COMP 2150 CS 2: Object-Oriented Programming and Data Structure – Spring 2020
Dr. James Yu

Contact Information:

<table>
<thead>
<tr>
<th>Office: Dunn Hall 320</th>
<th>Department Office: Dunn Hall 375</th>
</tr>
</thead>
<tbody>
<tr>
<td>Office Phone: 901.678.3712</td>
<td>Department Phone: 901.678.5465</td>
</tr>
<tr>
<td>Email: <a href="mailto:jyu8@memphis.edu">jyu8@memphis.edu</a></td>
<td></td>
</tr>
</tbody>
</table>

| TA/GA: MdMaminur Islam (mislam3@memphis.edu) (Section 002)(10) |
| Zinat Ara (zara@memphis.edu) (Section 001)(15) |
| Utsav Shrestha (shrtha3@memphis.edu) (Section 001)(10) |
| Anup Shakya (ashakya@memphis.edu) (Section 001)(5) |
| Sabira Khanam Shorna (shorna.sabira@gmail.com) (Section 2)(5) |

The best way to get in touch with me is through email – I usually respond within 24 hours.

Office Hours:
No formal office hours, I am usually around in the morning (M, Tu, Thr, F). You can drop by any time. My afternoons are booked with teaching. It is best to email me to set up an appointment in advance.

Lecture Meeting Times/Locations:
- 12680 – COMP 2510 – 001  TR  5:30 pm to 7:30 pm  Dunn Hall 351
- 19982 – COMP 2510 – 350  TR  5:30 pm to 7:30 pm  Dunn Hall 351
- 25605 – COMP 2510 – 002  MW  2:20 pm to 4:20 pm  Dunn Hall 123

Catalog Description:
COMP 2150 – (CS 2): Object-Oriented Programming and Data Structures: Principles of object-oriented programming and software development; problem-solving with recursion and abstract data types, including linked lists, stacks, queues, binary search trees, hash tables; basic GUIs. Prerequisite: MATH 1910 or MATH 1421 (or MATH 1830 for COMP minors) and COMP 1900, and co-requisite Comp 2700.

Student Learning Outcomes (ABET):
This course focuses on the following ABET student outcomes and performance indicators:

1. An ability to analyze a problem, identify and define the computing requirements appropriate to its solution.
   (Performance indicator: Demonstrate an ability to break down a problem into smaller components.)
2. An ability to apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems in a way that demonstrates comprehension of the tradeoffs involved in design choices.
   (Performance indicator: Demonstrate an ability to evaluate the benefits and tradeoffs of different data structures.)

Course Website:
You can find the course materials (lecture notes, assignments, codes written in class, sample codes, grades, etc.) at the University of Memphis eLearn system at https://elearn.memphis.edu.

Required Text:
- Zybooks: Object-Oriented programming and the Essentials of Data Structures.
  - Sign in or create an account at up @ http://www.zybooks.com
  - Enter zyBook code: MEMPHISCOMP2150YuSpring2020
  - Click Subscribe
    You will need this book for the in-class exercises and assignments

Highly recommended textbooks:
It is crucial that you attend class regularly. The class will keep building on itself and moves at a fairly brisk pace. You need to get a good handle on each concept soon after we discuss it. Besides submitting assignments on time, you will complete the assigned zyBook exercises in class before almost every lecture starts. You are required to bring your laptop with the assigned zyBook textbook installed for this course. You will need to complete a time-limited zyBook questions (in-class portion) to show your presence in class. If you are missing from the class, you will receive no marks for the assigned zyBooks exercise. There is absolutely NO Extension for the missing (in-class or take-home portion) zyBooks assignments. The laptop (or iPad, NotePad, mobile device) is used for the lecture only. Using the devices for other purposes (eg net surfing, games, sports, non-2150 zyBook) is disrespectful to yourself, the class, and your professor. You might be requested to leave or sit at the back roll of the room.

