CS1004: Intro to CS in Java, Spring 2005

Lecture #5: Java basics, data representation

Janak J Parekh
janak@cs.columbia.edu

Administrivia

- Two more TAs
- HW#1 due next Tuesday
  - Submission instructions up
  - Read the webboard!
- By the way, I update the slides after class
  - Usually truncate material that we didn't get to

Agenda

- Finish Java introduction
- Brief discussion on software development
- Introduce memory representation in computers

Identifiers

- Sometimes we choose identifiers ourselves when writing a program (such as `HelloWorld`)
- 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:

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


Whitespace

- Spaces, blank lines, and tabs are called *whitespace*
- 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
- Emacs helps you to automatically enforce this

Program development revisited

- The mechanics of developing a program include several activities
  - writing the program in a specific programming language (such as Java)
  - translating (compiling) the program into a form that the computer can execute
  - investigating and fixing various types of errors that can occur
- Software tools are used throughout this process

Running code

- The microprocessor (CPU) of a computer is its "heart", and is responsible for running code, but it doesn't understand Java directly
- Instead, each type of CPU has its own specific *machine language* ("instruction set architecture")
  - Comparatively primitive – like a fast, sophisticated scientific calculator
  - Intel/AMD processors use the *x86 ISA*
  - Mac computers use G4/G5 processors with a *PowerPC ISA*

8

Language levels

- There are four programming language levels:
  - machine language
  - assembly language
    - "Shorthand" for machine language
  - high-level language (i.e., Java, C, C++)
  - fourth-generation language (SQL, others)
- Levels were created to make it easier for a human being to read and write programs
- We'll see examples of machine code and assembly next week

Compiler

- A program must be translated into machine language before it can be executed
- A *compiler* is a software tool which translates higher-level *source code* into a specific target language
- Often, that target language is the actual machine language for a particular CPU type
  - C, C++ do this
- The Java approach is somewhat different

10

Java "translation"

- The Java compiler (**javac**) translates Java source code into a special representation called *bytecode*
  - Bytecode is *not* the machine language for any traditional CPU, although it's somewhat similar
- Another software tool, called an *interpreter*, translates bytecode into machine language and executes it "on the fly"
  - It's actually *recompiling* the bytecode into machine language as you run the program via the **java** tool

11

Why so complex?

- Compiled Java code is not tied to any particular machine
- In other words, you can compile a program, give someone the .class file, and they can run it without having to worry about compilation, on one of many different types of computers
- Java is considered to be *architecture-neutral*
- *Not* the case with C/C++; you need to recompile the original code for every possible machine, and different machines may behave a little differently

Syntax vs. semantics

- The *syntax rules* of a language define how we can put together symbols, reserved words, and identifiers to make a valid program
  - In English, we call this *grammar:* sentence structure, punctuation, etc.
- The *semantics* of a program statement define what that statement means (its purpose or role in a program)
  - What does the sentence actually *mean?*

13

Why do we care?

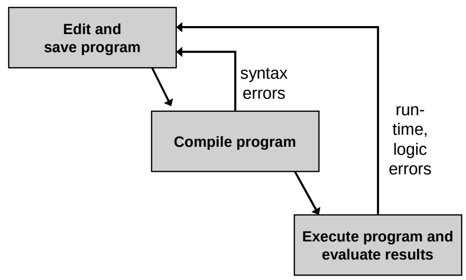
- A program that is syntactically correct is not necessarily logically (semantically) correct
- A program will always do what we tell it to do, not what we meant to tell it to do
- Example: a program to pack soda cans into crates
  - Given *n* cans, we need *n/6* crates.
  - `int nCrates = numCans / 6;`
  - This is *syntactically correct*, but may have semantic flaws

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, the executable bytecode is not generated
- A problem can occur during program execution, such as 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*)
- Semantic errors consist of *both* run-time and logical errors

15

Basic Program Development "Cycle"



Problem Solving

- Solving a problem consists of multiple activities:
  - Understand the problem
  - Design a solution (algorithm)
  - Consider alternatives and refine the solution
  - Implement the solution (program)
  - Test the solution
- These activities are not purely linear – they overlap and interact

Problem Solving

- The key to designing a solution is breaking it down into manageable pieces
- When writing software, we design separate pieces that are responsible for certain parts of the solution
- An *object-oriented approach* lends itself to this kind of solution decomposition
  - Pieces called *objects* and *classes*

What is OOP?

