Introduction to Java Programming

Lecture 1
CGS 3416 Spring 2017

1/9/2017

Main Components of a computer

- CPU - Central Processing Unit: The "brain" of the computer
  - ISA - Instruction Set Architecture: the specific set of low-level instructions available to a CPU. Differs for various CPU types (Intel Pentium, Mac G4, etc)
- ALU - Arithmetic Logic Unit responsible for performing arithmetic calculations, as well as logical operations (comparisons for equality, inequality, for instance).
- Main Memory (RAM - Random Access Memory)
  - Storage close to CPU
  - Faster to access than hard disk
  - Stores executing programs and data being currently worked on
- Secondary Memory
  - hard disk, floppy disk, CD, DVD, etc.

Main Components of a computer

- Input devices
  - mouse, keyboard, scanner, network card, etc.
- Output devices
  - screen/console, printer, network card, etc.
- Operating System
  - Examples: Mac OS, Windows XP, Linux
  - Controls computer operations
  - Manages allocation of resources for currently running applications

Memory Concepts

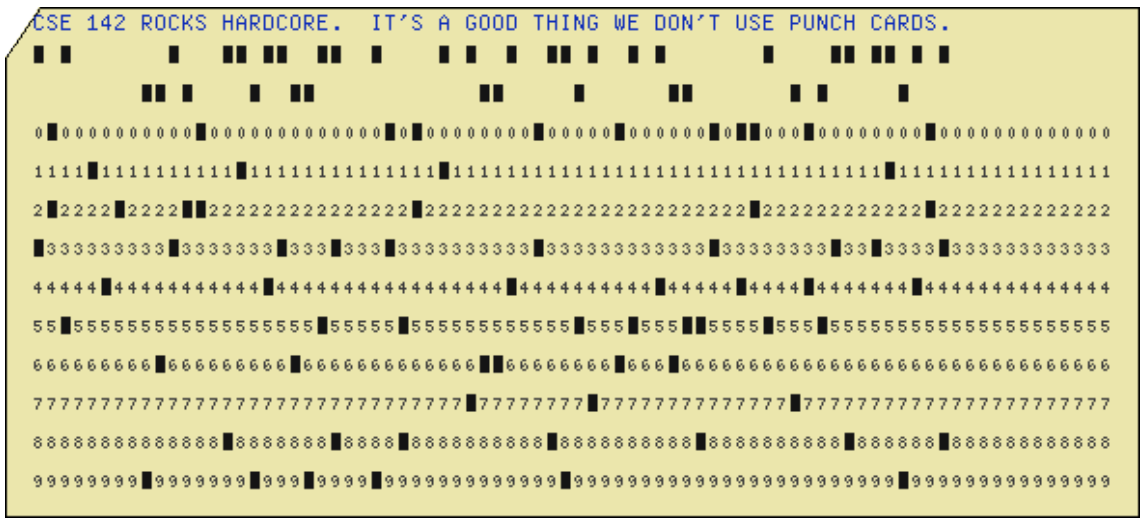
- Within a computer all data is stored as a sequence of bits, each of which can take on one of two values
- bit: a binary digit
  - Stores the value 0 or 1
  - Smallest unit of storage in a computer
- byte: 8 bits
  - Smallest addressable unit of storage in a computer
  - Storage units (variables) in a program are 1 or more bytes
  - Each byte in memory has an address (a number that identifies the location)

