CSE 8B discussion: week 1

The JVM, packages, Unix-like environments, Vim, SSH, SCP
Agenda

1. Introduction
2. JVM
3. Packages
4. Linux and the shell
5. Editors
6. Customizing your environment
7. Working remotely
About CSE 8B

- This class builds on the Java basics you already know...
  - primitive types (int, double, boolean, char, byte, etc.) vs. reference types (objects, like Person, Dog, etc.), classes, if/else blocks, logical comparisons, ...

- ...and builds upon it!
  - inheritance, polymorphism, exception handling, concurrency, recursion, testing

- I’ll be a Java expert after taking this class--so what?
  - All other modern languages have identical or otherwise similar concepts
  - You may be learning how Java does these things, but you’re also simply getting extremely general, transferrable knowledge, too
  - Coding literacy can help in all kinds of work, automating tedious tasks, etc.; it’s not just for professional software devs
  - You'll have a deeper understanding of the stuff on which the world increasingly relies
  - ...or you can simply just be building yourself up as a more employable individual to make $$$
About me

● Spencer Wilson, from the east bay area
● 4th-year Computer Engineering undergrad
  (switched halfway through third year from Electrical Engineering)
● Interested in full-stack web development (finna be a JavaScript ninja); want to
  learn backend, UI, everything in between--it’s all important
● Other hobbies and interests
  ○ music listening and performance, playing bass (jazz, prog rock)
  ○ physics, neuroscience
  ○ personal finance
  ○ effective altruism (shoutout to PHIL 27)
  ○ soccer, volleyball
Agenda

1. Introduction
2. JVM
3. Packages
4. Linux and the shell
5. Editors
6. Customizing your environment
7. Working remotely
The Java Virtual Machine

- Unlike some other programming languages (e.g., C-style languages, assembly), compiled Java code can’t be executed directly by any real-world CPU
- Compiling a .java file with javac yields a .class file that contains bytecode, which is instructions for a virtual machine to execute the program
- Every time you run java <class>, you’re starting up an instance of the Java Virtual Machine (JVM), which then attempts to run the method called main in the given class
- This architecture enables Java code to be compiled once, and run on any platform for which the JVM is implemented (which is very many platforms)

Packages

- Java classes should be logically grouped into packages
  - ex.: a GameCube emulator written in Java might have packages named video, sound, input
- A class’s package is specified at the top of the file with the statement
  ```java
  package <package_name>;
  ```
- Members of a package (e.g., classes, enum types) are referenced with ‘.’
  - `<package_name>..<member_name>`, e.g., `video.FrameProcessor`, `input.AnalogStick`
Packages: our use

- For CSE 8B, a nice file structure for your work might be:
  ```
  cs8bxxx/
  hw1/
  Animate.java (with package hw1; on line 1)
  ...
  hw2/
  ...
  ```

- As you write code, your workflow (while in cs8bxxx) might look something like:
  - edit, compile, run, repeat
  - `vim hw1/Animate.java ; javac hw1/Animate.java ; java hw1.Animate`

https://docs.oracle.com/javase/tutorial/java/package/index.html
https://docs.oracle.com/javase/8/docs/technotes/tools/unix/java.html
Agenda

1. Introduction
2. JVM
3. Packages
4. Linux and the shell
5. Editors
6. Customizing your environment
7. Working remotely
Linux: it’s not an operating system

- Windows, OS X, Chrome OS, Android, iOS, and Ubuntu are operating systems
- In the 1970s, an OS called Unix was developed at Bell Labs in New Jersey, U.S.
  - Multiuser, multiprocess, just a terminal (no graphical user interface)
- What is an operating system?
  - A bundle of programs that includes everything a user needs to be productive on the system, plus the program that manages multitasking, interaction with hardware, and other low-level affairs
  - That program, being the central, most important part of an OS, is called the kernel
- Linux is a kernel, one that’s heavily inspired by Unix’s kernel, so it is called Unix-like; OSes inspired by Unix are also called this
- Free and open source, started in 1991 by 21 year-old Finnish undergraduate student Linus Torvalds (also later created Git in 2005)
  - “I'm doing a (free) operating system (just a hobby, won't be big and professional like gnu)”
Linux: it’s an operating system kernel

