Last updated 04 January, 2019

CS162 – Introduction to Computer Science II
Winter 2019

CRN: 30166
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 Python programming as a tool. You should also expect to do some reading, much practicing and searching, and much discussing of the topics we cover.

Note about the layout of this document: you do not need to memorize the numbering system, it is only there so we can refer unambiguously to a section in case of discussion or questions about the administrative details of the class.

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:

2.1 Lecture:	1400 – 1520,	Monday and Wednesday,	MKH101
2.2 Lab:	1400 – 1550,	Friday,	MKH101

3. Learning Outcomes (as written in catalog...):

On completion of this course, students will be able to:

3.1 Write and debug Java code using class dependencies of aggregation and inheritance.
3.2 Write Java code that includes the use of two-dimensional arrays and arrays of objects as well as input and output to/from a text file.
3.3 Demonstrate the use of various layout managers, borders and listeners in an event driven Java program.
3.4 Demonstrate the use of polymorphism through inheritance and draw UML diagrams of class hierarchies.
3.5 Write Java code that uses try catch clauses to handle exceptions.
3.6 Write Java code that uses recursion rather than iteration.

4. Learning Outcomes (as I really expect them):

4.1 Write and debug code using the concepts of aggregation and composition.

4.2 Demonstrate the use of various graphical and event driven techniques.

4.3 Demonstrate the use of polymorphism through inheritance and draw UML diagrams of class hierarchies.

4.4 Write code to perform sorting, searching, both iteratively and recursively.

4.5 Write code to handle exceptional program state.

5. Learning resources:

5.1 **Note**: All class materials other than the book and possibly physical storage will be freely available in a digital format.

5.2 (recommended, but not required)
Python Programming: An Introduction to Computer Science
Authors:John Zelle
ISBN-13: 978-1-59028-275-5

Publisher: Franklin, Beedle
Edition: 3
(Note: older editions should be alright and we can adjust assignments if necessary)

5.3 **Accessible storage**: Minimum capacity 16MB...
5.3.1 USB flash memory is easily available,
5.3.2 Cloud and other Internet storage should be accessible.

5.4 **The Python language** – available through several media. We will discuss this some in class.

5.5 **A Python interpreter** – I will use the official interpreter available from https://www.python.org/. We can discuss this more in class.
5.5.1 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).
5.5.2 We will discuss some capabilities of smart code editors during the course.

5.6 (**strongly recommended**) A desire to learn and experiment, to design, test, and problem solve with code (both on and off of a computer).

6. Grading:

6.1 Scores for coursework items will be available upon grading completion. I will keep a spreadsheet in a folder that I share with you through your student email account using Google Drive. We will discuss this some in class, including how to access it and keep yourself organized (which may affect your grade).

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.

6.2.2 Most assignments must be typed, spell checked, and grammar checked.
6.2.2.1 Any initial interpretation of documents will be at the mercy of my wit.
6.2.2.2 A common exception to this typed-up rule is that designs will generally be accepted as images.

6.3 To receive a passing grade in this course you must demonstrate at least basic proficiency in each of the following coursework item grading categories (that is to say that by failing any component of these you may receive a failing grade in the class):

6.3.1 **Demonstration: Discussion and weekly assignments – 50%**
6.3.1.1 There are a number of different programming projects to be completed as homework for this class, which 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)
Note: You must explicitly explain why your programs work rather than just showing that they run.

6.3.1.3 **Important 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).

6.3.1.4 Projects will be easier when some related exercises and approaches are discussed in a group.

6.3.1.4.1 **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)

6.3.1.5 Be sure to submit *all* relevant files (all report related documents, source files, and any data files used) for each assignment in *each* submission.

6.3.1.5.1 **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.

6.3.2 **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: *administrativeInformation.pdf*.