Genesee Community College
Syllabus – CIS221 – Computer Programming 2 – Spring 2019

General Information:

Instructor: Ken Mead
Email: kjmead@genesee.edu
Office location: D395 Math Science Area, Building D, Main Campus
Office hours: TR 10:30-12:30, W 10-11am, or by appointment
Phone: 585-343-0055 x 6381
Homepage: https://faculty.genesee.edu/kjmead

IMPORTANT – Students enrolled in section CIS221-66: TESTS WILL BE PROCTORED. See the **Proctored Testing** section below.

Catalog description: A continuation of CIS219. Demonstrates advanced computing system concepts, problem solving and systematic program development in problems from a variety of application areas. Topics include program development, program testing and documentation, functions, files, advanced data structures, pointers, stacks, queues, linked lists, recursion, trees, sorting, searching, and object-oriented concepts. Students should plan sufficient time to complete the necessary programming projects using the college's computing facilities. Spring only.

Prerequisite: CIS219

Course Student Learning Outcomes (CSLOs):

The main objective of this course is for students to learn fundamental computer concepts and program development using a modern, high-level languages, such as Java. At the completion of this course, students will:

1. Demonstrate knowledge of the concept of arrays.
2. Know the steps involved in using arrays: declaring array reference variables, creating arrays, initializing arrays, and processing arrays.
3. Develop at least three methods with arrays arguments.
4. Know how to copy arrays.
5. Write at least two programs using multidimensional arrays.
6. Sort an array using the selection sort algorithm.
7. Search elements using the linear or binary search algorithm.
8. Demonstrate knowledge of objects and classes and the relationship between them.
9. Know how to define a class and how to create an object of a class.
10. Know the difference between object reference variables and primitive data type variables.
11. Demonstrate knowledge of the roles of constructors.
12. Declare private data fields with appropriate get and set methods to make a class easy to maintain.
13. Develop at least three methods with object arguments.
14. Know the difference between instance and class variables and methods.
15. Store and process objects in arrays.
16. Use UML graphical notations in at least two cases in order to describe classes and objects.

17. Determine the scope of variable in the context of a class.
18. Use the keyword this as the reference to the current object that invokes the instance method.
19. Declare inner classes.
20. Use the String class to process fixed strings.
21. Use the Character class to process a single character.
22. Use the StringBuffer Class to process flexible strings.
23. Use the StringTokenizer class to extract tokens from a string.
24. Know the differences among the String, StringBuffer, and StringTokenizer classes.
25. Know how to use command-line arguments.
26. Develop a subclass from a superclass through inheritance.
27. Demonstrate the ability to override methods in the subclass.
28. Invoke the superclass's constructors and methods using the super keyword.
29. Restrict access to data and methods using the protected visibility modifier.
30. Declare constants, unmodifiable methods, and nonextendable class using the final modifer.
31. Demonstrate knowledge of the useful methods (toString, equals, finalize, getClass, clone and hashCode) in the Object class.
32. Know the meaning of polymorphism, dynamic binding, and generic programming.
33. Describe casting and explain why explicit downcasting is necessary.
34. Develop familiarity with initialization blocks and static initialization blocks.
35. Develop a minimum of 10 programs which solve problems from a variety of areas using object oriented methods, creating applications that use objects, and employing the language elements that are listed in 1-34 above.*

* This course objective has been identified as a student learning outcome that must be formally assessed as part of the Comprehensive Assessment Plan of the college.

Content Outline:

1. Review of CIS219, conditions, loops, arrays.
2. Multidimensional arrays
3. Review (and introduction to) Objects
4. Object-Oriented Thinking
5. Inheritance and Polymorphism
6. Exception Handling and Text I/O
7. Abstract Classes and Interfaces.
8. GUI Programming and JavaFX
9. Event Driven Programming
10. Accessing Databases with Java
11. Extra topics to be determined.

Course Overview:

In this course, we'll be focusing on the Java programming language to learn how to program. Java is the most popular language in use on the planet (see the Tiobe Index at https://www.tiobe.com/tiobe-index/) , and is running on 3 billion + devices around the globe.

Here is a very general week-by-week description of the content of the course. This is by no means set in stone and should only be used as a guideline.

Week 1
Review of CIS219

Week 2
Multidimensional Arrays

Week 3
(Re)Introduction to Objects

Week 4
Object-Oriented Thinking

Week 5
Object-Oriented Thinking
Review for Test 1

Week 6
Test 1
Inheritance and Polymorphism

Week 7
Inheritance and Polymorphism

Week 8
Exception Handling
Text I/O

Spring Break

Week 9
Abstract Classes and Interfaces

Week 10
Finish Abstract Classes and Interfaces
Review for Test 2

Week 11
Test 2
Intro to GUI programming with JavaFX

Week 12
More GUI programming

Week 13
Using Java to access databases
SQL

Week 14
Finish Databases
Additional Topics

Week 15
Finish Additional Topics
Review for Test 3

Week 16
Test 3 and Semester Wrap Up.