Homework Assignments (225 pts), ZyBooks (ZB) Exercises (100 pts) and Programming Project: (100 pts)

There are 8 homework assignments given to reinforce the concepts discussed in lecture. You are required to do some of the zyBooks exercises in class after lectures. In addition to the assignments/ ZB exercises, there will be one programming project posted in the first week of March. You need to complete the project assignment one week before the last day of class (April 26). For the section 350 students, you are required to do an extra project assignment for the course. It is important to start working on the project as early as possible to avoid the last minute scrambling and produces poor quality software. It is MANDATORY that your project submissions successfully compile and runs without crashing in the "normal" scenario. A project submission that does not compile/run will receive zero credit. There is absolutely no extension to the project due date.

Attendance / participation: (part of zyBooks exercises)

It is crucial that you attend class regularly. The class will keep building on itself and moves at a fairly brisk pace. You need to get a good handle on each concept soon after we discuss it. Besides submitting assignments on time, you will complete the assigned zyBook exercises in class before almost every lecture starts. You are required to bring your laptop with the assigned zyBook textbook installed for this course. You will need to complete a time-limited zyBook questions (in-class portion) to show your presence in class. If you are missing from the class, you will receive no marks for the assigned zyBooks exercise. There is absolutely NO Extension for the missing (in-class or take-home portion) zyBooks assignments. The laptop (or iPad, NotePad, mobile device) is used for the lecture only. Using the devices for other purposes (eg net surfing, games, sports, non-2150 zyBook) is disrespectful to yourself, the class, and your professor. You might be requested to leave or sit at the back roll of the room.
Email:
Please check your University of Memphis email (or the email address you provided) regularly (daily), as that is my primary means of communicating with you outside of class.

Late/Makeup Policy:
All assignments (including zyBooks exercises) are expected to be completed and turned in on schedule. Each assignment will have specified due dates. Your TA/GA will not accept late assignments except in extreme circumstances. Likewise, makeup quizzes and exams will be given only under extreme circumstances. If you feel that your circumstances warrant a late work submission or a makeup quiz/exam, get in touch with me as soon as possible with documented proof of your situation.

Collaboration and Plagiarism/Cheating Policy:
An essential part of learning how to program is getting plenty of practice with it yourself. I also encourage you to work collaboratively and learn from each other. You are allowed to have similar design and codes. You have to comment at you program heading with “// collaborated with (list of name), and which parts of the program are your main contributions”. If I (or TA) determine that you have copied something directly from a book, the Internet, or some other source, you will receive a failing grade on the assignment and (at my discretion) a failing grade in the course. If I determine that you have copied another student’s assignment directly without any comment of your contribution, this will happen to both you and the person from whom you copied. The Office of Student Conduct will also receive a copy of the incident for further disciplinary action. Please don’t put me in this situation.

Important Dates and Deadlines for Academic Year 2019-2020

Getting Help:
Although I expect your work for this class to be done individually or collaboratively with conditions (see above), I encourage you to seek help as soon as possible if you get stuck: (this key is to start working on your assignment within 24 hours from the posted date)

• Talk to me! I’m very willing to sit down and try to provide hints without giving away the solution.
• Contact your course TA\GA.
• The Computer Science Learning Center (Dunn Hall 208) is open throughout the semester. Hours is posted on the door. The hours for this semester should also be posted online soon. You can find them by going to memphis.edu/cs/ and clicking on “Current Students” and then clicking on “Computer Science Learning Center.” The lab will be staffed by friendly, knowledgeable computer science students whom you can ask for help.

Student Disabilities:

If you require disability-related accommodations to meet the course objectives, please contact the Coordinator of Disability Resources located in the Student Development and Advising area of the student services building. For more information about Disability Resources or academic Accommodation, please visit the website at:

