CSCI 136: Fundamentals of Computer Science II
Spring 2012

Course Objective

A continuation of CSCI 135, this class explores the finer and more subtle elements of programming. Students will implement an industrial size project. Programming techniques and structures will include recursion, arrays, records, sets and linked lists. Prerequisite: CSCI 135.

Instructor

Keith Vertanen
kvertanen@mtech.edu
Museum 102, 496-4385

Office hours:	Mon	3:00 – 4:00pm
	Wed	2:00 – 3:00pm
	Fri	3:30 – 4:30pm
	or by appointment	

Classes

Mon	11:00 - 11:50am	Lecture	ENG 208
Wed	03:00 - 05:50pm	Lab	Main 205
Fri	11:00 - 11:50am	Lecture	ENG 208

Resources

Textbook	Head First Java 2nd edition by Kathy Sierra & Bert Bates
Class web page	http://katie.mtech.edu/classes/csci136/
Moodle	https://moodlemtech.mrooms3.net/course/view.php?idnumber=34606

Evaluation

A	90% - 100%	Exam 1	15%
B	80% - 89%	Exam 2	15%
C	70% - 59%	Exam 3	15%
D	60% - 69%	Programming assignments	55%
F	0% - 59%	Staff discretion (participation and extra-credit)	±?%

Assignment submission. All assignments need to be submitted via Moodle. You should upload all the source files required by the assignment. You should also include any other source files which your programs depend on (things not in the standard Java library). The top of every source file should include your name, username, and a description of what the class does.

Assignment deadline and late policy. All assignments are due at 10PM on the stated date. There is a two-hour grace period. Assignments arriving after 11:59PM are late. You get a total of four free late days. Each late day buys you a 24-hour extension to a submission deadline. If you are out of free late days, any further late submissions will be given a zero. Late days are on a per student basis (i.e. only students with sufficient late days get credit for a late group assignment). Weekends and holiday count as late days.

Assignment grading. Your code will be graded on correctness, programming style (including comments), and efficiency. Partial credit is possible so if you run out of time, submit what you have. If you want to do well, start well in advance of the deadline. This allows time to seek help if you run into trouble. Software bugs can be difficult to find and are often easily found after a good night's sleep.

Exams. There will be three exams that cover material from lectures, labs, and assignments. Makeup tests will only be given if you bring valid documentation explaining a legitimate reason for missing the test. Exams will occur during our scheduled lab period.

Academic dishonesty

Cheating will not be tolerated and can result in failure in the course. Submitted work must be your own (except for designated group assignments). Under no circumstances should you copy another person's solution or code. A student providing code to another student is considered as guilty as the student copying it (faculty handbook 308.4). Exams are to be strictly your own effort. Unless otherwise specified, no electronic devices are allowed in exams.

Programming is a creative process and no two programmers will solve the same problem in the same way. You are encouraged to discuss how to design a solution to a given problem with your classmates. But when it comes time to convert your design into code, you must write the code yourself. Be sure not to leave copies of your code where others might be able to access it (such as in the recycling bin of a lab computer). You may adapt code from the course materials provided you cite what code you used in your program's comments.

General

Any student who may need an accommodation due to a disability, please make an appointment to see me during my office hours. A letter from a Montana Tech Disability Coordinator authorizing your accommodations is needed.

Expectations

E1. Students should have basic problem solving skills and be able to apply those skills to create an algorithmic solution for a given programming problem. (CSCI 135)

E2. Students should be able to create programs using assignments, control constructs, and routines and be able to find and correct any syntax errors to produce a running executable. (CSCI 135)

E3. Students should be able to test their programs, find any logic errors, and correct these flaws to produce a working program. (CSCI 135)

Course Outcomes

R1. Students are proficient in a programming language and know basic error-handling, testing and debugging techniques. (CAC-c, i, j, k; EAC-k)

R2. Students use advanced programming techniques, including recursion, file I/O, abstraction, multi-file programs, and programming language libraries. (CAC-c, i, j, k; EAC-k)

R3. Students use and implement simple data structures, including stacks, queues, arrays, and linked lists. (CAC-c, i, j, k; EAC-k)

R4. Students use basic object-oriented programming methodologies including object-oriented problem decomposition, class design/implementation, and object creation/usage. (CAC-b, c, i, j, k; EAC-c, k)

R5. Students write defensive programs using an integrated design language and coding standard and basic error detection and reporting techniques with an emphasis on the responsibilities that computer professionals have for ensuring software quality. (CAC-b, c, i, j, k; EAC-c, k)

R6. Students understand and implement multi-threaded solutions to achieve provably correct synchronization among threads operating on shared resources. (CAC-c, i, j; EAC-k)

R7. Students integrate third party applications into their projects through socket connection/communications protocol, method API's, and command line interfaces. (CAC-c, i, j; EAC-k)

R8. Students implement client-server architectures which communicate over the Web using TCP and the sockets interface. (CAC-c, i, j; EAC-k)