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/s16/

What is this course about?
• Computer Science
• Problem solving
• Algorithmic thinking
• Data representation
• Software engineering

Course website
www.csc.villanova.edu/~map/1051/s16/

Links to:
• Schedule – topics, slides, projects, labs, code, etc.
• Syllabus – course information
• Piazza – class discussions, announcements
• Blackboard – submit projects, check grades
• Help – extra help available for this course
• Exam archive – past exams and quizzes & solutions

Our textbook
Java Software Solutions
Foundations of Program Design
Seventh Edition
John Lewis
William Loftus

(6th edition is ok, but some exercise numbers may be different)
An old quote

A priest asked: “What is Fate, Master?”
And he answered:
“It is that which gives a beast of burden its reason for existence. It is that which men in former times had to bear upon their backs. It is that which has caused nations to build byways from City to City upon which carts and coaches pass, and alongside which inns have come to be built to stave off Hunger, Thirst and Weariness.”
“And that is Fate?” said the priest.
“Fate... I thought you said Freight,” responded the Master.
“That's all right,” said the priest. “I wanted to know what Freight was too.”
- Kehlog Albran

Source unknown. This quote appeared as one of the “fortunes” displayed by the fortune cookie program on old unix systems. (“fortune” was a program that ran automatically every time you logged out of a unix session and displayed a random, pithy saying.)

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 and World Wide Web

- The Internet Protocol (IP) determines how data are routed across network boundaries.
  - devices are assigned unique IP addresses, e.g., 204.192.116.2
  - data are accessed using a Uniform Resource Locator (URL): eg: http://www.cnn.com
- The World Wide Web provides a common interface to data:
  - text, graphics, video, sound, audio, executable programs
- Web documents often use HyperText Markup Language (HTML)
- A browser is a program which accesses network resources and presents them
  - Popular browsers: Chrome, Internet Explorer, Safari, Firefox
  - My first browser: Mosaic <3
Introduction

The Internet

**History:** Started as a United States government project, sponsored by the Advanced Research Projects Agency (ARPA) in late 1960’s

See also: http://www.internethalloffame.org/internet-history/timeline

- 1970’s and 1980’s: **ARPANET**
  - wide area network
  - protocols for communication

- 1990’s: **World Wide Web**
  - html and web browsers

**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

Memory

CPU and Main Memory

The Central Processing Unit
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

**Historical Note: Automatic control of computation**

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

- **Early efforts**:
  - Jacquard loom (France 1801)
  - Babbage’s Difference engine and Analytical engine (England 1822)
  - Holerith’s census machine (USA 1890)
- **Colossus Mark I** – first electronic computer to be programmable (Alan Turing, England 1944)
- **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)

**Charles Babbage & Ada Lovelace**

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

1945: The word “computer” changes its meaning

The Electronic Numeric Integrator and Calculator (ENIAC)
Programmers Betty Jean Jennings (left) and Fran Bilas (right) operate ENIAC’s main control panel at the Moore School of Electrical Engineering. (U.S. Army photo from the archives of the ARL Technical Library)

Historical Note: Automatic control of computation

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 \textbf{digitally}, using binary codes:
  - numbers
  - text
  - images
  - audio
  - video
  - program instructions

2014: Benedict Cumberbatch shows the world how cool Alan Turing was

The Imitation Game (2014)
PG-13 | 124 min. | Biography, Drama, Thriller | 25 December 2014 (USA)

Your rating: 8.1/10 from 332,068 users | Metascore: 73/100
Reviews: 555 user | 423 critic | 49 from Metacritic.com

During World War II, mathematician Alan Turing tries to crack the enigma code with help from fellow mathematicians.

Director: Morten Tyldum
Writers: Graham Moore, Andrew Hodges (book)
Stars: Benedict Cumberbatch, Keira Knightley, Matthew Goode | See full cast and crew

http://www.imdb.com/title/tt2084970/

Captain Grace Hopper
and other computers
A **byte** is a group of eight bits

- a number?
- a letter?
- the red component of a pixel?
- a program instruction?

Computing devices store use binary codes to represent data of **all kinds**

**Example: Representing Text**

- Characters, including spaces, digits, and punctuation are represented by numeric codes

```
Hi, Heather.
```

<table>
<thead>
<tr>
<th>ASCII</th>
<th>UNICODE</th>
</tr>
</thead>
<tbody>
<tr>
<td>01110100</td>
<td>00000000 01110100</td>
</tr>
</tbody>
</table>

The **ASCII** (American Standard Code for Information Interchange) character set uses eight bits per character, allowing for 256 unique characters.

