Linfield College, Computer Science Department
COMP 161: Basic Programming
Syllabus Spring 2016
Prof. Daniel Ford

COURSE DESCRIPTION:
Introduces the basic concepts of programming: reading and writing unambiguous descriptions of sequential processes. Emphasizes introductory algorithmic strategies and corresponding structures. (QR). COMP 160 is required for a major in electronic arts and is a prerequisite for COMP 161 and all other required courses for a major or minor in computer science. Extensive hands-on experience with programming projects. Lecture/discussion, lab.

PREREQUISITES:
Prerequisite: COMP 160.

CREDITS: 3

LECTURES: Renshaw 211
Section 1: MWF 2:00 – 2:50 PM

WORKSHOPS: Renshaw 211
Workshop: TBD
Additional help and tutoring is available by request and through the learning center.

PROFESSOR: Daniel Ford
Office: Renshaw 210 Phone: x2706 E-mail: ford@linfield.edu
Office Hours: MWF 10:30—11:30 AM, TTh 1:30—2:30 PM, and by appointment.

COURSE OBJECTIVES:
- Practice decomposing complex algorithmic problems into smaller simpler problems.
- Practice solving problems via iterating over *strings* and *arrays*.
- Practice *tracing* and *debugging* programs.
- Practice testing programs, e.g. via JUnit tests.
- Practice writing programs with *style.*
- Practice researching programming solutions.

RECOMMENDED BOOKS:
- **Cay S. Horstmann,** Big Java, http://www.horstmann.com/bigjava.html. Recommended for COMP 160 & 161. Wiley 5th Edition (2012 Early Objects) ~ $90, 4th Edition, 3rd Edition, …, 2nd Edition (2005) ~ $10.
- **Y. Daniel Liang**, Introduction to Java Programming, Comprehensive Version, http://cs.armstrong.edu/liang/intro9e/. Recommended for COMP 160 & 161. Faster, more advanced, but with fewer tips than Big Java. 9th Edition (2012) ~ $90, 8th Edition, …, 6th Edition (2006) ~$10.
- **Cay S. Horstmann & Gary Cornell,** Core Java™, *Volume I – Fundamentals* and *Volume II – Advanced Features* (9th Edition), 2012 http://www.horstmann.com/corejava.html. The best reference manuals for Java. Recommended for CS majors and professional Java programmers. $37 each.

RECOMMENDED (FREE) WEB RESOURCES:
Java
- **CodingBat.com**. http://www.codingbat.com/
- **Java 6 API**. http://java.sun.com/javase/6/docs/api/
- **Liang's Java Debug Common Errors and Answers** http://cs.armstrong.edu/liang/intro9e/debug.html
- **Horstmann's Worked Examples** http://bcs.wiley.com/he-bcs/Books?action=resource&bcsId=7872&itemId=1118431111&resourceId=30501
- **Horstmann's Sample Programs** http://horstmann.com/bigj5/bigjava.zip
- **Horstmann's Animations** http://bcs.wiley.com/he-bcs/Books?action=resource&bcsId=7872&itemId=1118431111&resourceId=31273

- **The Java Tutorials** http://java.sun.com/docs/books/tutorial/index.html
- **Liang's Algorithms and Data Structure Animations** in JavaScript and Processing http://cs.armstrong.edu/liang/animation/animation.html

JavaScript and Processing

- **Kahn Academy Programming** https://www.khanacademy.org/cs/programming
- **Processing.org (Reference** http://processing.org/reference/, **Tutorials** http://processing.org/tutorials/, **and Examples** http://processing.org/examples/.)
- **Sketchpad.cc** http://sketchpad.cc/ sketchpad is built on processing.js and etherpad http://etherpad.org/ a handy tool for collaboration.)
- **Has Canvas** http://hascanvas.com/
- **Open Processing** http://www.openprocessing.org/
- **Fun Programming** http://www.funprogramming.org/ (not a fan of the videos, but the organized list of 128 simple examples on how to do various things is excellent!!)
- **Proclipsing (processing in Eclipse)** https://code.google.com/p/proclipsing/
- **Codecademy** http://www.codecademy.com/
- **AppendTo JavaScript Video Lessons** http://learn.appendto.com/
- **LearnStreet Beginner JavaScript Course V3** http://www.learnstreet.com/lessons/study/javascript
- **W3Schools JavaScript Introduction** http://www.w3schools.com/js/js_intro.asp
- **Mozilla Development Network JavaScript Guide** https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide
- **Cheatography JavaScript Cheat Sheet** http://www.cheatography.com/davechild/cheat-sheets/javascript/

