requests to a running JVM
Java 15 Development Kit Tools
40 in Total

• javac - read Java class definitions and compile them into bytecode code
• javadoc - generate HTML pages of API documentation from Java source files
• javap - disassemble one or more class files
• jcmd - send diagnostic command requests to a running JVM
• jconsole - start a graphical console to monitor and manage Java applications
Java 15 Development Kit Tools

40 in Total

- javadoc - generate HTML pages of API documentation from Java source files
- javap - disassemble one or more class files
- jcmd - send diagnostic command requests to a running JVM
- jconsole - start a graphical console to monitor and manage Java applications
- jdb - find and fix bugs in Java platform programs
Java 15 Development Kit Tools
40 in Total

- javap - disassemble one or more class files
- jcmd - send diagnostic command requests to a running JVM
- jconsole - start a graphical console to monitor and manage Java applications
- jdb - find and fix bugs in Java platform programs
- jdeprscan - static analysis tool that scans a jar file (or some other aggregation of class files) for uses of deprecated API elements
Java 15 Development Kit Tools
40 in Total

- **jcmd** - send diagnostic command requests to a running JVM
- **jconsole** - start a graphical console to monitor and manage Java applications
- **jdb** - find and fix bugs in Java platform programs
- **jdeprscan** - static analysis tool that scans a jar file (or some other aggregation of class files) for uses of deprecated API elements
- **jdeps** - launch the Java class dependency analyzer
Java 15 Development Kit Tools
40 in Total

• jconsole - start a graphical console to monitor and manage Java applications
• jdb - find and fix bugs in Java platform programs
• jdeprscan - static analysis tool that scans a jar file (or some other aggregation of class files) for uses of deprecated API elements
• jdeps - launch the Java class dependency analyzer
• jfr - parse and print Flight Recorder files
Java 15 Development Kit Tools
40 in Total

- **jdb** - find and fix bugs in Java platform programs
- **jdepscan** - static analysis tool that scans a jar file (or some other aggregation of class files) for uses of deprecated API elements
- **jdeps** - launch the Java class dependency analyzer
- **jfr** - parse and print Flight Recorder files
- **jhssdb** - a postmortem debugger to analyze the content of a core dump from a crashed JVM
Java 15 Development Kit Tools
40 in Total

- jdeprscan - static analysis tool that scans a jar file (or some other aggregation of class files) for uses of deprecated API elements
- jdeps - launch the Java class dependency analyzer
- jfr - parse and print Flight Recorder files
- jhsdb - a postmortem debugger to analyze the content of a core dump from a crashed JVM
- jinfo - generate Java configuration information for a specified Java process
Java 15 Development Kit Tools
40 in Total

- jdeps - launch the Java class dependency analyzer
- jfr - parse and print Flight Recorder files
- jhsdb - a postmortem debugger to analyze the content of a core dump from a crashed JVM
- jinfo - generate Java configuration information for a specified Java process
- jlink - assemble set of modules and their dependencies into a runtime image
Java 15 Development Kit Tools
40 in Total

- jfr - parse and print Flight Recorder files
- jhsdb - a postmortem debugger to analyze the content of a core dump from a crashed JVM
- jinfo - generate Java configuration information for a specified Java process
- jlink - assemble set of modules and their dependencies into a runtime image
- jmap - print details of a specified process
Java 15 Development Kit Tools

40 in Total

- `jhsdb` - a postmortem debugger to analyze the content of a core dump from a crashed JVM
- `jinfo` - generate Java configuration information for a specified Java process
- `jlink` - assemble set of modules and their dependencies into a runtime image
- `jmap` - print details of a specified process
- `jmod` - create JMOD files and list the content of existing JMOD files
Java 15 Development Kit Tools
40 in Total

- jinfo - generate Java configuration information for a specified Java process
- jlink - assemble set of modules and their dependencies into a runtime image
- jmap - print details of a specified process
- jmod - create JMOD files and list the content of existing JMOD files
- jpackage - package a self-contained Java application
Java 15 Development Kit Tools
40 in Total

- jlink - assemble set of modules and their dependencies into a runtime image
- jmap - print details of a specified process
- jmod - create JMOD files and list the content of existing JMOD files
- jpackage - package a self-contained Java application
- jps - list the instrumented JVMs on the target system
Java 15 Development Kit Tools
40 in Total

• jmap - print details of a specified process
• jmod - create JMOD files and list the content of existing JMOD files
• jpackage - package a self-contained Java application
• jps - list the instrumented JVMs on the target system
• jrunscript - run a command-line script shell that supports interactive and batch modes
Java 15 Development Kit Tools
40 in Total

- jmod - create JMOD files and list the content of existing JMOD files
- jpackage - package a self-contained Java application
- jps - list the instrumented JVMs on the target system
- jrunscript - run a command-line script shell that supports interactive and batch modes
- jshell - interactively evaluate declarations, statements, and expressions in a read-eval-print loop (REPL)
Java 15 Development Kit Tools
40 in Total

- jpackage - package a self-contained Java application
- jps - list the instrumented JVMs on the target system
- jrunscript - run a command-line script shell that supports interactive and batch modes
- jshell - interactively evaluate declarations, statements, and expressions in a read-eval-print loop (REPL)
- jstack - print Java stack traces of Java threads for a specified Java process
Java 15 Development Kit Tools