The **Unicode** character set extends ASCII to sixteen bits per character, allowing for 65,536 unique characters.

**Example: Representing Pixels**

```
red=116
green=84
blue=142
```

**Example: Representing Program Instructions**

Intel opcode for the instruction **JZ** (jump if zero):

```
01110100
```
Introduction

Historical Note: Symbolic Representation & Mechanization of Arithmetic

- Development of number systems & geometry
- The notion of an algorithm
- Creation of special purpose calculators

Historical Note: Symbolic Representation & Mechanization of Arithmetic

Basic human needs: **Symbolism**

.Basic human needs: counting & measuring

- Abacus (China ~2400 BC)
- Number systems (Babylonian, Greek, Roman, Arabic 1000 BC - 800 AD)
- Geometry (Egypt/Greece 300 BC)

Basic human needs: Symbolism

:-) ♫ π

apple ∞ <3

Historical Note: Symbolic Representation & Mechanization of Arithmetic

Basic human needs: counting & measuring

- Abacus (China ~2400 BC)
- Number systems (Babylonian, Greek, Roman, Arabic 1000 BC - 800 AD)
- Geometry (Egypt/Greece 300 BC)
Historical Note: Symbolic Representation & Mechanization of Arithmetic

- The notion of an algorithm

Euclid (300 BC)  Muhammad ibn Musa al-Khwarizmi (800 AD)

Historical Note: Symbolic Representation & Mechanization of Arithmetic

- Creation of special purpose calculators

Stonehenge (1900-1600 BC)  Pascal's adder (1642)  Leibniz's calculator (1670s)

1975 Texas Instruments calculator

Historical Notes:
Trends that gave rise to the modern computer

- Symbolic representation and the 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)
- Connecting the world – networks and telecommunications

= modern computer

Computer Science

A new paradigm in humanity’s search for understanding of:
- Representation & encoding
- Computation
- Problem solving
- Mechanization

History Epilogue: 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

High-level programming languages

- Programmer writes **Source code**
- Translation produces the binary equivalent – **Object code**
- Translation is performed by an assembler, compiler, or interpreter (stay tuned)

Java Translation

- Java source code
- Java bytecode
- Java compiler
- Bytecode interpreter
- Bytecode compiler
- 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
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

Java Program Example

```java
//********************************************************************
// Lincoln.java       Author: Lewis/Loftus
// // Demonstrates the basic structure of a Java application.
//********************************************************************
public class Lincoln
{
    // Prints a presidential quote.
    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
{
    // comments about the method
    public static void main (String[] args)
    {
        // comments can be placed almost anywhere

    }
}
```
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:
  
  // This comment runs to the end of the line

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

White Space (Spaces, blank lines, and tabs)

• Extra white space is ignored

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

  See *Lincoln2.java*, *Lincoln3.java*

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*)
Introduction

The original "bug" found in the relays of Harvard's Mark II computer by Admiral Grace Murray Hopper's team.

Source: en.wikipedia.org/wiki/File:H96566k.jpg

Lab 1:

- Learn about jGrasp - the programming environment that we will be using in this class
  - Compile and run a java program
- Understand the relationship between a Java class name and the name of the .java file where the class is defined
- Practice using basic Java output statements and adding comments
- Learn about variables, string literals, concatenation. E.g.,
  ```
  System.out.println("Hello " + name);
  System.out.println("The answer is " + x);
  System.out.println("Counting... up: " + (count + 1));
  System.out.println("... and down: " + (count - 1));
  ```
- Explore Java syntax
- Experience some errors!

Character Strings

- A string literal is represented by putting double quotes around the text
- Examples:
  ```
  "This is a string literal."
  "123 Main Street"
  "X"
  ```

spaces matter in here!