- Linux is a free kernel, woohoo. Are there OSes powered by Linux?
  - Yes, they’re called Linux distributions, and there are close to 500 in active development
  - Many are community-supported: Debian, Mint, Arch
  - Others are maintained by companies: Android, Chrome OS, Ubuntu, Fedora, CentOS
  - UCSD’s computer lab machines run CentOS

- If Linux is Unix-like, how do OSes powered by Linux have graphical UIs (GUIs)?
  - Modern desktop operating systems ship with a program called a windowing system that creates the “window” sort of user interface that is seen on most all desktop operating systems
  - So it’s just another layer of software running. Underneath the graphics is still a terminal, accessible via the operating system’s terminal application (e.g., Terminal, Command Prompt)

- That’s good. The terminal is so unpleasant, I h8 it, #stopterminal2k15
  - Naw, it can be pretty dope
“The terminal”: Distinguishing terminal from shell

- Goes by many names
  - Very sloppy equivalents: terminal, shell, command line, CLI (contrasted with GUI)

- More correctly,
  - **terminal**: the window that displays the text (manages font face/size/color, bg color, sound, panels or tabs inside the window)
  - **shell**: the program *running* in that window (determines what commands you can type at the prompt (the line that traditionally ends with ‘$’))

- Most Linux distributions, and OS X, come with a shell called Bash
  - OS X comes with Bash?! Yes. As a matter of fact, **OS X is a Unix-like OS. All of the basic shell commands you’ll learn (cd, ls, ssh, scp, etc.) work out of the box in OS X’s Terminal app.**
  - Windows’s Command Prompt is a singular terminal+shell, and uses different commands :
  - Bash is very scriptable, Command Prompt much less so (see Wikipedia: Unix philosophy)

The shell: it doubles as a toaster

- **Summary-analogy:** Unix-like systems come with a Swiss Army knife set of commands and programs that can be combined together to solve many problems in a few short commands
  - periodically run a command or script, search files for a string, do something with the files containing it, download things from the Internet, send email, start and stop a web server, chat in a chat room, **edit text files**, etc.—all doable, and automatable, on Unix-like systems

- **Holy learning curve, Batman!**
  - Learning of any kind requires active trial/error/thinking
  - The value usually far exceeds the up-front effort
  - A weaker reason to learn it: it's used in industry (for the reasons above)
Agenda

1. Introduction
2. JVM
3. Packages
4. Linux and the shell
5. Editors
6. Customizing your environment
7. Working remotely
Editors

- Source code files are simple text files, so you need a program to write text files
- Text editors, or simply “editors”, range in sophistication
  - Notepad, TextEdit, nano, Notepad++, Sublime Text, Vim, emacs, Eclipse, IntelliJ IDEA, etc.
- The fanciest editors are called IDEs: Integrated Development Environments
  - These will notify you immediately of compile-time errors, can be used to debug, and more
- Why would anyone not use an IDE?
  - They require a stronger computer to run, as well as a graphical environment
  - Using a GUI to do stuff can be convenient, but it also masks what’s really happening; the user cedes control to buttons, and as a result may sacrifice a fuller understanding of what they’re doing
    - aside: If you want to learn Git, do it via command line, NOT via SourceTree or other tools
  - With a bit of practice, one can be more productive in a CLI than in a GUI
Editors: Vim

