Who am I?

Shaunak Pawagi

E-mail: shaunak at cs dot stonybrook.edu

Office Location: CS Room 1437

S11 Office hours: Mon Wed 4 pm – 5:30 pm and by appointment
Course Homepages

• http://www.cs.stonybrook.edu/~cse114
  – announcements
  – syllabus
  – schedule (lecture slides, hw, etc …)
  – etc.

• http://blackboard.stonybrook.edu
  – Grades
  – Your e-mail address on Blackboard
Registration Issues

• There are no seats left in lecture.

• We have 4 lab sections.
  – L01, L02, L03, and L04 (all taught in CS 2116)

• This course will be taught in summer 11.

• It will have 3 Lab sections (90 students).
What book do you need?

  by Y. Daniel Liang
  Published by Prentice Hall, 2011. ISBN: 0-13-213079-3
  Web address: [www.pearsonhighered.com](http://www.pearsonhighered.com)

Pre-requisites

- MAT 123 with a grade of C, or passing Math placement exam at level 4.
- CSE 110 is recommended, but not required.
- For some students CSE 114 is too much to start with. Perhaps it is a good idea to get some background from CSE 110.
Why Computer Science?

• Do what you love

• Computers can change the world

• There are lots of jobs
What do these things all have in common?

- Microsoft
- Cisco
- Netscape
- Google
- Facebook
- Apple
- Yahoo!
- Napster
- Dell
You’re in luck

• Because you’re in demand

• Read Newsday’s assessment “LI companies struggle to fill high-tech jobs”
What is this course about?

• An introduction to procedural and object-oriented programming methodology. Topics include program structure, conditional and iterative programming, procedures, arrays and records, object classes, encapsulation, information hiding, inheritance, polymorphism, file I/O, and exceptions. Software debugging and testing techniques are emphasized. Includes required laboratory.
And?

- Think of this as an opportunity to put Java programming *experience* on your resume

- When this semester is over, you will be a Java developer
**Major Course topics**

- **Procedural Programming Basics** - Variable declarations and data types, assignment statements & expressions, textual manipulation & strings, input/output, method construction, conditional (branching) statements, & iteration.

- **Object Oriented Programming** - Designing and constructing classes using containment, aggregation, inheritance, & polymorphism.

- **Arrays** - Using arrays to collect like data as well as searching, sorting, and other array manipulations.
What course work is involved?

- 6 Programming Assignments
- 1 Final Programming Project
- 2 Written Midterm Exams
- Coding or On-paper Final Exam
  - in CS 2129 or other rooms
How are grades computed?

- Lab Exercises (done in CS 2116): 10%
- CodeLab (online exercises): 5%
- 6 Programming Assignments: 25%
- 1 Final Programming Project: 10%
- 2 Midterm Exams: 30%
- Final Exam: 20%

These weights are approximate.
Lab Exercises

- Simple Coding Exercises done in CS sinc-site room CS 2116.
- You have only the lab-hour to edit, compile and execute your solution.
- Attendance is mandatory, if you want credit
- Demonstrate your work to Lab-TA before you leave
- Standard grading procedure
- 0 – 3 points
What is CodeLab?

- CodeLab is a set of online exercises
- It is a good learning tool for beginners
- There is a sign-up fee of $25 + transaction fee. The fee is nonrefundable. So do not pay until I tell you to do so.
- We will discuss this again when the first assignment is posted.
- There is a help document on the course webpage
What’s a final coding exam?

• If we decide to have it, you will each program solutions to problems on a computer.

