CMPT 166: Object-Oriented Programming Using Java

11 Jan 2011
CMPT166
Dr. Sean Ho
Trinity Western University

TRINITY WESTERN UNIVERSITY

What's on for today

- Languages: machine, assembly, high-level
- Java code translation
- JavaSE vs. JavaEE vs. JavaME
- JDK vs. JRE
- Principles of object-oriented languages
- A first Java program: “Hello, World!”
- Comments and doc-comments
- Compiling and running a Java program
- Java development environments

Review: Languages

- Machine language
  - "Native tongue" of computer (CPU, etc.)
  - Highly specific to machine (P4, Atom, ...)
- Assembly language
  - English-like abbreviations for operations
- High-level language
  - More "English-like" instructions (Common operations: arithmetic, I/O, etc.)
  - Compiler converts to machine language
- Interpreter: execute high-level w/o compile

Compiling: Python vs. C

- Python is an interpreted language:
  - When you press F5 in IDLE to run,
  - Code is first compiled into bytecode ***.pyc** files (e.g., compiled libs), then
  - Executed by the Python virtual machine
- C++ is a compiled language:
  - C++ compiler produces object files (*.o)
  - Linked together with libraries
  - To produce executable (*.exe)

Compiling: Java

- Edit: programmer writes program
  - IDE: Eclipse, NetBeans, plain-text editor, etc.
- Compile: compiler translates to bytecode
  - Machine-independent
- Load: class loader stores bytecodes in RAM
- Verify: check security (e.g., www)
- Execute: interpreter translates bytecodes into machine language

Where is Java used?

- Originally for consumer electronic devices
  - Popularized with web applets (client-side)
- JavaSE (Standard Edition): general use
- JavaEE (Enterprise Edition): application servers, e.g., web apps: fault-tolerant, distributed
- JavaME (Micro Edition):
  - Cell phones, Kindle, etc.
  - Point-of-sale (Sears, K-mart, Home Depot, ...)
  - Embedded Java: Ricoh copiers, Systronix for robotics, SunSPOT for remote sensors, etc.

Java kits: JDK vs. JRE

- JRE: Java Runtime Environment
  - Everything you need to run other people's compiled Java programs
  - Interpreter translates bytecode to machine language: java
- JDK: Java Development Kit
  - JRE plus everything you need to write your own Java programs
  - Compiler translates Java to bytecode: javac
- On Oracle's Java site or Savitch textbook CD

Object-oriented principles

- Alan Kay's Smalltalk (1980): very pure OO
  - Java: C-like, but designed OO from the start
- Five basic principles:
  - Everything is an object: w/ attribs, methods
  - A program is a set of objects passing messages
  - Each object has its own memory, storing other objects
  - Every object has a type (class)
  - All objects of the same type can receive the same messages

Java is object-oriented

- Everything is an object
  - Objects are instances of classes
- Write your program by defining classes:
  - Attributes (variables; data)
  - Methods (behaviour; functions)
  - Interfaces (collections of methods)
    - A class may implement more than one interface
    - An interface may be implemented by more than one class
- Can't have orphaned code outside of any class!

A first Java program

- (see HelloWorld.java in course directory)
- Rule of thumb is one public class per file.
  - Same name as the *.java file
  - Sometimes can have small helper classes within the file, too
- The main() method begins execution
  - Like C/C++
  - Declare it public and static, return type void
    - Public means other classes can see it
    - We'll get to public and other keywords later

Comments and doc-comments

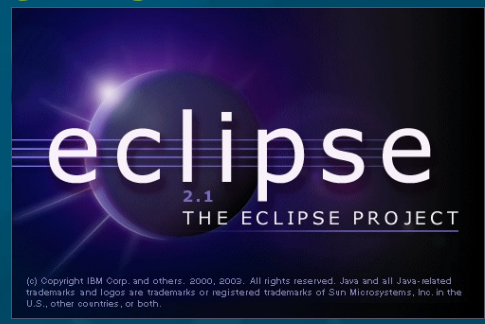
- Comments can either be
  - C-style: /* hi there! */
  - C++ - style: // hi there!
- Doc-comments start with /** (note two stars)
  - Structured comments: interpreted by javadoc
  - Similar to Python docstrings
  - @keywords: e.g., @author, @copyright
  - Pre/post-conditions: @param, @return
  - See JavaDoc webpage for more info

Compile and run

- Compile: javac HelloWorld.java
- Run: java HelloWorld
  - Use name of the class (HelloWorld), not the name of the file (HelloWorld.class)
- javac is standard JavaSE compiler from Sun
- Eclipse has its own incremental compiler, ecj
- Other compilers by IBM, gcj (gcc), Apple Xcode

Development environments

- Source code is just plain-text, all we need is a text editor and the compiler
- But integrated development environments (IDEs) make life easier
  - IDLE is a basic one for Python
  - MS Visual Studio is a very complex and expensive one for C++, C#, etc.
  - Eclipse is also sophisticated, and free
- Manage multiple projects, classes, and files
- Syntax highlighting, indent, auto-complete



Class policy on IDEs

- For class purposes, you may use any IDE you feel comfortable with, **but**:
- We must be able to re-compile & run your code!
  - There could be incompatibilities between compilers or versions of Java
  - We will be using Eclipse 3.6.1 (w/ ecj)
  - I also have Sun's JavaSE 1.6 on my laptop
- The only officially-supported setup is Eclipse 3.6.1 in the senior lab
  - *(demo Eclipse)*

CMPT166 programming labs

- CMPT166 is weighted heavily on programming labs (about 6 total)
- These are sizeable programming projects; allocate plenty of time to work on them!
- Individual work – you may discuss with your classmates, but your code should be your own
  - I'm open to team projects if you want, but the scope should expand accordingly
- Write-ups (see sample): design, libraries, variables, pseudocode(s), sample IO, test cases

TODO

- Lab0 (due next Tue): “Hello, World!”
  - Get familiar with a Java development environment: Eclipse, NetBeans, or other
  - Write a simple “Hello, World!” program
  - Nothing to turn in
- Lab1 (due in Thu 20 Jan): Control/Flow
  - Savitch text, pp.162-164. Choose one of:
  - #2: game of craps
  - #5: loan calculator
  - #8: cryptarithmetic puzzles