Last updated 07 January, 2019

CS162 Winter Term 2019 Schedule

Date	Topics (Tentative)
Week 1 07 Jan	On success in this class and hints on learning techniques (classwide notes) Cover syllabus, schedule, and getting registered for the class
	Intro, review, and crash course week Discuss **Assignment 1** (*review and early designs*) Quickly: variables, conditionals, and loops A bit more time: functions, classes, methods, and objects Discussion: What kinds of tests should we see and how can we design smart tests?
	lab
Week 2 14 Jan	Discussion: Go over Assignment 1, talk about challenges File IO (example of FileNotFoundError and try-catch block) Discuss **Assignment 2** (*refactoring for fileIO, documentation, and testing*)
	Importance of reflection, design, and testing! Code organization, code decomposition
	Lab
Week 3 21 Jan	Object identification, class decomposition, source decomposition (modules), teamwork, and tools First mention of recursion (factorial), recursive behavior vs a recursively defined structure Discuss **Assignment 3** (*team design*)
	Second mention of recursion (a fibonacci sequence) Basic graphical user interface (GUI) components with frames, panels, and output components
	Lab
Week 4 28 Jan	Getting all that decomposed code to communicate, interfaces, aggregate objects and nested classes, More GUIs, layout managers, borders, and some input components Accessibility of objects and adding behavior to make things more accessible (just like accessors and mutators) Discuss **Assignment 4** (*representing data sets and more GUIs*)
	Event driven programs and an MVC approach event listeners, registering an event listener, integrating multiple components into a logical action
	Lab
Week 5 04 Feb	Discuss **Assignment 5** (*animated search*) Version Control Systems Bonus lab! Start assignment 5 in class!
	Some searching algorithm ideas (random, linear, sequential on sorted list) Visualizing searching algorithms More recursion (recursively searching in a collection with binary search and doubling search)!
	Lab
Week 6 11 Feb	Some basic sorting algorithms (bubble, insertion, and selection sorts) Visualizing sorting algorithms One more bit on recursion (mergesort)! Discuss **Assignment 6** (*animated sort*)
	Bonus lab!?
	Lab
Week 7 18 Feb	Inheritance (inheritance, extends, generic vs specific, parent vs child, superclass vs subclass, UML, relationship) Discuss **Assignment 7** (*inheritance and polymorphism*)
	Exceptions (exception, exception class, exception inheritance, exception hierarchy, precedence of exceptions) Polymorphism (what is it and how can we benefit from it)
	Lab
Week 8 25 Feb	Review class topics!
	Initial discussion of final projects (**due 19 March**) and start in-class examples to support your final projects
	Lab
Week 9 04 Mar	Final projects (in class demos, design ideas, initial design)
	Final projects
	Final projects
Week 10 11 Mar	Final projects
	Final projects
	Final projects
Week 11 18 Mar	**Final projects due by 2359, Tuesday 19 March, 2019** **(We will not be meeting in person during finals week for this class)**

* Note that with assignment due dates extended, you may consider incorporating the 7th assignment, on inheritance and polymorphism, into your final project

What to turn in:

A *digital* copy of your code, with the appropriate header (sample header provided at the end of this document), in the appropriate folder, inside the Google Drive folder shared between your student account and the instructor.

A *digital* copy of your design and testing notes; this could be a picture of a drawing if this works better for you, just make sure and writing is very readable.

To be clear: I do not need a physical copy of your files (this item is included in case there are references to printing physical copies still around...).

Weekly Assignment Details (final project is at the end of the list):

Note: **keep notes** on things in these assignments, in a README file or the header of your driver script for each assignment, be sure to write down at least:

1. How hard or easy something is
2. Where did you find the info to try to solve the problem (books, webpages, social communities, friends, instructors, …)
3. Could we have made something easier or expanded on it in some way to make it more meaningful?

Assignment 1 (review and early designs):

1. Show me that you can ***design*** a Python class for a complex object (car, house, computer, being, container structure) by making me a drawing that shows:
   1. a few components (for a car it would be something like an interior, a transmission, a chassis, and a motor)
   2. a few attributes for each component (variables),
   3. a few behaviors for each component (methods),
   4. and the relationship between the components, the attributes, and the behaviors (perhaps with lines showing how a value would get used or calculated)
2. a simple program with a menu (user input and output, text in a console is perfectly acceptable) to interact with one of the components that you came up with that:
   1. Instantiates an object from a class that you write that has attributes and behaviors reflecting your design,
   2. gets basic input from a user that fills in the attributes of your object,
   3. displays basic info about your object and its behaviors,
   4. A menu item that allows the user to quit the program; saying goodbye when selected,
   5. Note: Be sure to use a loop to get back to the menu after finishing an item (other than quitting) to ask the user what to do next.
3. Include a comment near the top of your driver's source file that describes a few tests that would show the program working correctly if you did them (testing).
   (might be some new ideas here, we will practice these as we move forward in class)

Assignment 2 (refactoring for fileIO, documentation, and testing):

1. Discuss with classmates and show that you can create a ***document*** with the following:
   1. how do you think you would add file reading and writing functionality to your menu program,
   2. how you think the try and except clauses of Python work and what exactly the FileNotFoundError class is,
   3. identify sections of your code that you could create methods out of,
   4. design tests that would show your program working correctly (positive tests and negative tests),
2. Get another person's menu program (Assignment 1) and remember to **keep notes** on the following:
   1. Try to add file reading and writing to their program, note whether you accomplish this or not, how difficult was this,
   2. identify sections of their code that you think should be made into methods,
   3. what tests would show their programming working correctly (or would even break their code!),
3. create a list of the computer science topics that you have covered in your classes that are difficult, try to explain why you think they are difficult, and mention ways that we might make them easier to understand or work with.