40 in Total

• jps - list the instrumented JVMs on the target system
• jrunscript - run a command-line script shell that supports interactive and batch modes
• jshell - interactively evaluate declarations, statements, and expressions in a read-eval-print loop (REPL)
• jstack - print Java stack traces of Java threads for a specified Java process
• jstat - monitor JVM statistics
Java 15 Development Kit Tools
40 in Total

• jrunscript - run a command-line script shell that supports interactive and batch modes
• jshell - interactively evaluate declarations, statements, and expressions in a read-eval-print loop (REPL)
• jstack - print Java stack traces of Java threads for a specified Java process
• jstat - monitor JVM statistics
• jstatd - monitor the creation and termination of instrumented Java HotSpot VMs
Java 15 Development Kit Tools
40 in Total

• jshell - interactively evaluate declarations, statements, and expressions in a read-eval-print loop (REPL)
• jstack - print Java stack traces of Java threads for a specified Java process
• jstat - monitor JVM statistics
• jstatd - monitor the creation and termination of instrumented Java HotSpot VMs
• keytool - manage a keystore (database) of cryptographic keys, X.509 certificate chains, and trusted certificates
Java 15 Development Kit Tools
40 in Total

- jstack - print Java stack traces of Java threads for a specified Java process
- jstat - monitor JVM statistics
- jstatd - monitor the creation and termination of instrumented Java HotSpot VMs
- keytool - manage a keystore (database) of cryptographic keys, X.509 certificate chains, and trusted certificates
- rmid - start the activation system daemon that enables objects to be registered and activated in a Java Virtual Machine (JVM)
Java 15 Development Kit Tools
40 in Total

- `jstat` - monitor JVM statistics
- `jstatd` - monitor the creation and termination of instrumented Java HotSpot VMs
- `keytool` - manage a keystore (database) of cryptographic keys, X.509 certificate chains, and trusted certificates
- `rmid` - start the activation system daemon that enables objects to be registered and activated in a Java Virtual Machine (JVM)
- `rmiregistry` - create and start a remote object registry on the specified port on the current host
Java 15 Development Kit Tools
40 in Total

- jstatd - monitor the creation and termination of instrumented Java HotSpot VMs
- keytool - manage a keystore (database) of cryptographic keys, X.509 certificate chains, and trusted certificates
- rmid - start the activation system daemon that enables objects to be registered and activated in a Java Virtual Machine (JVM)
- rmiregistry - create and start a remote object registry on the specified port on the current host
- serialver - return the 'serialVersionUID' for one or more classes in a form suitable for copying into an evolving class
Java 15 Development Kit Tools
40 in Total

• keytool - manage a keystore (database) of cryptographic keys, X.509 certificate chains, and trusted certificates
• rmid - start the activation system daemon that enables objects to be registered and activated in a Java Virtual Machine (JVM)
• rmiregistry - create and start a remote object registry on the specified port on the current host
• serialver - return the ‘serialVersionUID‘ for one or more classes in a form suitable for copying into an evolving class
Java 15 Development Kit Tools
40 in Total

• *rmid* - start the activation system daemon that enables objects to be registered and activated in a Java Virtual Machine (JVM)
• *rmiregistry* - create and start a remote object registry on the specified port on the current host
• *serialver* - return the ‘serialVersionUID’ for one or more classes in a form suitable for copying into an evolving class
Java 15 Development Kit Tools

40 in Total

- `rmiregistry` - create and start a remote object registry on the specified port on the current host
- `serialver` - return the ‘serialVersionUID‘ for one or more classes in a form suitable for copying into an evolving class
Java 15 Development Kit Tools
40 in Total

- `serialver` - return the ‘serialVersionUID’ for one or more classes in a form suitable for copying into an evolving class
Some of the major components surrounding Java:

- Java virtual machine (JVM) specification
- Virtual machine implementation (for Solaris, Window, and Linux), translation tools (java and javac), and development tools
- Java programming language specification
- A core library (the package java.lang), extensive libraries (APIs) for networking, graphics, etc., and additional APIs for special purposes (e.g., telephony)
Additional components surrounding Java:

- API documentation
  (No reference material will be given to you. We expect you to go out on the Internet, find it, and know some parts of it in detail.)

- An IDE for developing Java programs and GUls: Netbeans
  (We expect you to be able to develop Java programs; we don’t explicitly teach using an IDE.)

(In lab and lecture we have other priorities and we expect a lot from you. However, don’t be reluctant to ask your classmates, instructors, the help desk, etc., if you have questions. Asking knowledgeable people is still the fastest way to learn.)
Precarious pyramid
Will programming evolve into specialized (and different) skills for telecommunications, enterprise computing, etc, each supported by sophisticated libraries?
Will programming evolve into specialized (and different) skills for telecommunications, enterprise computing, etc, each supported by sophisticated libraries?

I don’t think so. Clearly, domain specific knowledge and powerful libraries are useful, but the I think programming will always be the glue that is necessary, independent of the application.
• Software development cycle
• Files
• Language translation
• Streams

See notes elsewhere.