Programming, and Programming Languages

Program - a set of instructions for a computer to execute

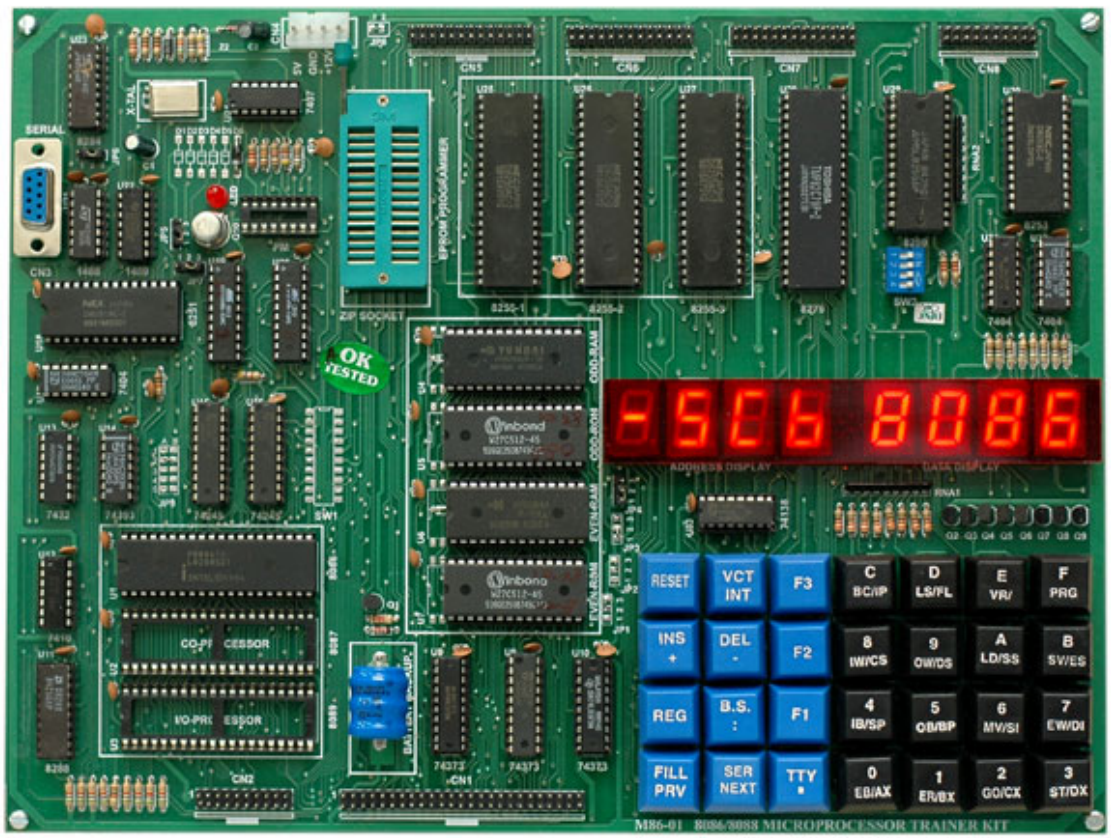
Evolution of Programming languages

- Machine Language
  - Based on machine's core instruction set
  - Needed by computer, hard for humans to read (1's and 0's)
  - Example: 11101101010101100011101010



Programming, and Programming Languages

- Assembly Language
  - translation of machine instructions to symbols, slightly easier for humans to read
  - Example: ADD $R1, $R2, $R3



Programming, and Programming Languages

- High-level procedural languages
  - Abstraction of concepts into more human-readable terms
  - Closer to "natural language" (i.e. what we speak)
  - Easy to write and design, but must be translated for computer
  - Examples include C, Pascal, Fortran
- Object-oriented languages
  - Abstraction taken farther than procedural languages
  - Objects model real-world objects, not only storing data (attributes), but having inherent behaviors (operations, functions)
  - Easier to design and write good, portable, maintainable code
  - Examples include Smalltalk, C++, Java

