Last updated 24 March, 2020

CS162 – Introduction to Computer Science II
Spring 2020, CRN: 41720, Credits: 4

Instructor: Joseph Jess
Web: http://cf.linnbenton.edu/bcs/cs/jessj/web.cfm?pgID=5404
email: jessj@linnbenton.edu

Initial class note: This may be an introduction course to computer science (CS), but that does not mean it will always be easy. This is a continuation of the programming design, testing, and implementation concepts using the Python programming language as a tool. You should also expect to do some reading, much practicing and searching, and much discussion of the topics we cover.

1. LBCC catalog course description, including pre-requisites/co-requisites:
Covers software engineering principles, basic data structures and abstract data types (arrays, strings, array-list and graphics). Introduces analysis of algorithms, testing, sorting and searching. Expands on Graphical User Interfaces, Swing components, layout managers and event driven programming. Also covers polymorphism, inheritance, recursion and exceptions. The Java programming language is used.

Official Prerequisite: CS 161 Introduction to Computer Science I with a grade of "C" or better.

2. Class Time-space:
1.1 **Note: spring 2020 will be delivered completely remotely due to local and state rules.**
2.1 Lecture + demo + lab: MWF 1000 – 1150, MKH101

3. Course Objectives:
Upon successful completion of this course, students will be able to:
3.1 Write and debug object-oriented code using class dependencies of aggregation and inheritance.
3.2 Write object-oriented code that includes the use of two-dimensional collections of data and objects.
3.3 Demonstrate the use of various layout managers and listeners in an event-driven programming paradigm.
3.4 Demonstrate the use of polymorphism through inheritance and draw Unified Modeling Language (UML) diagrams of class hierarchies.
3.5 Write code to sort and search one-dimensional collections.
3.6 Write code that detects and handles exceptions.
3.7 Demonstrate an understanding of recursion, contrast it to iteration, and select the best approach for a given problem.

4. Learning resources:
4.1 **Note:** All class materials and storage will be freely available in a digital format
4.2 The Python language – available through several media. We will discuss this some in class
4.3 A Python interpreter – I will use the interpreter available from https://www.python.org/. We can discuss this more in class
4.4 The integrated development environment (IDE) recommended in our past Python courses is possibly Vidle, I will plan to use VS Code or a plain text editor with some scripts (likely Notepad++ if I have my way)
    4.4.1 We will discuss some capabilities of smart code editors during the course.
4.5(**strongly recommended**) A desire to learn, experiment, design, test, and problem solve with code (both on and off of a computer).

5. Other Learning resources that you might consider using:
(these are not required, but you might find helpful, along with many other resources we could talk about in class)

5.1 Python Programming: An Introduction to Computer Science (Amazon link)
Author: John Zelle
ISBN: 978-1-59028-275-5

5.2 Practical Programming (Amazon link)
Authors: Paul Gries, Jennifer Campbell, Jason Montojo
ISBN: 978-1-68050-268-8

6. Grading:
6.1 Scores for coursework items will be initially available when demonstrated to the instructor. We will keep a spreadsheet in a shared folder through our student email account using Google Drive. We will discuss this in class, including how to access it and keep yourself organized (which may affect your grade).

6.1.1 My favored grading method is to have people show me where they are throughout the week so we can discuss approaches and any requirements that would affect the grade, which also allows for better understanding of requirements.

6.2 Students will be required to turn in all coursework items before 23:59 (Pacific Time Zone) on the date that they are due (generally Monday in my courses).

6.2.1 Students must be sure to give themselves plenty of time to submit coursework, as late work will not be accepted without prior consent or special circumstances.

6.3 To earn a passing grade in this course you must pass each of the following coursework categories:

6.3.1 **Demonstration**: Discussion and weekly assignments – 50%

6.3.1.1 There are a number of projects to be completed for this class, designed to challenge and solidify design, coding, and testing skills.

6.3.1.2 Project components are generally graded based on:

6.3.1.2.1 completeness (does it compile and run)

6.3.1.2.2 correctness (does it meet the listed requirements)

6.3.1.2.3 quality and explanation of the design (features in advance, organized)

6.3.1.2.4 quality and explanation of the tests (test each feature and expected successes and failures)

6.3.1.2.5 quality of the implementation (consistent and readable style, runs well, is easy to learn and use)

6.3.1.2.6 **Note**: careful design, systematic testing, consistent style, and readability of code are important software quality factors (all of which are subject to interpretation but graded by the instructor based on the spirit and letter of the requirements, so be sure to explain your decisions).

6.3.1.2.7 **Note well**: Your submission should be explained and able to be compiled and run from just your submitted files in your final submission. This means that you need to include any files provided to you that are necessary for your project to compile or run.

6.3.1.2.8 **Note very well**: Source code and related documents submitted must be designed and implemented by the student submitting the work and any code must compile and run on one of the instructor's machines in order to be graded. (to create a working program quickly: get it working simply, then add to it; if at some point it stops compiling you will better know where an error was introduced)

6.3.2 **Final**: Final project – 50%

6.3.2.1 There will be a final project to test the overall ability to understand, design, implement, test, and reflect on the problem solving and programming knowledge and skills covered in the class.

6.3.2.2 The final project will be a mix of in-class (initial design, testing, and implementation discussion) and take-home (final design, testing, and implementation) elements.

6.3.3 Final grades will be given out based on the following based on score in the class:

90-100%: A
80-89%: B
70-79%: C
60-69%: D
00-59%: F

6.4 Reminder: A passing grade in order to count for course requirements for CS classes is generally a C or above.

7. Other Administrative Information:

7.1 For a list of general administration information (note that this list is not intended to be exhaustive), such as:

7.1.1 contacting me,
7.1.2 accessibility resources,
7.1.3 expectations of student conduct,
7.1.4 communications,
7.1.5 student assistance,
7.1.6 miscellany,
7.1.7 nondiscrimination & nonharrasment,

(each section contains a number of sub-sections and is not meant to be exhaustive of all situations)

see my administrative information document: *administrative_information* document.