CSCI 152 Interdisciplinary Computer Science II
Fall 2020

Class meets: Monday, Wednesday 9:00 – 9:50 a.m.
Labs are held Thursday and Friday at 9:00 – 9:50 a.m. You should register for one lab section.

Professor: Dr. Melissa Holmes
email: melissa.holmes@umontana.edu
office: Social Sciences 411
text messages: 406.565.6079
office hours: Posted on Moodle in the Getting Help section
Zoom Office: https://umontana.zoom.us/my/melissaholmes

Course Description:
This class will build on knowledge constructed in CSCI 135 and CSCI 151. The Java programming language will be introduced along with some object-oriented programming concepts and introductory data structures concepts.

Grading:

Short programming assignments, quizzes and problems	70%
Exams (2)	20%
Final Project	10%

Textbook:
None required. A variety of online resources will be used.

Accommodations:
Students who need any type of accommodation should work with Student Disability Services and provide appropriate documentation as soon as possible.

Academic Dishonesty:
You are encouraged to work in teams and use many resources including books and the Internet. However, each student must turn in his/her own work, and each student is responsible for understanding anything that is turned in. Refer to the Student Conduct Code for more information regarding plagiarism and cheating.

Student Learning Outcomes: Upon the successful completion of this class, students will be able to:

1. read a problem specification and define functional requirements for the problem;
2. design a program to elegantly implement requirements;
3. write Java programs of intermediate complexity;
4. articulate introductory Data Structures & Algorithms concepts;
5. use tools such as the JVM, command line compilation, and a modern IDE.

Course Schedule

Week	Topics
0 Aug. 19- 21	Syllabus, IDE and JDK, intro to Java, Hello World, Input/Output, Hello Name
1 Aug. 24 - 28	Variables, arithmetic, Payroll program Decision statements, fortune picker program
2 Aug. 31 – Sept. 4	Loops, Christmas tree/square printing problems Average of die tosses (random)
3 Sept. 7 - 11	Methods - types, parameters Arrays – declaration, traversals
4 Sept. 14 - 18	Methods and Arrays, pass by value/reference, exam review
5 Sept. 21 - 25	Midterm Exam, Disease Spread simulation (no efficiency improvements)
6 Sept. 28 – Oct. 2	Intro OOP / ADT Intro UML, simple classes and inheritance
7 Oct. 5 - 9	Arrays of objects
8 Oct. 12 - 16	Lists in Java (ArrayList, LInkedLIst, Stack, Queue, simple “classic” problems)
9 Oct. 19 - 23	Intro to Data Structures topics Big-O, Linear Search, Binary Search, File Input Spellchecker program (read in a dictionary of words, provide search capability)
10 Oct. 26 – 30	Implement a linked list (not Java’s linked list class) that maintains a sorted list of integers – insert, remove, find
11 Nov. 2 - 6	More linked lists – up through doubly-linked lists
12 Nov. 9 - 13	Intro to sorting algorithms (just insertion and selection, per Travis)
13 Nov. 16 - 20	Final problem: Evaluate and improve the efficiency of the Disease Spread Simulation – code and explanation Final exam question: Articulate implementation and performance differences between arrays and linked lists for searching, sorting, insertions in order, etc..
14 Nov. 23-25	Appointments to present your efficiency improvements; final exam paper due.

CSCI 152 Coding Standard

Comments

There should be 3-5 lines of comments at the top of each file. This heading should include your name, the class and semester, the lab # and title and the date. If it is part of a package you can include more information as needed.

Example:

```
/*      Melissa Holmes
        CSCI 152 Fall 2020
        Lab #1 Fortune Picker Lab          August 30, 2020
*/
```

Short comments throughout the code should describe the tasks of the program, i.e. "get n1 from the user", purpose of variables, and the like. However, do not over-comment if things are obvious – comments are supposed to make your code easily readable by others.

Whitespace

A line of whitespace should exist between the tasks of the program.

Indenting should be used to indicate nesting in blocks of code. Spaces, not tabs, should be used to indent. Modern IDE's have a setting to save whitespace as spaces.

Braces

When curly braces are used to indicate blocks of code, the first brace should appear on the same line as the code, and the closing brace should appear on its own line, i.e.:

```
if (condition)   {
        //lines of code

}  //closing brace on its own line
```

If appropriate for clarity, closing braces should be labeled with comments.

Identifier Names

Use identifier names that are meaningful, i.e. "firstSelection" may have more meaning than "a" or "n1"

Camel case should be used for variable and method names, i.e. theFirstVariable, myMethod

Camel case with the first letter capitalized should be used for class names, i.e. HelloWorld