http://www.memphis.edu/drs/
<table>
<thead>
<tr>
<th>Wk</th>
<th>Date</th>
<th>Lecture</th>
<th>Quizzes</th>
<th>Assignments</th>
</tr>
</thead>
<tbody>
<tr>
<td>20-Jan</td>
<td>Martin Luther day (No Class)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>01/21 - 01/23</td>
<td>introduction / COMP 1900 (review)</td>
<td></td>
<td>HW 1: review COMP1900</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OO: Class and objects</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>01/27 - 01/30</td>
<td>Class and objects: OO thinking</td>
<td></td>
<td>HW 2: Classes and Objects</td>
</tr>
<tr>
<td>3</td>
<td>02/3 - 02/6</td>
<td>Inheritance</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>02/10 - 02/13</td>
<td>Polymorphism</td>
<td>Quiz1</td>
<td>HW3: Inheritance</td>
</tr>
<tr>
<td>5</td>
<td>02/17 - 02/20</td>
<td>Abstract and interfaces classes</td>
<td></td>
<td>HW4 Polymorphism</td>
</tr>
<tr>
<td>6</td>
<td>02/24 - 02/27</td>
<td>Exception handling</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>03/2 - 03/3</td>
<td>File I/O / Recursion</td>
<td></td>
<td>HW5 Abs/interface/Error</td>
</tr>
<tr>
<td></td>
<td>03/4 - 03/5</td>
<td>Midterm 1</td>
<td>MT</td>
<td>Project 1 (posting)</td>
</tr>
<tr>
<td>8</td>
<td>03/9 - 03/12</td>
<td>Spring Break - No Class</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>03/16 - 03/19</td>
<td>Generics ,Algorithm efficient: Big-O</td>
<td></td>
<td>HW6: Generics, Big O,List</td>
</tr>
<tr>
<td>10</td>
<td>03/23 - 03/26</td>
<td>Java Collection: Lists, Queues, Sets, and Maps</td>
<td>Quiz2</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>03/30 - 04/02</td>
<td>Linked Lists , Arraylist</td>
<td></td>
<td>HW7: List, Stack, and Queues</td>
</tr>
<tr>
<td>12</td>
<td>04/06 - 04/9</td>
<td>Stacks and Queues</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>04/13 - 04/16</td>
<td>Sorting (Insertion, bubble, Merge, Quick, bucket, Radix)</td>
<td></td>
<td>HW8: sorting/BST</td>
</tr>
<tr>
<td>14</td>
<td>04/20 - 04/23</td>
<td>Binary search trees (BST), Heap tree</td>
<td>Quiz3</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>04/27 - 04/28</td>
<td>Hashing; Hash tables</td>
<td></td>
<td>Project 1 due</td>
</tr>
<tr>
<td></td>
<td>29-Apr</td>
<td>Last day of class</td>
<td></td>
<td></td>
</tr>
<tr>
<td>30-Apr</td>
<td>Study Day (No class)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>05/1 - 05/7</td>
<td>Final Exam week</td>
<td>Final</td>
<td></td>
</tr>
</tbody>
</table>

The quiz dates are subject to change. Each quiz is approximately 45 to 60 minutes. Plan to attend the entire lecture on the quiz day. If you miss a quiz there is absolutely no retake.
This page summarizes the materials in COMP 1900. (From Kriangsiri Malasri (COMP1900 instructor) )