• Programming problems (methods) related to HWs

• Programs will be graded based on performance testing
  – Ex: we will provide 10 test input to your solution
    • if 9/10 input produces correct output, you will receive 9/10 points for that question

• 0 points for when:
  – work that doesn’t compile
  – an infinite loop
  – your work never produces the correct output
Grading Process

- You must show competence on exams.
- Before letter grades are calculated, a preliminary curve will be made using only exam scores.
- This curve will be used to determine who is eligible to get a C.
- This does not mean you can pass this course just by doing exams.
Regrading of Homework/Exams

• You have one week from the day grades are posted or mailed or announced.
• Please meet with the person (either a TA or the instructor) and arrange for regrading.
• Late requests will not be entertained.
Disability

• If you have a physical, psychological, medical or learning disability, contact the DSS office at Room 128 ECC. Phone 632-6748/TDD.

• If you are planning to take an exam at DSS office, you need to tell me ahead of time for every exam. Otherwise you may not be able to take it there.

• All documentation of disability is confidential.
The Tentative HW & Exam Plan

• HW 1 – Strings & Basic I/O
• HW 2 – If Statements & Loops

MIDTERM EXAM 1
• HW 3 – Object Oriented Programming
• HW 4 – Inheritance & Polymorphism

MIDTERM EXAM 2
• HW 5 – Arrays
• HW 6 – Searching & Sorting
• Final Project

FINAL EXAM
What do you need to get started?

• MySBMail (and thus Sparky) account
  – Unix operating system
  – HW 1 will be handed in and tested on Sparky

• Blackboard account
  – http://blackboard.stonybrook.edu/
  – grades
Login to MySBMail

- See [https://tlt.stonybrook.edu/StudentServices/Email/Pages/default.aspx](https://tlt.stonybrook.edu/StudentServices/Email/Pages/default.aspx)

Make sure that your login-id (same as NetId) and the password (same as NetId password) are working. They will be set up in a few days for the whole class.

- You will need it for Lab #1 also for submitting programming assignments

- If you have any difficulty, go to Melville Library SINC Site for help

- To login: [http://mysbmail.stonybrook.edu/uwc/auth](http://mysbmail.stonybrook.edu/uwc/auth)
Java

• Java 2 v. 1.6 (a.k.a. Java 6)
  – For info on changes from Java 6:
    • http://java.sun.com/javase/6/webnotes/features.html

• Download

• API:
  – http://java.sun.com/javase/6/docs/api/

• The Java Tutorial:
What is Unix?

• We will use it for the early labs & programming assignments (HW)

• Sparky is the server name

• Lab 1 will teach you the important UNIX commands
Java IDE

• eclipse

• Download

• Learn to use the debugger (important)
Lab Facilities

- SINC Sites
  
  - Labs will start next week (Feb 10 or Feb 15) depending on when we get TAs

  - All Labs will be taught in CS SINC site (room 2116)
Uses of Java

• Applications programming
• Database programming using Sun’s jdbc library
• Internet programming to create dynamic web sites:
  – XML manipulation for data and documentation
  – Servlets and JSP technology
  – Applets technology
  – JavaBeans technology
• Java libraries exist for every high-level programming technology
  – Security, Networking, Multi-Threaded Applications, Graphics, Graphical User Interfaces (GUIs), etc…
Tools for Writing Java Programs

• 1\textsuperscript{st} Approach – the bare minimum
  – edit Java source code in text editor (ex: Notepad or Pico)
  – compile source code into class files from command line
  – can be tedious
  – poor interactivity

• 2\textsuperscript{nd} Approach – IDEs
  – Integrated Development Environment
  – combines writing, compiling, running and debugging Java code into a single application
  – eclipse, JBuilder, Visual Studio, etc.
  – makes coding much more efficient and organized
How does it work?

Java Source Code
• you write this
• .java files

Compiler Program
• part of eclipse IDE
• javac Unix command

Java Executable Code
• not readable
• .class files

Java Virtual Machine
• part of eclipse IDE
• java Unix command

Running Java Program
AN IMPORTANT NOTE ON ACADEMIC DISHONESTY

• All work you submit for homework, projects, or exams MUST be your own work.

• If you cheat or aid someone in cheating, you will automatically fail this course and be brought up on charges of academic dishonesty without warning.