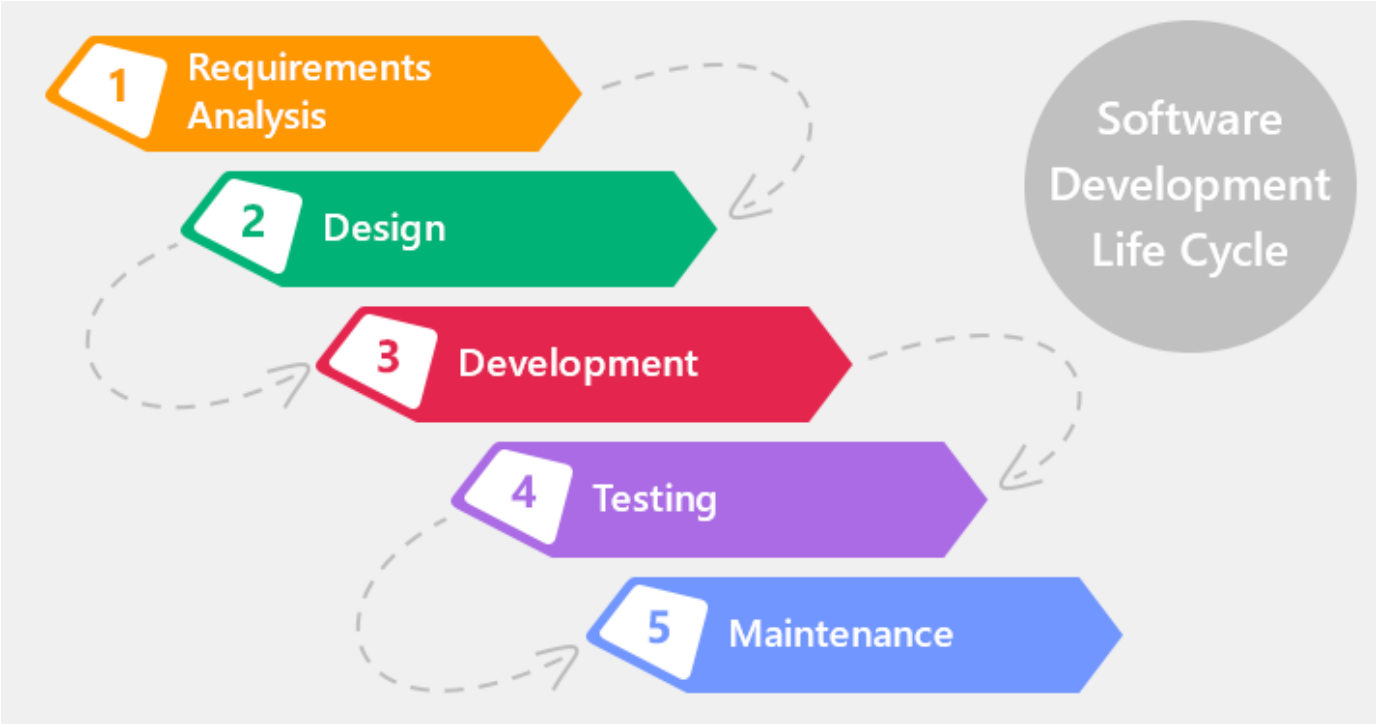
Code Translation

Bridging the gap between high-level code and machine code

- Interpreted languages – source code is directly run on an interpreter, a program that runs the code statements
- Compiled Languages
  - A compiler program translates source code (what the programmer writes) to machine language (object code)
  - A linker program puts various object code files together into an executable program (or other target type, like a DLL)
  - C and C++ are compiled languages
- Java is a mix of both!

Software Development

Involves more than just writing code



Software Development

- Analysis and problem definition
- Design - includes design of program or system structure, algorithms, user-interfaces, and more
- Implementation (coding)
- Testing - can be done during design, during implementation, and after implementation
- Maintenance - usually the major cost of a software system. Not part of "development", but definitely part of the software life cycle

The Java Language

- Java is a programming language that evolved from C++
  - Both are object-oriented
  - They both have much of the same syntax
- Began in the early 90's, originally used for programming in intelligent consumer-electronic devices (internal chips, etc).
- Was originally named Oak by its creator, but changed when it was reliazed that there was already a language called Oak
- When the Web took off in the early 90s, Java gained popularity for use in adding dynamic content to web pages
  - While applets surely helped Java gain quick popularity, they are by no means the most important use of the language

The Java Language

- Java is now used for a wide variety of purposes.
- Its large and rich set of pre-built packages makes it a very popular choice of software developers
- The Java language specification is owned and controlled by Sun Microsystems (An Oracle Company)
- API (Application Programmer Interface) documentation for standard libraries available on the Oracle website.
- Standard Development Kit, along with other development tools can be downloaded from http://www.oracle.com/technetwork/java/javase/downloads/index.html
- Latest version is Java SE 8 – Java Standard Edition 8.0

Compiling and Running a Java program

- Java code compiled to an intermediate level – bytecode
- bytecode runs on an interpreter – the Java Virtual Machine
- Each platform needs its own JVM, but the same bytecode (generally speaking) runs on any JVM on any platform (i.e. the compiled version is portable)
- Typically Slower runtime than languages like C++, since running on an interpreter (and due to other factors)