Assignment 3 (team design):

1. Show me that you can work as a team to ***design*** a basic GUI:
   1. In a team, decide on a couple complex objects that could interact with each other,
   2. sketch a design with at least one component for each team member to work on,
   3. consider ways that your components and objects could interact with each other,
   4. as a team, design at least two tests for each of the components or objects,
   5. prototype a basic GUI with at least three different GUI components usable for input and output
2. Work with other people in the class (shared folder, email, telephone, telepathy, IP over Avian Carriers, or whatever) to:
   1. implement your assigned component(s), but do not worry too much if it does not end up properly interacting with each other as we will expand on this in the next project,
   2. Create a basic GUI that displays the attributes of your component (it does not need to be an interactive GUI yet, just set some text on a label or textbox as the program starts up,
   3. **Note**: I recommend using tkinter as it is built in to Python, but we could experiment with other GUI frameworks or libraries as well
3. Try to implement the tests that your team came up with (and more as you think of them),
4. Be ready to discuss how things go with your team when we next meet.

Assignment 4 (representing data sets and more GUIs):

1. Show me that you know what a set of data is and some ways of looking through it for important information.
    1. Describe a set of data that could be important to someone,
       (Maybe the set of quarter mile times for a group of racers in a group of cars or lines of code from programmers)
    2. Describe a sequence of steps to find a useful piece of information in that data set,
       (Maybe find the best times for individual racers (were they all using the same car))
    3. Discuss ways that grouping or ordering your data might help you find certain items or categories of data,
       (Maybe find the best times for each car grouped by make or model)

2. Prototype a GUI with a button, a text box, and a label,
3. Show how event listeners work by having the text of the label reflect the text of the textbox when the button is clicked

4. Extend the simple GUI to now include 9 more text boxes (all 10 held within some kind of collection) and change the listener to find the smallest number of the textboxes and display it in the label
5. Now have the button, when clicked, change the label's text to match that of the next smaller number in the list of textboxes.
   (so if you had the numbers 1 – 10 in the text boxes, then the first click would display a 1, the second click would display a 2, the next a 3, and so on)

Assignment 5 (animated search):

1. Show me that you can search through data:
    1. Create a collection of 100 int values,
    2. Create a GUI to display the collection of int values as rectangles,
    3. Include a text box in your GUI to allow a user to enter a value to search for,
    4. Include a button to start the search process,
    5. Highlight the candidate value at each step of the searching process,
    6. Pause for 250ms after highlighting the candidate value before moving on,
    7. Make it obvious when it has either found the value it is searching for or knows that the value is not in the data set.

Assignment 6 (animated sort):

1. Show me that you can sort data:
    1. Create a collection of 100 int values,
    2. Create a GUI to display the collection of int values as rectangles,
    3. Include a button to start the sort process,
    4. Highlight the candidate value(s) both before and after moving the value(s) at each step of the sorting process,
    5. Pause for 250ms after highlighting the candidate value(s)(before and after moving, possibly with different colors),
    6. Make it obvious when it has finished sorting the data set.

Assignment 7 (inheritance and polymorphism):

Note: this assignment may be split into multiple assignments or may be heavily modified this term!

1. Show me that you understand exceptions, inheritance, and the idea of polymorphism, how they are designed, and how they are handled
    1. Pick a generic object where something could cause exception in its state (example: parts breaking on a vehicle),
    2. Describe a generic exception that could happen in your object (example: BrokenCarPartError),
    3. Describe two subclasses of the object (examples: car, truck, starship),
    4. Describe two subclasses of the exception (examples: flat tire, busted hitch, reactor coolant leak)
    5. Draw an inheritance diagram illustrating the relationships between the various classes and the exceptions.
    6. Design a program that uses your classes and inheritance diagram showing how inheritance can work.
2. Implement your design in a simple program showing:
    1. classes working properly (just print their valid values and such),
    2. exceptions being raised, caught, and handled,
    3. an exception that does not get handled that crashes your program (save this part for last in your program please!).
3. Show me that you understand polymorphism
    1. Implement a different simple program with a collection of instances of each class at each level of the hierarchy:
       Example:
        1. a vehicle collection to hold instances of its subclasses,
        2. a car collection and a truck collection with some cars and trucks placed in them,
        3. then also place each of the elements of the car and truck collection into the vehicle collection (remember to make sure it is large enough!).
4. Print the data in an organized way, being sure to show the whole list of instances that could be in a given classification.
5. Add a third level that has sub-types, to follow the example, of cars (sedan, coupe, go-cart) and trucks (pickup, van,

semi) and show that they can be held in any collection with a type up the inheritance hierarchy tree.

Final project
Create a program that designs, uses, and demonstrates understanding of the following: 1. variables, conditionals, loops, and collections 2. code organization (formatting, identifiers, placement of definitions) 3. code decomposition (functions, classes, methods, and modules), 4. an understanding of design (hierarchy) 5. an understanding of testing (test methods and maybe a separate driver that runs tests on your other classes!) 6. user IO, file IO, and input validation, 7. recursion, 8. GUI components and event driven programming, 9. exceptions, 10. Inheritance and polymorphism.
Create a document that shows both an image of your program demonstrating the above topic and an image of some source code that makes that example work.

Sample Source File Header

(everything in angle brackets, < and >, should be replaced with your own information)

```
//*****************************************************************
// Name: <Your Name>
// Class: <your class, such as: CS162 Fall 2017>
// Class time: <your class starting time such as: 1200>
// Date:  <date this file was created>
// Project #: <number of project>
// Driver Name: <Name of driver source file for this program>
// Program Description: <What does this program do, stated in a few sentences?>
// Test Oracle: <Test Oracle goes here!>
//*****************************************************************
```