• NO EXCEPTIONS WILL BE MADE!
The mystery of the disappearing student

• Why would a student not take an exam, but remain registered for a course?

• Why would a student not do any programming assignments, but remain registered for a course?
Rules of thumb to remember

• 3 ½ months is not a long time

• Points are more difficult to makeup later in the semester

• Have the same sense of urgency for the 1st assignment as you would for the 6th
A professional attending meetings

• Professionals:
  – go to boring meetings
  – go to boring conferences
  – go to boring lectures
  – are expected to learn and retain knowledge from these events

• BTW, hopefully you will reach the point in your professional career where you will look forward to attending lectures & conferences
Get the most out of your $(investment)$

• Do your best each class:
  – not to fall asleep
    • if you do, not to snore or drool
  – not to zone out
  – not to fall prey to “I’ll learn this later on my own”
  – to focus your brain on detail

• This course requires a great deal of attention to detail
And another thing

- Eat your veggies
- Be kind to children and animals
- Don’t eat and drive
- Just say no to drugs
- Get 8 hours of sleep each night
- Don’t punch old people
- Clean your room
- Don’t play with knives and/or matches
- Don’t play ball in the house
Our Daily Reminder

• You’re paying me to be here

• This material cannot be learned in one night, or even in one week
  – please give me your brain

• This is your opportunity to learn this stuff
  – as Computer Scientists, you’ll have many opportunities to use it
How do these magic machines work?

• Abstraction

• Abstraction

• Abstraction

http://www.solarnavigator.net/computers.htm
What's abstraction?

High-Level Language (HLL)

Assembly Language

Machine Language

Hardware
Hardware likes binary

- What’s binary?
  - a base-2 number system

- What do humans use?
  - base-10
  - Why?

- Why do computers like binary?
  - don’t be silly, computers don’t have feelings

http://upload.wikimedia.org/wikipedia/en/0/09/DataTNG.jpg
Computer *Designers* Like Binary

• Why?
  – it’s easier to make hardware that stores and processes binary numbers than decimal numbers
  – results are more efficient
    • space & cost

http://msp222.photobucket.comalbums/dd297/ponceje81/nerds.jpg
So Hardware stores 0s & 1s

- 0101010101010101010101010101 …

- Data is typically stored in 32 or 64 bit words

- Data is byte addressable
  - we can access or change any byte (group of 8 bits) independently as needed
So what data does the hardware store?

• Everything!
  – Text: 0101010101010101010101000100011111
  – Numbers: 0100001000101111101011010101101
  – Programs: 1110100010111101001101010101001
  – Images: 00100010101110100100101010100010
  – Etc.

• Programs?
  – we use stored program computers
Humans hate binary

• Imagine programming in binary,
  011010101001011010101010101010101101010

• Torture!
By the way, how do we store text?

- Numerically (using its code)

- Each character is stored in memory as a number

- When it’s time to display, draw the character differently depending on it’s numeric value

- NOTE: the OS or program needs to know how to display each character
ASCII & Unicode

• Standard character sets

• ASCII uses 1 byte per character
  – How many different ASCII characters are there?

• Unicode uses 2 bytes per character
  – How many are there?