Basic Creation and Execution of a Java program

1. Create source code with a text editor, store to disk
   - Source code is just a plain text file.
   - In Java, we give the filename an extension of .java to identify it as a source code file
2. Compilation – The compiler does syntax checking, translation to bytecode in files with the .class extension
   - bytecode is a translation of the source code to an intermediate level of code
3. Execution of Java program
   - The loader is part of the Java Virtual Machine
   - It loads the bytecode into memory and executes the instructions via an interpreter for the given platform (Windows, Mac, Linux, etc)

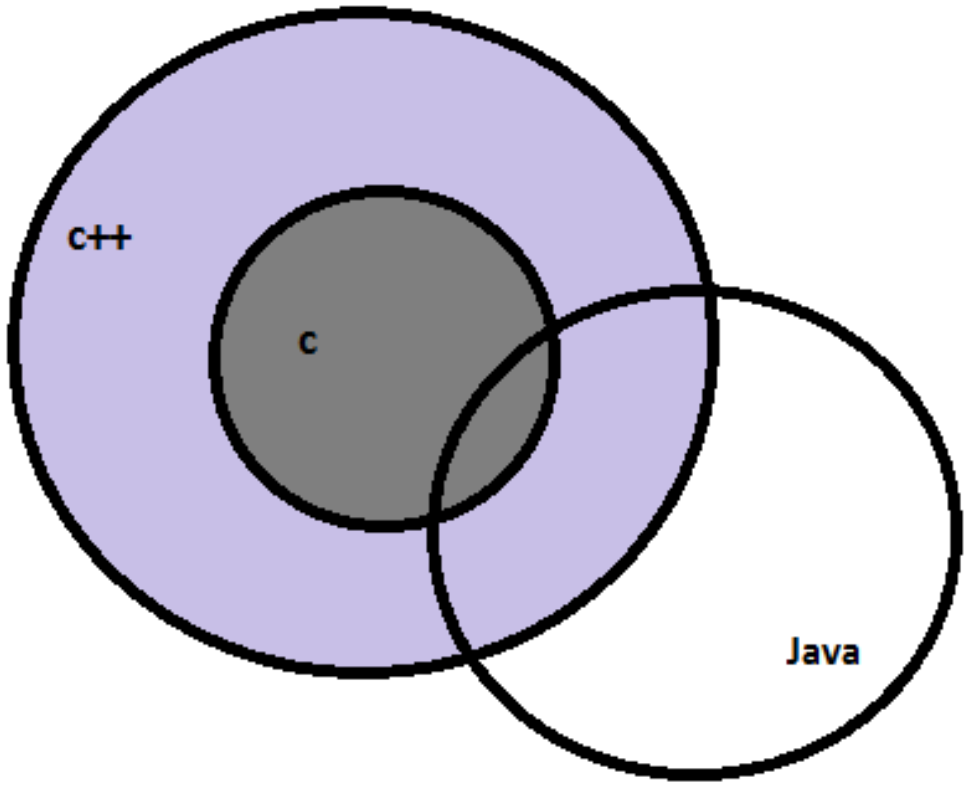
Integrated Development Environments

- An Integrated Development Environment (IDE) is a software package that includes all necessary tools for creating a program. Typcially includes:
  - Text editor
  - Compiler
  - Loader
  - Debugger
  - Ability to launch and run applications from within IDE environment
  - Other useful tools
- Java IDEs frequently use the Java Standard Development Kit (SDK) tools underneath, and provide a graphical interface through menus to access the underlying tools.
- Examples of Java IDEs
  - IntelliJ
  - NetBeans
  - Eclipse

Some Important Java Tools

- javac - java compiler
- java - java interpreter
- jar - the java archive utility
- javadoc - utility for auto-generating Java documentation API pages
- JSP - Java Server Pages
- JRE - Java Runtime Environment
- J2SDK - Java 2 Standard Development Kit (sometimes JDK, Java Development Kit, for short) – includes JRE

Relationship between Java, C++,C



Overlapping of C, C++, and Java

Some benefits of Java (over C++)