- Vim (superceded vi in 1991, short for Vi IMproved) is a CLI editor that’s extremely customizable
- Two modes: Normal and Insert
  - In Normal mode (aka Command mode), keyboard buttons do some command
  - In Insert mode, each keyboard button just enters its respective character into the document
- Where speed comes from: cursor movement, cut/paste/delete, search/replace
  - quickly jump to line \( n \) with “\( :n \)”, top of the file with “\( gg \)”, enter Insert mode at the end of the current line with “\( A \)”, delete from current position to end of line with “\( D \)”, etc.
  - Run \texttt{vimgtutor} on the lab computers for an interactive introduction to the commands
- Eclipse users: get Vrapper, a plugin that adds Vim-like controls to Eclipse
- One downside: once you learn it, you’ll be upset when other editors don’t have a “Vim mode” option
1. Introduction
2. JVM
3. Packages
4. Linux and the shell
5. Editors
6. Customizing your environment
7. Working remotely
Customizing your environment

A couple things will make your coding experience a little smoother:

1. Get a better terminal than your OS’s stock terminal
2. Get Bash, Vim, and a vimrc file that works for you
Customizing: Terminal

- **OS X:** iTerm2: [https://www.iterm2.com/](https://www.iterm2.com/)
  - extremely easy text selection and search, multiple panes to the window, and more

- **Windows:** wait until the next slide
Customizing: Bash, Vim

- **OS X: comes with Bash**
  - open Terminal or iTerm2 and you’re sitting in a Bash shell

- **Windows: Git for Windows:** [https://git-for-windows.github.io/](https://git-for-windows.github.io/)
  - comes with Git Bash, a terminal that runs Bash and supports the basic shell commands
  - gives you a Unix-like environment on your Windows machine--much more usable than Command Prompt

- **Vim:** I think both OS X and Git Bash include it (not sure)

- **The Vim configuration file is located at ~/.vimrc**
  - the file, named .vimrc, is in your home directory. “Dotfiles” like these are often used for programs to store their settings in, and are typically hidden in `ls` output

- **$ cd ~; curl http://acsweb.ucsd.edu/~ssw021/.vimrc > .vimrc**
  - Do this on lab machines too, to get a usable Vim set up
Agenda

1. Introduction
2. JVM
3. Packages
4. Linux and the shell
5. Editors
6. Customizing your environment
7. Working remotely
Working remotely: SSH

- **Scenario:** the lab machines are in the CSE building basement; you are not
- **Fact:** we strongly recommend doing your assignments on the lab machines
  - they are graded on those machines
- **Fact:** you can access a lab machine shell from almost any computer
- **Secure Shell, or SSH,** is the program that enables this
  - **syntax:** `ssh <user>@<host>`
  - `$ ssh cs8bxxx@ieng6.ucsd.edu`
  - `ssh` opens a shell on a networked computer (e.g., over the Internet) on which you have an account
  - comes with OS X; need Git Bash on Windows (see previous slide)
Working remotely: SCP

- **Scenario:** you wish to flout your instructor’s recommendation and write code on your personal machine
- **Fact:** to submit your assignment, you need to get your files onto ieng6
- **Fact:** you can easily transfer files between your local computer and a remote computer
- **Secure Copy, or SCP, is the program that enables this**
  - Syntax: `scp <file [file...]> <user>@<host>:<destination_dir>`
  - `$ ls
  
  Thing.java
  
  $ scp Thing.java cs8bxxx@ieng6.ucsd.edu:~/hw1
  
  `scp` copies files to/from a networked computer on which you have an account
Agenda

1. Introduction
2. JVM
3. Packages
4. Linux and the shell
5. Editors
6. Customizing your environment
7. Working remotely
Oracle (the corporation that maintains the Java language) has Java tutorials that come up any time you google Java stuff
They’re very helpful, and I’ll reference them often once we get into more Java
  ○ https://docs.oracle.com/javase/tutorial/index.html
For a quick intro to shell scripting: http://learnxinyminutes.com/docs/bash/
For help with shell commands, see their manual pages: man <command>
Post on Piazza and tag your post with the “discussion” tag and I’ll check it out
Go see the other tutors during lab hours! http://cseweb.ucsd.edu/classes/fa15/cse8B-a/resources.html