• Ex, in both, ‘A’ is 65
<table>
<thead>
<tr>
<th>Dec</th>
<th>Hx</th>
<th>Oct</th>
<th>Char</th>
<th>Dec</th>
<th>Hx</th>
<th>Oct</th>
<th>Html</th>
<th>Chr</th>
<th>Dec</th>
<th>Hx</th>
<th>Oct</th>
<th>Html</th>
<th>Chr</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>000 NUL (null)</td>
<td>32</td>
<td>20</td>
<td>040</td>
<td>&lt;#32;&gt;</td>
<td>Space</td>
<td>64</td>
<td>40</td>
<td>100</td>
<td>&lt;#64;&gt;</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>001</td>
<td>SOH (start of heading)</td>
<td>33</td>
<td>21</td>
<td>041</td>
<td>&lt;#33;&gt;</td>
<td>!</td>
<td>65</td>
<td>41</td>
<td>101</td>
<td>&lt;#65;&gt;</td>
<td>A</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>002</td>
<td>STX (start of text)</td>
<td>34</td>
<td>22</td>
<td>042</td>
<td>&lt;#34;&gt;</td>
<td>&quot;</td>
<td>66</td>
<td>42</td>
<td>102</td>
<td>&lt;#66;&gt;</td>
<td>B</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>003</td>
<td>ETX (end of text)</td>
<td>35</td>
<td>23</td>
<td>043</td>
<td>&lt;#35;&gt;</td>
<td>#</td>
<td>67</td>
<td>43</td>
<td>103</td>
<td>&lt;#67;&gt;</td>
<td>C</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>004</td>
<td>EOT (end of transmission)</td>
<td>36</td>
<td>24</td>
<td>044</td>
<td>&lt;#36;&gt;</td>
<td>$</td>
<td>68</td>
<td>44</td>
<td>104</td>
<td>&lt;#68;&gt;</td>
<td>D</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>005</td>
<td>ENQ (enquiry)</td>
<td>37</td>
<td>25</td>
<td>045</td>
<td>&lt;#37;&gt;</td>
<td>%</td>
<td>69</td>
<td>45</td>
<td>105</td>
<td>&lt;#69;&gt;</td>
<td>E</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>006</td>
<td>ACK (acknowledge)</td>
<td>38</td>
<td>26</td>
<td>046</td>
<td>&lt;#38;&gt;</td>
<td>&amp;</td>
<td>70</td>
<td>46</td>
<td>106</td>
<td>&lt;#70;&gt;</td>
<td>F</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>007</td>
<td>BEL (bell)</td>
<td>39</td>
<td>27</td>
<td>047</td>
<td>&lt;#39;&gt;</td>
<td>'</td>
<td>71</td>
<td>47</td>
<td>107</td>
<td>&lt;#71;&gt;</td>
<td>G</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>010</td>
<td>BS (backspace)</td>
<td>40</td>
<td>28</td>
<td>050</td>
<td>&lt;#40;&gt;</td>
<td>(</td>
<td>72</td>
<td>48</td>
<td>110</td>
<td>&lt;#72;&gt;</td>
<td>H</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>011</td>
<td>TAB (horizontal tab)</td>
<td>41</td>
<td>29</td>
<td>051</td>
<td>&lt;#41;&gt;</td>
<td>)</td>
<td>73</td>
<td>49</td>