- Vast collection of packages available in the Standard Development Kit (SDK)
  - Easy-to-use API descriptions in HTML format on the Sun web site
  - Standard format for building API descriptions for classes
- Easier to build programs with graphic interfaces (GUI)
  - latest packages for GUI (Swing classes) not platform specific
  - compiled bytecode runs on multiple platforms
  - In C++, one would commonly have to use the GUI libraries for each different platform
- Some syntax has been made simplified (no header file, no destructors)
- Java Runtime Environment (JRE) does some things for you
  - Automatic garbage collection (for dynamically allocated objects)
  - more dynamic run-time checking
  - automatic dynamic binding and polymorphic behavior

Some benefits of C++ (over Java)

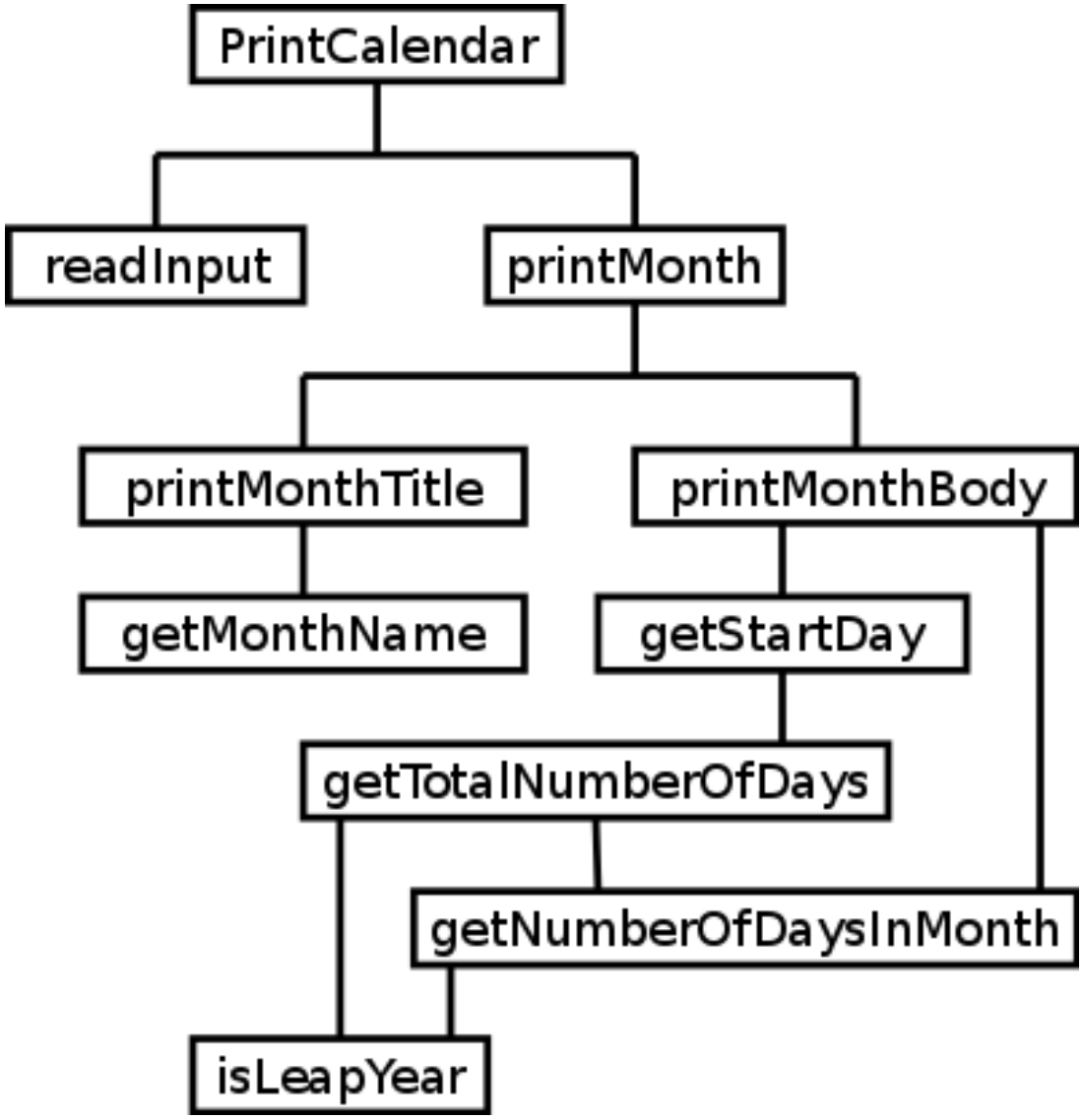
- Programmer has more control and power in C++
  - In C++, programmer responsible for the details
  - Control over addresses with pointers
  - More control over efficient execution time and resource allocation/deallocation
- C++ programs will typically run faster, because
  - compiled to machine's native instruction set
  - dynamic allocation doesn't have to be used for all objects
  - programmer has more power to optimize what they want
- C++ still has some extra and versatile features (that Java doesn't), like operator overloading and multiple inheritance, global variables.