Required materials:

Textbook: Introduction to Java Programming and Data Structures, Comprehensive Version (11th Edition), Daniel Liang, Pearson Publishing, ISBN: 978-0134670942

Proctored Testing:

Students in section 01 of this course will take exams in class on the dates listed above. Missed exams will result in a grade of zero unless the student has an unavoidable reason for missing the test AND you notifies his/her instructor in advance (either in person, by sending email, or by phone call and leaving voicemail).

Students in section 66 should plan on taking their exams during the scheduled exam time and location as well. However, if this is not possible, the student needs to make arrangements to take each test either:

a) with the instructor at a different time, or,

b) at one of GCC's testing centers during their regularly scheduled testing hours. (see http://www.genesee.edu/offices/cap/testing/ for all the details on testing hours and locations).

In either case, the student needs to notify the instructor at least one week in advance with the game plan. Your instructor understands that the campus centers have very limited testing hours, and he will work with you to allow you to take your test within a reasonable, one to two day, window around the scheduled test date. Any test taken outside this two-day window must approved by the instructor.

Grading:

Final grades are assigned according to the following scheme, with the final average rounded to the nearest integer (in %): 92 or higher = A, 90-91 = A-, 88-89 = B+, 82-87 = B, 80-81 = B-, 78-79 = C+, 72-77 = C, 70-71 = C-, 68-69 = D+, 62-67 = D, 60-61 = D-, 59 or less = F.

Grades will be weighted as follows:

- 60% - three in-class exams on 2/19, 4/2 and 5/7.
- 40% - Programming projects and homework assignments.

To compute your final average, divide your total number of points by 5, round to the nearest integer, and compare to the scale above.

Breakdown of Projects and Assignments:

There will be many short programming assignments that must be completed throughout the semester. These will be graded as either complete or not, all or nothing, binary 0 or 1. Collectively, these homework assignments will count as a single programming project: see next paragraph.

There will also be between 3 and 5 larger programming projects which must be completed to specifications provided by the instructor.

Student Responsibilities:

- Come to class, especially on exam day! If you must miss an exam due to extenuating and unavoidable reasons, you need to contact your instructor beforehand to re-schedule. Missing an exam without permission and good reason will result in a grade of zero.

- Complete projects on time! Late penalties will apply as follows: 10% within 3 days of deadline, 20% within first week of deadline, 30% after the first week. **If a project is not completed at all, it will receive a grade of zero and 2 points will be deducted from your overall average.**

- Do your own work! This means no cutting and pasting code from the internet. Students are encouraged to discuss programming assignments and projects, and even develop overall strategies for solving problems together, but must ultimately write their own code. Any evidence of plagiarism will result in a grade of zero for the assignment or project. Multiple offenses will result in a grade of F for the course.

- Stay engaged! Come up with ideas for problems to solve! Enjoy the material because it's really good stuff!!!

- Put the phone away. Respect your classmates. Don't be disruptive. Blah blah blah. You know the drill.

Plagiarism and Cheating:

Cheating is obtaining or intentionally giving unauthorized information to create an unfair advantage in an examination, assignment, or classroom situation. Plagiarism is the act of presenting and claiming words, ideas, data, programming code or creations of others as one's own. Plagiarism may be intentional – as in a false claim of authorship – or unintentional – as in a failure to document information sources using MLA (Modern Language Association), APA (American Psychological Association), Chicago or other style sheets or manuals adopted by faculty at the College. Presenting ideas in the exact or near exact wording as found in source material constitutes plagiarism, as does patching together paraphrased statements without in-text citation. The purchasing or sharing of papers or projects between students or the re-use of papers or projects submitted for more than one assignment or class also constitutes plagiarism.

In short - **do your own work!** This means no cutting and pasting code from the internet. Students are encouraged to discuss programming assignments and projects, and even develop overall strategies for solving problems together, but must ultimately write their own code. Any evidence of plagiarism, or of cheating or sharing answers on a test, will result in a grade of zero for the assignment, project or test. Multiple offenses will result in a grade of F for the course.

Accessibility Statement:

If you have a physical, psychological, medical or learning disability that may impact your coursework or participation in this class, please contact the Assistant Dean of Student Services/Disabilities Coordinator, Success Coach, or Academic Advisor who will arrange an intake meeting. The Assistant Dean/Coordinator will determine with you what accommodations are necessary, appropriate and reasonable. All information and documentation is confidential.

The instructor reserves the right to make any reasonable and necessary modifications to the statements above. This document is subject to change.