<td>111</td>
<td>&lt;#73;&gt;</td>
<td>I</td>
</tr>
<tr>
<td>10</td>
<td>A</td>
<td>012</td>
<td>LF (NL line feed, new line)</td>
<td>42</td>
<td>2A</td>
<td>052</td>
<td>&lt;#42;&gt;</td>
<td>*</td>
<td>74</td>
<td>4A</td>
<td>112</td>
<td>&lt;#74;&gt;</td>
<td>J</td>
</tr>
<tr>
<td>11</td>
<td>B</td>
<td>013</td>
<td>VT (vertical tab)</td>
<td>43</td>
<td>2B</td>
<td>053</td>
<td>&lt;#43;&gt;</td>
<td>+</td>
<td>75</td>
<td>4B</td>
<td>113</td>
<td>&lt;#75;&gt;</td>
<td>K</td>
</tr>
<tr>
<td>12</td>
<td>C</td>
<td>014</td>
<td>FF (NP form feed, new page)</td>
<td>44</td>
<td>2C</td>
<td>054</td>
<td>&lt;#44;&gt;</td>
<td>,</td>
<td>76</td>
<td>4C</td>
<td>114</td>
<td>&lt;#76;&gt;</td>
<td>L</td>
</tr>
<tr>
<td>13</td>
<td>D</td>
<td>015</td>
<td>CR (carriage return)</td>
<td>45</td>
<td>2D</td>
<td>055</td>
<td>&lt;#45;&gt;</td>
<td>-</td>
<td>77</td>
<td>4D</td>
<td>115</td>
<td>&lt;#77;&gt;</td>
<td>M</td>
</tr>
<tr>
<td>14</td>
<td>E</td>
<td>016</td>
<td>SO (shift out)</td>
<td>46</td>
<td>2E</td>
<td>056</td>
<td>&lt;#46;&gt;</td>
<td>.</td>
<td>78</td>
<td>4E</td>
<td>116</td>
<td>&lt;#78;&gt;</td>
<td>N</td>
</tr>
<tr>
<td>15</td>
<td>F</td>
<td>017</td>
<td>SI (shift in)</td>
<td>47</td>
<td>2F</td>
<td>057</td>
<td>&lt;#47;&gt;</td>
<td>/</td>
<td>79</td>
<td>4F</td>
<td>117</td>
<td>&lt;#79;&gt;</td>
<td>O</td>
</tr>
<tr>
<td>16</td>
<td>G</td>
<td>020</td>
<td>DLE (data link escape)</td>
<td>48</td>
<td>30</td>
<td>060</td>
<td>&lt;#48;&gt;</td>
<td>0</td>
<td>80</td>
<td>50</td>
<td>120</td>
<td>&lt;#80;&gt;</td>
<td>P</td>
</tr>
<tr>
<td>17</td>
<td>H</td>
<td>021</td>
<td>DC1 (device control 1)</td>
<td>49</td>
<td>31</td>
<td>061</td>
<td>&lt;#49;&gt;</td>
<td>1</td>
<td>81</td>
<td>51</td>
<td>121</td>
<td>&lt;#81;&gt;</td>
<td>Q</td>
</tr>
<tr>
<td>18</td>
<td>I</td>
<td>022</td>
<td>DC2 (device control 2)</td>
<td>50</td>
<td>32</td>
<td>062</td>
<td>&lt;#50;&gt;</td>
<td>2</td>
<td>82</td>
<td>52</td>
<td>122</td>
<td>&lt;#82;&gt;</td>
<td>R</td>
</tr>
<tr>
<td>19</td>
<td>J</td>
<td>023</td>
<td>DC3 (device control 3)</td>
<td>51</td>
<td>33</td>
<td>063</td>
<td>&lt;#51;&gt;</td>
<td>3</td>
<td>83</td>
<td>53</td>
<td>123</td>
<td>&lt;#83;&gt;</td>
<td>S</td>
</tr>
<tr>
<td>20</td>
<td>K</td>
<td>024</td>
<td>DC4 (device control 4)</td>
<td>52</td>
<td>34</td>
<td>064</td>
<td>&lt;#52;&gt;</td>
<td>4</td>
<td>84</td>
<td>54</td>
<td>124</td>
<td>&lt;#84;&gt;</td>
<td>T</td>
</tr>
