Introduction to Algorithms and Data Structures

CSC 1051 – Algorithms and Data Structures I

Dr. Mary-Angela Papalaskari
Department of Computing Sciences
Villanova University

Course website:
www.csc.villanova.edu/~map/1051/

Some slides in this presentation are adapted from the slides accompanying Java Software Solutions by Lewis & Loftus
What is this course about?

• Computer Science
• Problem solving
• Algorithmic thinking
• Data representation
• Software engineering
Our textbook

Java Software Solutions
Foundations of Program Design
Seventh Edition

John Lewis
William Loftus
Course website

www.csc.villanova.edu/~map/1051/

• Course information
• Announcements
• Week by week schedule, including links to relevant course materials (ppt slides, assignments, etc.)
• Links to details about course, including grades, office hours, online discussions, and much more
• Updated several times/week
• Please *bookmark*!
Reverse History of computing

Examine what we already know, travel backwards…

1. What we see now all around us – a connected world of computing

2. Focus on a single “traditional” computer

3. Dig deeper – data and processing
Networks

A network is two or more computers that are connected so that data and resources can be shared.

A Local-Area Network (LAN) covers a small distance and a small number of computers.

A Wide-Area Network (WAN) connects two or more LANs, often over long distances.
The Internet

- **History:** Started as a United States government project, sponsored by the Advanced Research Projects Agency (ARPA) in late 1970’s
  - 1980’s: *ARPANET*
    - the wide area network and Protocols for communication, including url’s developed
  - 1990’s: *World Wide Web*
    - html and web browsers
The World Wide Web

The Internet Protocol (IP) determines how data are routed across network boundaries.

Each **computer** on the Internet has a unique **IP address**, such as: **204.192.116.2**

**Data** are accessed using a **Uniform Resource Locator (URL)**:

- eg: [http://www.cnn.com](http://www.cnn.com)

- A URL specifies a protocol (http), a domain, and possibly specific documents
- Web documents are often defined using the **HyperText Markup Language (HTML)**
The World Wide Web

- The *World Wide Web* allows many different types of information to be accessed using a common interface

- Resources presented include:
  - text, graphics, video, sound, audio, executable programs

- A *browser* is a program which accesses network resources and presents them
  - Popular browsers: Chrome, Internet Explorer, Safari, Firefox
  - My first browser: Mosaic <3
Reverse History of computing

Examine what we already know, travel backwards…

1. What we see now all around us – a connected world of computing

2. Focus on a single “traditional” computer

3. Dig deeper – data and processing
A Computer Specification

• Consider the following specification for a personal computer:
  – 3.07 GHz Intel Core i7 processor
  – 4 GB RAM
  – 750 GB Hard Disk
  – 16x Blu-ray / HD DVD-ROM & 16x DVD+R DVD Burner
  – 17” Flat Screen Video Display with 1280 x 1024 resolution
  – Network Card
Computer Architecture

Diagram of computer architecture:
- Central processing unit
- Main memory
- Bus
- Disk controller
- Video controller
- Controller
- Other peripheral devices
Memory

Main memory is divided into many memory locations (or *cells*)

Each memory cell has a numeric *address*, which uniquely identifies it
Why is main memory called “RAM”????
“Random Access Memory (RAM)"

You don’t have to scan the memory sequentially – go to data directly using the address
What is “ROM”? is it the opposite of “RAM”??
Read Only Memory

What is “ROM”? is it the opposite of “RAM”???
Read Only Memory

What is “ROM”? is it the opposite of “RAM”? NO!

ROM is also random access
RAM vs. ROM

- **RAM** - Random Access Memory
  - synonymous with main memory:
    - fast
    - read/write
    - volatile
    - random access

- **ROM** - Read-Only Memory
  - ROM typically holds the firmware, eg BIOS
    - fast (except in CD-ROM)
    - read only
    - non-volatile
    - random access
CPU and Main Memory

Central Processing Unit

Chip that executes program commands

Main Memory

Primary storage area for programs and data that are in active use

Synonymous with RAM
CPU and Main Memory

Historic note:
Von Neuman architecture

John Von Neuman, USA 1945

Primary storage area for programs and data that are in active use
Synonymous with RAM

Central Processing Unit

Chip that executes program commands

Von Neumann in the 1940s

CSC 1051 M.A. Papalaskari, Villanova University
The Central Processing Unit

- A CPU is on a chip called a microprocessor
- It continuously follows the *fetch-decode-execute cycle*:

  - **fetch**: Retrieve an instruction from main memory
  - **decode**: Determine what the instruction is
  - **execute**: Carry out the instruction
The Central Processing Unit

• A CPU is on a chip called a microprocessor

• It continuously follows the *fetch-decode-execute cycle*:

  - **fetch**: Retrieve an instruction from main memory
  - **decode**: Determine what the instruction is
  - **execute**: Carry out the instruction

  **System clock** controls speed, measured in **gigahertz (GHz)**

CSC 1051 M.A. Papalaskari, Villanova University
The Central Processing Unit

Arithmetic / Logic Unit

Performs calculations and makes decisions

Control Unit

Coordinates processing (system clock, decoding, etc)

Registers

Small, very fast memory
Historic Note: Automatic control of computation

- The concept of a machine that can follow a series of steps - a “program”

- Some early steps:
  - Jacquard loom (France 1801)
  - Babbage's Difference engine and Analytical engine (England 1822)
  - Holerith's census machine (USA 1890)

- Stored program and the fetch/decode/execute cycle (John von Neumann, USA 1945)

- ENIAC - first fully electronic digital computer (Eckert and Mauchley, University of Pennsylvania, 1946)
Jacquard Loom

This portrait of Jacquard was woven in silk on a Jacquard loom and required 24,000 punched cards to create (1839). It was only produced to order. Charles Babbage owned one of these portraits; it inspired him in using punched cards in his analytical engine. Collection of the Science Museum in London, England. (Source: Wikipedia)

punched cards determine the pattern
Charles Babbage & Ada Lovelace

- Designed the Analytical Engine
- First “Programmer” for (not yet built) Analytical Engine

CSC 1051 M.A. Papalaskari, Villanova University
Reverse History of computing

Examine what we already know, travel backwards…

1. What we see now all around us – a connected world of computing

2. Focus on a single “traditional” computer

3. Dig deeper – data and processing
Data Representation

• Computers store all information *digitally*, using *binary* codes:
  
  – numbers
  – text
  – images
  – audio
  – video
  – program instructions
Analog vs. Digital Data

- **Analog**
  - continuous, in direct proportion to the data represented
  - music on a record album - a needle rides on ridges in the grooves that are directly proportional to the voltages sent to the speaker

- **Digital**
  - information is broken down into pieces, and each piece is represented separately
  - *sampling* – record discrete values of the analog representation
Binary Numbers

• Number system consisting of 1’s & 0’s
• Simplest way to represent digital information
• modern computers use binary numbers internally

A binary digit is called a bit - binary digit
A byte is a group of eight bits
Representing and processing bits

- Electronic circuits: high/low voltage
- Magnetic devices (e.g., hard drive): positive/negative
- Optical devices (e.g., DVD): light reflected/not reflected due to microscopic grooves
# Bit Permutations

<table>
<thead>
<tr>
<th>1 bit</th>
<th>2 bits</th>
<th>3 bits</th>
<th>4 bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>00</td>
<td>000</td>
<td>0000</td>
</tr>
<tr>
<td>1</td>
<td>01</td>
<td>001</td>
<td>0001</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>010</td>
<td>0010</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>011</td>
<td>0011</td>
</tr>
<tr>
<td></td>
<td></td>
<td>100</td>
<td>0100</td>
</tr>
<tr>
<td></td>
<td></td>
<td>101</td>
<td>0101</td>
</tr>
<tr>
<td></td>
<td></td>
<td>110</td>
<td>0110</td>
</tr>
<tr>
<td></td>
<td></td>
<td>111</td>
<td>0111</td>
</tr>
</tbody>
</table>

Each additional bit doubles the number of possible permutations
Bit Permutations

• How many permutations of N bits?
• How many bits are needed to represent 64 items?
• How many bits are needed to represent 100 items?

How many items can be represented by

1 bit ?
2 bits ?
3 bits ?
4 bits ?
5 bits ?
Binary Representation of Information

• Computers store all information *digitally*, using *binary* codes:
  
  – numbers
  – text
  – images
  – audio
  – video
  – program instructions
Representing Text Digitally

• For example, every character is stored as a number, including spaces, digits, and punctuation

• Corresponding upper and lower case letters are separate characters

**Hi, Heather.**

```
72 105 44 32 72 101 97 116 104 101 114 46
```

```
01100001 binary
```

ASCII / UNICODE

CSC 1051 M.A. Papalaskari, Villanova University
Representing Images

- **Bitmap**: 1 bit
- **Grayscale**: 8 bits
- **RGB Color**: 3 colors: red, green, blue; 8 bits/color; 24 bits
Program instructions are also encoded in binary.