Android

- **Bill Phillips and Brian Hardy** Android Programming: The Big Nerd Ranch Guide http://www.bignerdranch.com/book/android_the_big_nerd_ranch_guide ($29 with shipping on Amazon 2/14)
- **Shawn Van Every** http://itp.nyu.edu/~sve204/mobilemedia_spring10/android101.pdf and http://itp.nyu.edu/~sve204/mobilemedia_spring10/syllabus.html
- **Mark L. Murphy**, The Busy Coder's Guide to Android Development, (Version 3.6 – 2008 – Android SDK 3.0 – free) http://commonsware.com/Android/Android_3-6-CC.pdf (version 4.5 - 2012 - Android SDK 4.2 - $45) http://commonsware.com/Android/.
- **Reto Meier**, Professional Android 4 Application Development, 2012 Wrox, **ISBN** 978-1118102275.
- **Paul Deitel, Harvey Deitel, and Abbey Deitel** Android How to Program with an Introduction to Java. Prentice Hall 2012 **ISBN** 978-0132990547.
- **Android Developer Training, Tools/Samples & API**, http://developer.android.com/sdk/index.html.
- **Lars Vogel Tutorials** http://www.vogella.com/.

Mobile Development

- http://buzztouch.com/

git Version Control System

- Download:
  - http://git-scm.com/downloads
  - https://help.github.com/articles/set-up-git
- Help / tutorials :
  - https://www.codeschool.com/courses/try-git
  - https://help.github.com/
  - http://git-scm.com/book/en/Getting-Started
  - http://www.codecademy.com/blog/74-getting-started-with-git

GRADES:

Grading for this course will be based on: (I thought I talked to Jacob about weekly mini quizzes?)

Homework & Project	~50%
Exams & Quizzes	~50%

MAKE UP QUIZZES AND EXAMS

Make-up quizzes and exams will not be granted without prior arrangement or for reasons stated in the college catalog.

LABS/WORKSHOPS:
This class will have weekly workshops run by students on a day and time to be decided. These are not mandatory, but are highly recommended especially if you find it challenging to finish all of the weekly programming problems. Extra-credit will be given for attendance.
Many students have claimed that the workshops were even more helpful then the lectures.

HOMEWORK PROBLEMS
Weekly homework will be due Monday at Midnight.

PRACTICE, DEBUGGING, AND LEARNING ON YOUR OWN:
Learning programming is a bit like learning a sport: I will guide your efforts to help them to be as productive and enjoyable as possible, but most of your progress will come from practice and learning on your own.
A major part of this process is debugging, i.e. generating and implementing ideas to investigate and determine the cause of problems, a.k.a. bugs, as well as ideas to generate, test, evaluate and select solutions to fix these problems. These debugging ideas are implicit in any solution you submit for credit.
The purpose of having credit for the course based on problem solving and debugging skills is that repeated practice in generating and evaluating solutions is the best way to increase your programming and problem solving skills.

LINFIELD CURRICULUM:
In order to earn a QR for this course, you must submit relevant exemplars of your work to TaskStream by the last day of finals week, as discussed in the Linfield College Course Catalog, pages 6-8.

This course contributes to the Linfield Curriculum in the area of *Quantitative Reasoning* (QR). Courses with this designation develop the student's ability to:

1. *Frame contextual questions using mathematical representation.*

When programming, we often ask the following question, "What sequence of instructions would accomplish our goal?"
Programming frames real world questions about achieving a goal first into a desired quantifiable outcome and then creates a hypothesis, i.e. a program or sequence of logical instructions, on how to achieve that outcome. Programming involves repeatedly testing these hypotheses and observing the result. Any difference between the result and the desired outcome is generally the result of faulty deductive reasoning.