<tr>
<td>21</td>
<td>L</td>
<td>025</td>
<td>NAK (negative acknowledge)</td>
<td>53</td>
<td>35</td>
<td>065</td>
<td>&lt;#53;&gt;</td>
<td>5</td>
<td>85</td>
<td>55</td>
<td>125</td>
<td>&lt;#85;&gt;</td>
<td>U</td>
</tr>
<tr>
<td>22</td>
<td>M</td>
<td>026</td>
<td>SYN (synchronous idle)</td>
<td>54</td>
<td>36</td>
<td>066</td>
<td>&lt;#54;&gt;</td>
<td>6</td>
<td>86</td>
<td>56</td>
<td>126</td>
<td>&lt;#86;&gt;</td>
<td>V</td>
</tr>
<tr>
<td>23</td>
<td>N</td>
<td>027</td>
<td>ETB (end of trans. block)</td>
<td>55</td>
<td>37</td>
<td>067</td>
<td>&lt;#55;&gt;</td>
<td>7</td>
<td>87</td>
<td>57</td>
<td>127</td>
<td>&lt;#87;&gt;</td>
<td>W</td>
</tr>
<tr>
<td>24</td>
<td>O</td>
<td>030</td>
<td>CAN (cancel)</td>
<td>56</td>
<td>38</td>
<td>070</td>
<td>&lt;#56;&gt;</td>
<td>8</td>
<td>88</td>
<td>58</td>
<td>130</td>
<td>&lt;#88;&gt;</td>
<td>X</td>
</tr>
<tr>
<td>25</td>
<td>P</td>
<td>031</td>
<td>EM (end of medium)</td>
<td>57</td>
<td>39</td>
<td>071</td>
<td>&lt;#57;&gt;</td>
<td>9</td>
<td>89</td>
<td>59</td>
<td>131</td>
<td>&lt;#89;&gt;</td>
<td>Y</td>
</tr>
<tr>
<td>26</td>
<td>Q</td>
<td>032</td>
<td>SUB (substitute)</td>
<td>58</td>
<td>3A</td>
<td>072</td>
<td>&lt;#58;&gt;</td>
<td>;</td>
<td>90</td>
<td>5A</td>
<td>132</td>
<td>&lt;#90;&gt;</td>
<td>Z</td>
</tr>
<tr>
<td>27</td>
<td>R</td>
<td>033</td>
<td>ESC (escape)</td>
<td>59</td>
<td>3B</td>
<td>073</td>
<td>&lt;#59;&gt;</td>
<td>&lt;</td>
<td>91</td>
<td>5B</td>
<td>133</td>
<td>&lt;#91;&gt;</td>
<td>[</td>
</tr>
<tr>
<td>28</td>
<td>S</td>
<td>034</td>
<td>FS (file separator)</td>
<td>60</td>
<td>3C</td>
<td>074</td>
<td>&lt;#60;&gt;</td>
<td>&lt;&lt;</td>
<td>92</td>
<td>5C</td>
<td>134</td>
<td>&lt;#92;&gt;</td>
<td>\</td>
</tr>
<tr>
<td>29</td>
<td>T</td>
<td>035</td>
<td>GS (group separator)</td>
<td>61</td>
<td>3D</td>
<td>075</td>
<td>&lt;#61;&gt;</td>
<td>=</td>
<td>93</td>
<td>5D</td>
<td>135</td>
<td>&lt;#93;&gt;</td>
<td>]</td>
</tr>
<tr>
<td>30</td>
<td>U</td>
<td>036</td>
<td>RS (record separator)</td>
<td>62</td>
<td>3E</td>
<td>076</td>
<td>&lt;#62;&gt;</td>
<td>&gt;</td>
<td>94</td>
<td>5E</td>
<td>136</td>
<td>&lt;#94;&gt;</td>
<td>{</td>
</tr>
<tr>
<td>31</td>
<td>V</td>
<td>037</td>
<td>US (unit separator)</td>
<td>63</td>
<td>3F</td>
<td>077</td>
<td>&lt;#63;&gt;</td>
<td>?</td>
<td>95</td>
<td>5F</td>
<td>137</td>
<td>&lt;#95;&gt;</td>
<td>_</td>
</tr>
</tbody>
</table>
How about a Unicode Table?