- Installing the Java Development Kit (JDK) and an IDE of your choice (I’ll be using Eclipse in class) onto your computer
- Structure of a basic Java program
  - Always starts with the class (program) name, followed by a `main` method that contains all the steps that you want the program to perform. You can have other methods besides `main` too, but whatever’s inside `main` is what will be executed when the program runs.
  - Standard Java conventions for `ClassNames, variableAndMethodNames, CONSTANT_NAMES`
- Single and multi-line comments (`//`, `/* */`)
- Java expressions and how they’re evaluated
  - Order of operations in numerical expressions (parentheses, multiplication/division/modulo, addition/subtraction)
  - Order of operations in Boolean expressions (parentheses, `!`, `&&`, `||`)
  - Constructing Boolean expressions using `relational operators` (`<`, `>`, `<=`, `>=`, `==`, `!=`)
  - Integer division and modulo
  - How expressions work when different data types are mixed (such as `int` and `double`, or `String` and anything else)
- Variables
  - Java’s `primitive data types` (`byte`, `short`, `int`, `long`, `float`, `double`, `boolean`, `char`) and what each can hold
  - Declaring a variable
  - Assigning a value to a variable
  - Declaring and working with constants
  - Shorthand notations for changing variable values (`+=`, `-=`, `*=` , `/=`, `%=`, `++`, `--`)
  - Implicit and explicit casting
  - Variable scope: a variable exists and is usable only within the block where it’s declared. Examples: a variable declared within a method body exists only within that method; a variable declared within a loop exists only within that loop.
- Program input/output
  - Using a `Scanner` object to read information from the user as your program is running
  - Remember that before you can use `Scanner`, you need to 1) include the `import` statement at the top of your program, and 2) create a `Scanner` object within the method where you want to read information
  - Displaying stuff on the screen using `System.out.println` and its variants
- Conditionals
  - `if`: execute a segment of code if a condition is true
  - `if-else`: execute one of two possible branches depending on whether a condition is true
  - `if-else if`: execute one of the multiple possible branches depending on a set of conditions. At most one branch can execute. An optional `else` may be added to the end to provide code to execute if none of the provided conditions are true.
  - `Switch`: allows you to test the value of an expression and execute code based on different case values. Case values must be integers. Remember that more than one case may execute if you don’t have a `break` statement. An optional `default` case may be added to the end to provide code to execute if none of the provided cases apply.
- Loops
  - `while`: repeatedly execute a segment of code as long as the provided condition is true. Condition is checked at the beginning of the loop, so the body of a `while` loop may not execute at all if the condition is initially false.
  - `do-while`: repeatedly execute a segment of code as long as the provided condition is true. Condition is checked at the end of the loop, so the body of a `do-while` loop is guaranteed to execute at least once.
  - `for`: consists of initialization, termination, and increment parts. Remember that this is just a concise way of writing a `while` loop – you can rewrite one type as the other very easily! You usually use `for` loops when working with arrays, and in other situations where you know exactly how many iterations the loop needs to go through.
  - Infinite loops
- Methods – a `method` is a block of code that performs some specific task. Useful for organizing a complex program into more manageable “chunks.” Once you define a method, you can use (a.k.a. `call`, `invoke`) it as many times as you want!
  - Terminology: parameters/arguments (method inputs), return value (method output)
  - Parameter passing: Java uses `pass-by-value`. Basically this means that the parameters specified in the method header (formal parameters) and the actual values you use when you call the method (arguments or actual parameters) are stored in two different memory locations. When you call a method, the values of the actual
parameters are copied over to the formal parameters, and the method performs its actions using the formal parameters. Hence, the actual parameters themselves can never be altered by a method!

- Using built-in Math methods such as Math.random, Math.sqrt, Math.pow
- Writing and calling your own methods to perform specific actions
- Method overloading – defining two or more methods with the same name. The methods MUST differ in the number and/or type of parameters. (Why can’t they differ only in return type?)
- Recursive methods – these are methods that call themselves. Get evaluated using your computer's call stack. Simple problems that lend themselves well to recursive solutions: computing the factorial of a number, computing powers, finding Fibonacci numbers.

**Arrays** – an array is just a collection of data of the same type. Each element in an array is associated with a numerical index. Indices start counting from 0 and go up to 1 less than the total number of elements in the array. In other words, an array of length n has indices from 0 to n – 1, inclusive.

- Declaring and instantiating an array
- Using .length to get an array’s length (number of elements)
- Remember, array variables are references, which are fundamentally different from primitive variables. Primitive variables store information directly. References store memory addresses where information is kept (you can think of this as “pointing to” a memory address). For you C/C++ folks – a Java reference is more or less the same idea as a C/C++ pointer.
  - Using = and == with array variables — what do they mean?
- Doing things to individual array elements — this usually involves a loop that performs the same action(s) at each array index
- How arrays work with methods – arrays as parameters, arrays as return values. Remember that what gets passed to/from methods are actually references to the arrays and not the array elements themselves.
- Working with 2-D arrays – these are really just arrays of arrays (or more precisely, arrays of references to arrays)

**Fundamentals of object-oriented programming (OOP)**

- In OOP, software is developed as a collection of software objects that interact with one another, as opposed to a single step-by-step sequence of instructions
- Every object is created from a description called a class. A single class can be used to create as many objects as desired. Objects created from a class are known as instances of that class, and creating an object from a class is known as instantiating the class.
- A class consists of attributes (a.k.a. instance variables), which are characteristics or qualities of the object, and methods, which are actions or behaviors. Different objects created from the same class can (and often do) have different values for their instance variables, but they all share access to the same methods.
- Syntax for creating an instance of a class, and calling methods using that instance (think about how you use the Scanner class)