2. *Apply models to deduce consequences and make predictions.*

When programming, the best question is often, "What would be the result of executing the following sequence of instructions? "
In order to better understand our deductive reasoning errors, improve our deductive reasoning skill, and debug/improve our program/hypothesis, programming involves repeatedly making predictions about the consequence of executing a sequence of logical instructions and then testing these predictions/hypotheses by writing a program consisting of these instructions, executing the programs and observing the results. During this process the programming language, e.g. Java, and related abstractions, e.g. data structures and algorithms, act as a model. We use our understanding of this model to predict the consequences of executing specific sequences of instructions. Note the similarity between the scientific method, which explores questions by forming, testing, and modifying hypotheses, and programming.

3. *Verbally communicate and critique quantitative arguments.*

It is often much easier for others to spot our mistakes than it is for us to spot them, and sometimes simply explaining something to someone else will help us realize errors in our logic. As social and fallible creatures we humans often find it enjoyable, enlightening, and better to program with others, and this benefit increases with practice. Moreover, real world projects are almost always so large that working in groups is unavoidable.

ILLNESS/MEDICAL POLICY
In the unfortunate event that you must miss class you are encouraged to contact your workshop partner to find out what we worked on in class and to pick up any materials that were handed out. In general, homework extensions and make-up exams will be granted as needed on an individual basis.

DISABILITY SUPPORT:
Students with disabilities are protected by the Americans with Disabilities Act and Section 504 of the Rehabilitation Act. If you are a student with a disability and feel you may require academic accommodations please contact Learning Support Services (LSS), as early as possible to request accommodation for your disability. The timeliness of your request will allow LSS to promptly arrange the details of your support. LSS is located in Melrose Hall 020 (503-883-2562). We also encourage students to communicate with faculty about their accommodations.

ACADEMIC CONDUCT:
In this course we will adhere to the college policy on academic honesty, as published in the Linfield College Course Catalog. Please review this policy if you are not already familiar with it.
Linfield College operates under the assumption that all students are honest and ethical in the way they conduct their personal and scholastic lives. Academic work is evaluated on the assumption that the work presented is the student's own, unless designated otherwise. Anything less is unacceptable and is considered a violation of academic integrity. Furthermore, a breach of academic integrity will have concrete consequences that may include failing a particular course or even dismissal from the college.
Violations of academic integrity include but are not limited to the following:

- *Cheating*: Using or attempting to use unauthorized sources, materials, information, or study aids in any submitted academic work.
- *Plagiarism:* Submission of academic work that includes material copied or paraphrased from published or unpublished sources without proper documentation. This includes self-plagiarism, the submission of work created by the student for another class unless he or she receives consent from both instructors.
- *Fabrication*: Deliberate falsification or invention of any information, data, or citation in academic work.
- *Facilitating Academic Dishonesty*: Knowingly helping or attempting to help another to violate the college's policy on academic integrity.

Note that plagiarism is claiming that someone else's work or ideas are your own. Copying or using someone else's work or ideas is not plagiarism unless you claim that the work or ideas are your own. By submitting academic work you are implicitly claiming, unless you explicitly state otherwise, that the submitted work and the ideas presented in it are your own.
In this class you will earn credit by combining and adapting basic building blocks, e.g. common data structures and algorithms to solve a particular problem or accomplish a specific task. You are expected to generate, evaluate, and illustrate your own ideas on how these building blocks can be put together to solve particular problems. In order to avoid plagiarizing,

- Obey the 30 second long-term memory rule. We all get stuck now and again. If you get help on a homework problem, e.g. in the form of looking at someone else's code, don't just implement their solution. Instead take a minute to think about their idea why it works or doesn't work and how you might improve or customize it before writing your own solution.
- Always add comments to explain any non-trivial code you submit, and never submit code that uses a solution that you don't completely understand.