• Won’t fit on a single slide of course

• Try http://www.tamasoft.co.jp/en/general-info/unicode.html
An improvement: Assembly

• Early computers were programmed numerically
  – in binary or decimal or otherwise

• Problems?
  – difficult
  – slow
  – error prone

• Solution?
  – write code in a more human-friendly format
  – enter the code as text into the computer
  – have a program turn the text into binary instructions
hello.asm – an assembly program

## hello.asm - print out "hello world"

# text segment #
.text # tells assembler program code starts here
.globl main # defines label for execution start
main: # execution starts here
la $a0,str # put string address into a0
li $v0,4 # system call to print
syscall # out a string
li $v0, 10 # Load exit syscall value
syscall # Exit

# data segment #
.data # tells assembler data segment begins here
str: .asciiz "hello world\n" # declaration of a string
What do you think of assembly?

- Far easier to use than binary

- Still not very user friendly. Why?
  - very low-level operations
  - programming is time consuming

- BTW, anyone know what type of program converts an assembly program into binary?
  - an assembler
Solution to dreary assembly programming?

• **Abstraction**
  – add another layer of programming
  – high level programming languages (HLL)

• **HLLs should be:**
  – more user friendly, easy to use
  – more flexible
  – platform independent
High level languages

- Lots and lots and lots of them
  - remember, they are platform independent

- C, C++, Java, Pascal, Ada, Perl, Python, BASIC, Fortran, COBOL, Lisp, etc.

- HLLs are supposed to be relatively easier for humans to write and understand right?

- So let’s do a comparison
```c
#include <stdio.h>

int main(void)
{
    printf("Hello, world!\n");
    return 0;
}
```
hello.asm

.text
.globl main
main:
la $a0,str
li $v0,4
syscall
li $v0, 10
syscall

.data
str: .asciiz "hello world\n"
Assembly code translated into binary code

00111100000000010001000000000001
00110100001001000000000000000000
00100000000000100000000000000100
000000000000000000000000001100
00100000000000100000000000001010
000000000000000000000000001100
What’s a compiler?

• A software program

• Input: High Level Language source code
• Output: Assembly Code

• It is typically integrated with an assembly
  – together they can make an executable or binary program
Java is somewhat different, however

• Why?

• Java has a principle, “write once, run anywhere”
  – What does that mean?
    • Platform independence for compiled Java code
  – How?
    • The Java Virtual Machine

• Java programs are compiled into Java bytecode
  – bytecode is executed by the Java Virtual Machine
Java Virtual Machine

• A program that runs Java programs
  – also manages memory for Java programs

• Why?
  – each platform is different (Mac/PC/Linux/etc.)
  – the JVM is a layer between a compiled Java program (in bytecode)

• BTW, the Operating System is a layer in between the JVM and the hardware
So where do we come in?

• Computer Engineering
  – creating hardware

• Computer Science
  – creating tools in high level programming languages

• Information Systems
  – using tools effectively for business
Integrated Development Environment (IDE)

- A tool for making programs
  - edit HLL source code
  - compile
  - test
  - deploy

- We’ll use eclipse
What is memory?

• Think of it as a giant array

• How do we assign data to/get data from memory?
  – memory addresses
  – these are indices into the memory array
    • addresses are typically byte addresses
What goes in each memory segment?

- **Stack Segment**
  - temporary variables declared inside methods
  - removed from memory when a method returns

- **Heap Segment**
  - for dynamic data (whenever you use new)
  - data for constructed objects
  - persistent as long as an existing object variable references this region of memory

- **Global Segment**
  - data that can be reserved at compile time
  - global data (like static data)
Where is our data being stored?

```java
public class PhoneListing {
    public int number = 0;
    public String name = "NONE";
    public PhoneListing() {}

    public static void main(String[] args) {
        test1();
    }

    public static void test1() {
        PhoneListing test = new PhoneListing();
        test.number = 8675309;
        test.name = new String("Jenny");
        printDetails(test);
    }

    public static void printDetails(PhoneListing d) {
        System.out.println("Call " + d.name + " at " + d.number);
    }
}
```

Where will test be stored?

How much memory will be reserved?
Why do we care about this?

- You must understand that in Java, every object variable stores a memory addresses
  - 32 bit numbers (4 bytes)

- These addresses point to memory locations where the objects’ data is stored

- This is important to understand proper data manipulation
Next Time

- Java
- Java
- And more Java