Programming is about Problem Solving

- Algorithm - a finite sequence of steps to perform a specific task
  - To solve a problem, you have to come up with the necessary step-by-step process before you can code it
  - This is often the trickiest part of programming
- Some useful tools and techniques for formulating an algorithm
  - Top-down Refinement: Decomposing a task into smaller and simpler steps, then breaking down each step into smaller steps, etc
  - Pseudocode: Writing algorithms informally in a mixture of natural language and general types of code statements
  - Flowcharting: If you can visualize it, it's often easier to follow and understand!

Top-down Refinement

Printing a calendar for any given month.



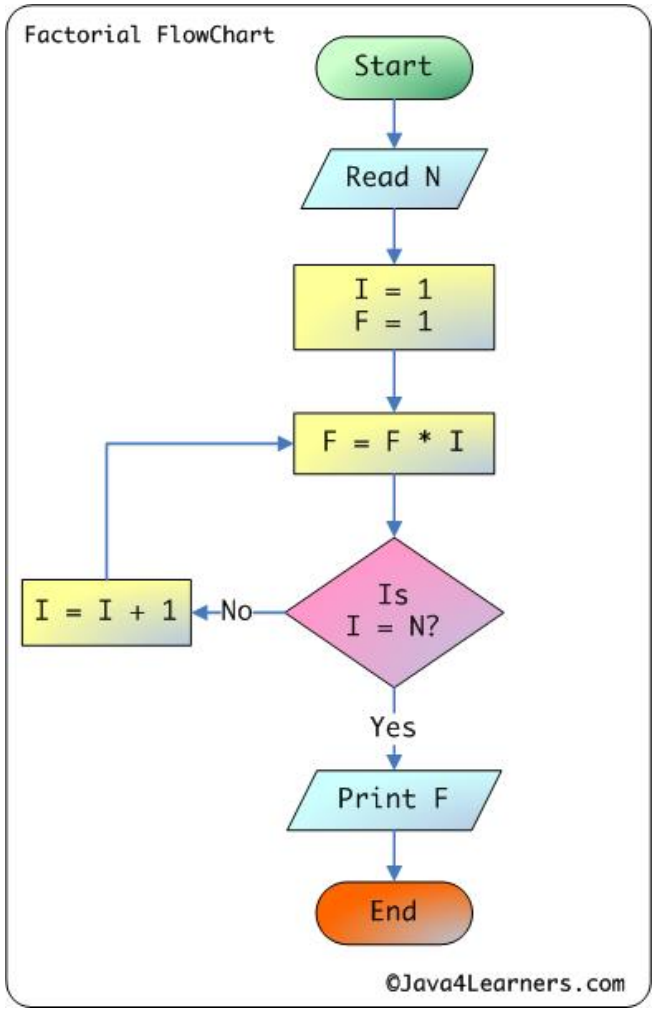
Example of a Pseudocode

Write down a pseudocode for the following problem:
The program gets a number from the user and then if it is less 100, it prints cheap; if it is less than 200, it prints acceptable, else it prints expensive.

- **Example Pseudocode:**
  1. **begin**
  2. **Input** price
  3. **If** price < 100 then print “cheap”
  4. **Else** if price < 200 then print “acceptable”
  5. **Else** print “expensive”
  6. **end**

Example of a Flow Chart

Write a program to compute factorial of a number in Java



Programming is about Problem Solving

- Testing - algorithms must also be tested!
  - Does it do what is required?
  - Does it handle all possible situations?
- Syntax vs. Semantics
  - Syntax – the grammar of a language.
    A syntax error: "I is a programmer."
  - Semantics – the meaning of language constructs
    Correct syntax, but a semantic error: "The car ate the lemur."