- Java is an object-oriented programming language
- As the term implies, an object is a fundamental entity in a Java program
  - Often translate to "real" entities, e.g., an *Employee* object
  - Each Employee object handles the processing and data management related to that employee

Objects

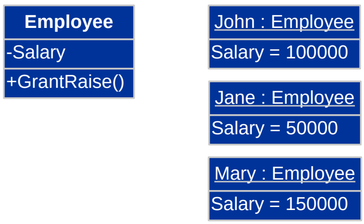
- An object has:
  - *state* - descriptive characteristics (storage)
  - *behaviors* - what it can do (algorithms)
- The state of a bank account includes its account number and its current balance
- The behaviors associated with a bank account include the ability to make deposits and withdrawals
- Note that the behavior of an object might change its state

20

Classes

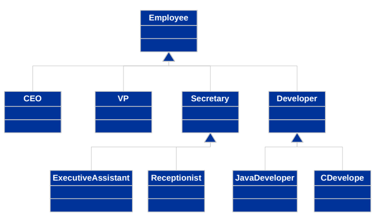
- An object is defined by a *class*
  - "Blueprint" of an object
- The class uses *methods* to define the behaviors of the object
- The class that contains the main method of a Java program "represents" the entire program
- A class represents a concept, and an object represents the embodiment of that concept
  - John, Jane, Mary (*objects*) are Employees (*class*)
  - Multiple objects can be created from the same class

Objects and Classes



Inheritance

- One class can be used to derive another via *inheritance*
- Classes can be organized into hierarchies
- Don't confuse this with objects!
- We'll think more about this later; don't worry too much about it for now

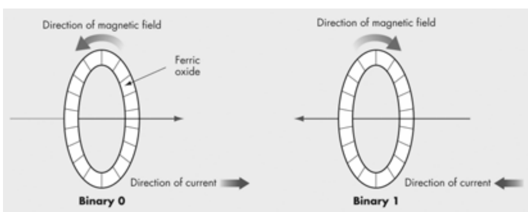


How is all this stuff stored, anyway?

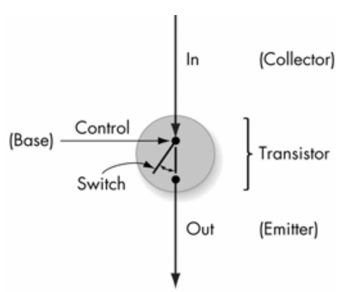
- A computer's internal storage techniques are different from the way people represent information in daily lives
- Information inside a digital computer is stored as a collection of binary data
  - *Everything* is stored as 0s and 1s ultimately
  - Convention
    - We call any individual 0 or 1 a *bit*
    - A *byte* can vary, but most computers today equate 8 bits to one byte

Why binary!?

- Electronic devices are most reliable in a bistable environment
- Bistable environment
  - Distinguishing only two electronic states: current flowing or not, or direction of flow
- Computers are bistable: hence binary representations
- It *is* theoretically possible to build base-10 computers, but less stable
  - Different voltages for each value (analog cassettes?)
  - Risk of degradation over time
- So how do we store it?



Using Magnetic Cores to Represent Binary Values



Transistors (usually designed as semiconductors):
use a control to turn on and off flow, i.e., really tiny switches

Representing numbers

- Decimal numbering system
  - Base-10
  - Each position is a power of 10
    $3052 = 3 \times 10^3 + 0 \times 10^2 + 5 \times 10^1 + 2 \times 10^0$
- Binary numbering system
  - Base-2
  - Built from ones and zeros
  - Each position is a power of 2
    $1101 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$

Binary	Decimal	Binary	Decimal
0	0	10000	16
1	1	10001	17
10	2	10010	18
11	3	10011	19
100	4	10100	20
101	5	10101	21
110	6	10110	22
111	7	10111	23
1000	8	11000	24
1001	9	11001	25
1010	10	11010	26
1011	11	11011	27
1100	12	11100	28
1101	13	11101	29
1110	14	11110	30
1111	15	11111	31

Representing numbers (II)

- Representing integers
  - Decimal integers are converted to binary integers
  - Given k bits, the largest *unsigned* integer is $2^k - 1$
    - Given 4 bits, the largest is $2^4-1 = 15$
    - Java obviously supports larger than 4-bit numbers
  - Signed integers must also represent the sign (positive or negative)
    - One bit is then used for the sign itself
    - Negative zero?

Representing numbers (III)

- Representing real numbers
  - First, convert into binary numbers
    - A little trickier than it first seems: to the right, each bit represents ½, ¼, 1/8, etc.
    - 5.75 = ?
  - Next, put into binary scientific notation: $a \times 2^b$
    - 101.11×2^0
  - Normalize so that first significant digit is immediately to the right of the binary point
    - $.10111 \times 2^3$
  - Mantissa and exponent (and signs) then stored
  - What's the ultimate result?

Next time

- Finish data representation
- Manipulating data in Java
- Start working on HW1 if you haven't already!