E.g., could be the code that causes input of a symbol from the keyboard.
Memory devices store data of **all kinds**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>9278</td>
<td></td>
</tr>
<tr>
<td>9279</td>
<td>10011010</td>
</tr>
<tr>
<td>9280</td>
<td></td>
</tr>
<tr>
<td>9281</td>
<td></td>
</tr>
<tr>
<td>9282</td>
<td></td>
</tr>
<tr>
<td>9283</td>
<td></td>
</tr>
<tr>
<td>9284</td>
<td></td>
</tr>
<tr>
<td>9285</td>
<td></td>
</tr>
<tr>
<td>9286</td>
<td></td>
</tr>
</tbody>
</table>

a number? a letter? the red component of a pixel? a program instruction?
Memory devices store data of all kinds.

Each memory cell stores a set number of bits (usually 8 bits, or one byte).

Large values are stored in consecutive memory locations.

CSC 1051 M.A. Papalaskari, Villanova University
Historic note: Great human developments that gave rise to the modern computer

- Mechanization of arithmetic – the concepts of numbers, symbols, algorithms, and computation
- Automatic control of computation – a “program” to control operations (fetch/decode/execute cycle and the stored program concept)
Historic Note: Mechanization of arithmetic

• Development of number systems
  – Abacus (2400 BC)
  – Number systems (Babylonian, Greek, Roman, Arabic 1000 BC - 800 AD)

• The notion of an algorithm
  – Euclid (300 BC)
  – al-Khwārizmī (780 AD)

• Creation of special purpose calculators
  – Stonehenge (1900-1600 BC)
  – Napier's bones (1600, a precursor of the slide rule)
  – Pascal's adder (1642)
  – Leibniz's calculator (1670s)
  – modern calculators
Mechanization of Arithmetic
+
Automatic Control of Computation
=
Modern Computer
Computer Science

Can be viewed as a culmination of humanity’s search for understanding of:

• Problem solving
• Mechanization
• Computation
• Representation & encoding
• Abstraction

Just like Physics and other sciences branched off from philosophy during the renaissance, so CS emerged in the 20th century from the work of philosophers and mathematicians (with the help of dedicated, visionary practitioners, experimental scientists and engineers).
Part 2 – introduction to Java
Hardware and Software

• Hardware
  – the physical, tangible parts of a computer
  – keyboard, monitor, disks, wires, chips, etc.

• Software
  – programs and data
  – a *program* is a series of instructions

• A computer requires both hardware and software

• Each is essentially useless without the other
Software – What is it?
Communicating with a Computer

- Programming language:
  - A series of specifically defined commands
  - Given by human programmers
  - To give directions to the digital computers
Translation Needed

- Special program to translate into binary
- Programmer writes – **Source code**
- Translation produces the binary equivalent – **Object code**
- The translator is an assembler, compiler, or an interpreter
  - Takes in the source code
  - Yields computer understandable instructions
Java Program Structure

• In the Java programming language:
  – A program is made up of one or more classes
  – A class contains one or more methods
  – A method contains program statements

• These terms will be explored in detail throughout the course

• A Java application always contains a method called main

• See Lincoln.java
public class Lincoln {
    
    public static void main (String[] args) {
        System.out.println("A quote by Abraham Lincoln:");
        System.out.println("Whatever you are, be a good one.");
    }
}
Java Program Structure

```java
// comments about the class
public class MyProgram
{
    class header

    class body

    Comments can be placed almost anywhere

}
```
Java Program Structure

// comments about the class
public class MyProgram
{
    // comments about the method
    public static void main (String[] args)
    {
        // method body
    }
}
Comments

• Comments in a program are called *inline documentation*

• They should be included to explain the purpose of the program and describe processing steps

• They do not affect how a program works

• Java comments can take three forms:

  // Basic this comment runs to the end of the line

  /* Basic this comment runs to the terminating symbol, even across line breaks */

  /** this is a javadoc comment */
Identifiers

- *Identifiers* are the words a programmer uses in a program
- An identifier can be made up of letters, digits, the underscore character ( _ ), and the dollar sign
- Identifiers cannot begin with a digit
- Java is case sensitive - *Total, total, and TOTAL* are different identifiers
- By convention, programmers use different case styles for different types of identifiers, such as
  - *title case* for class names - *Lincoln*
  - *upper case* for constants - *MAXIMUM*
Identifiers