The println Method

- In the Lincoln program we invoked the println method to print a character string
- The System.out object represents a destination (the monitor screen) to which we can send output
  ```
  System.out.println("Whatever you are, be a good one.");
  ```

object method name information provided to the method (parameters)
The print Method

- In the Lincoln program we invoked the println method to print a character string.
- The System.out object represents a destination (the monitor screen) to which we can send output.
- print is similar to println except that it does not advance to the next line.

```
System.out.print ("Whatever you are, be a good one.");
```

String Concatenation

- The string concatenation operator (+) is used to append one string to the end of another.

```
"And one more " + "thing"
```

Hands on:

- Use MyQuote.java as a starting point (program from Lab 1), focus on this part of the code:

```
System.out.println ("Howdy " + name);
System.out.println ("The answer is " + x);
System.out.print ("Counting... up: " + (count + 1));
System.out.println ("... and down: " + (count - 1));
```

- Try the following:
  1. What if you remove the parentheses around (count + 1)?
  2. What happens if we try this way of breaking a line:

```
System.out.print ("Counting... up: " + (count + 1));
```

- How can we get all this output to print all in one line?

- Other examples (textbook): Countdown.java, Facts.java

Escape Sequences

- What if we wanted to print the quote character? e.g.,

```
System.out.println ("I said "Hello" to you."); // wrong!
```

- An escape sequence is a series of characters that represents a special character.

```
System.out.println ("I said \"Hello\" to you.");
```

- Some Java escape sequences:

<table>
<thead>
<tr>
<th>Escape Sequence</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>\t</td>
<td>tab</td>
</tr>
<tr>
<td>\n</td>
<td>newline</td>
</tr>
<tr>
<td>&quot;</td>
<td>double quote</td>
</tr>
<tr>
<td>'</td>
<td>single quote</td>
</tr>
<tr>
<td>\</td>
<td>backslash</td>
</tr>
</tbody>
</table>
Example from textbook: **Roses.java**

```java
//********************************************************************
// Roses.java  Author: Lewis/Loftus
// Demonstrates the use of escape sequences.
//********************************************************************
public class Roses
{
    // Prints a poem (of sorts) on multiple lines.
    public static void main (String[] args)
    {
        System.out.println("Roses are red,\nViolets are blue,\nSugar is sweet,\nBut I have "commitment issues",\nSo I'd rather just be friends\nAt this point in our relationship.");
    }
}
```

**Output**

```
Roses are red,
Violets are blue,
Sugar is sweet,
But I have "commitment issues",
So I'd rather just be friends
At this point in our relationship.
```

Quick Check

Write a single `println` statement that produces the following output:

"Thank you all for coming to my home tonight," he said mysteriously.

Next: variables

From Lab 1:

```java
int x = 42, count = 100;
String name = "Kripke";
System.out.println ("Howdy " + name);
System.out.println ("The answer is " + x);
name = "Sheldon";
x = 33;
System.out.println ("Howdy " + name);
System.out.println ("The answer is " + x);
```

Variables

- A **variable** is a name for a location in memory
- A variable must be **declared** by specifying the variable’s name and the type of information that it will hold
  ```java
  int sum;
  double milesPerGallon;
  String name, petName;
  ```
Some types of data in Java

<table>
<thead>
<tr>
<th>type</th>
<th>set of values</th>
<th>literal values</th>
<th>operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>char</td>
<td>characters</td>
<td>'A'</td>
<td>compare</td>
</tr>
<tr>
<td>String</td>
<td>sequences of characters</td>
<td>&quot;Hello World&quot;</td>
<td>concatenate</td>
</tr>
<tr>
<td>int</td>
<td>integers</td>
<td>17</td>
<td>add, subtract, multiply, divide</td>
</tr>
<tr>
<td>double</td>
<td>floating-point numbers</td>
<td>3.1415</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>

Identifiers

- Identifiers are used for naming variables, classes, and other components of a program.
- An identifier can be made up of:
  - letters (upper or lower case – case sensitive!)
  - digits (but cannot begin with a digit)
  - underscore character (_)
  - the dollar sign ($_$)
  - NOTHING ELSE!
- Example: Total, total, and TOTAL are different identifiers
- Conventions: use case to indicate whether it is a class or a variable etc.

Reserved Words

These identifiers have a special meaning in Java and cannot be used in any other way:

- abstract
- assert
- boolean
- break
- byte
- case
- catch
- char
- class
- const
- continue
- default
- do
- double
- else
- enum
- extends
- false
- final
- finally
- float
- for
- goto
- if
- implements
- import
- instanceof
- int
- interface
- long
- native
- new
- null
- package
- private
- protected
- public
- return
- short
- static
- strictfp
- switch
- synchronized
- this
- throw
- throws
- transient
- true
- try
- void
- volatile
- while

Summary

- History of computing
- Computer hardware and software overview
- An introduction to Java:
  - Identifiers
  - Comments
  - Errors
  - Strings and printing