- Sometimes we choose identifiers ourselves when writing a program (such as Lincoln).
- Sometimes we are using another programmer's code, so we use the identifiers that he or she chose (such as println).
- Often we use special identifiers called reserved words that already have a predefined meaning in the language.
- A reserved word cannot be used in any other way.
## Reserved Words

- The Java reserved words:

<table>
<thead>
<tr>
<th>reserved word</th>
<th>reserved word</th>
<th>reserved word</th>
<th>reserved word</th>
</tr>
</thead>
<tbody>
<tr>
<td>abstract</td>
<td>else</td>
<td>interface</td>
<td>switch</td>
</tr>
<tr>
<td>assert</td>
<td>enum</td>
<td>long</td>
<td>synchronized</td>
</tr>
<tr>
<td>boolean</td>
<td>extends</td>
<td>native</td>
<td>this</td>
</tr>
<tr>
<td>break</td>
<td>false</td>
<td>new</td>
<td>throw</td>
</tr>
<tr>
<td>byte</td>
<td>final</td>
<td>null</td>
<td>throws</td>
</tr>
<tr>
<td>case</td>
<td>finally</td>
<td>package</td>
<td>transient</td>
</tr>
<tr>
<td>catch</td>
<td>float</td>
<td>private</td>
<td>true</td>
</tr>
<tr>
<td>char</td>
<td>for</td>
<td>protected</td>
<td>try</td>
</tr>
<tr>
<td>class</td>
<td>goto</td>
<td>public</td>
<td>void</td>
</tr>
<tr>
<td>const</td>
<td>if</td>
<td>return</td>
<td>volatile</td>
</tr>
<tr>
<td>continue</td>
<td>implements</td>
<td>short</td>
<td>while</td>
</tr>
<tr>
<td>default</td>
<td>import</td>
<td>static</td>
<td></td>
</tr>
<tr>
<td>do</td>
<td>instanceof</td>
<td>super</td>
<td></td>
</tr>
<tr>
<td>double</td>
<td>int</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
White Space

• Spaces, blank lines, and tabs are called *white space*

• White space is used to separate words and symbols in a program

• Extra white space is ignored

• A valid Java program can be formatted many ways

• Programs should be formatted to enhance readability, using consistent indentation

• See [Lincoln2.java](Lincoln2.java), [Lincoln3.java](Lincoln3.java)
Program Development

• The mechanics of developing a program include several activities
  – writing the program in a specific programming language (such as Java)
  – translating the program into a form that the computer can execute
  – investigating and fixing various types of errors that can occur

• Software tools can be used to help with all parts of this process
Errors

• A program can have three types of errors

• The compiler will find syntax errors and other basic problems (*compile-time errors*)
  – If compile-time errors exist, an executable version of the program is not created

• A problem can occur during program execution, such as trying to divide by zero, which causes a program to terminate abnormally (*run-time errors*)

• A program may run, but produce incorrect results, perhaps using an incorrect formula (*logical errors*)
Lab 1 errors/changes

• Change the first println to print
• Change the second println to print
• Change the second println to bogus
• Remove the semicolon at the end of one of the statements
• Remove the last brace of the program
• Change main to man
• Something else you tried?
Java Translation

Java source code → Java compiler

Java bytecode → Bytecode interpreter

Java bytecode → Bytecode compiler

Java bytecode → Machine code
Development Environments

• There are many programs that support the development of Java software, including:
  – Sun Java Development Kit (JDK)
  – Sun NetBeans
  – IBM Eclipse
  – IntelliJ IDEA
  – Oracle JDeveloper
  – BlueJ
  – jGRASP

• Though the details of these environments differ, the basic compilation and execution process is essentially the same
Summary

• History of computing
• Computer hardware and software overview
• Programming and programming languages
• An introduction to Java
Next week: Variables

10. Now add some variables to the program MyQuote by including the following statements (type these in after the output statements but before the closing quote of method definition):
   - int x = 42, count = 100;
   - String name = "Steve";

1. Add another output statement that incorporates the values of your variables. For example, try something like this:
   - System.out.println ("Howdy " + name);
   - System.out.println ("The answer is " + x);
   - System.out.println ("And counting up: " + (count + 1));
Homework

• Review Chapter 1
  – **Always** do all self-review exercises when you review material

• Do Exercises EX 1.1-1.8 and 1.15-1.20

• Read Sections 2.1-2.6 to prepare for next week