UNIVERSITY OF NORTHWESTERN
ST. PAUL

OFFICE OF DUAL ENROLLMENT

COS2071
Java Programming

SPRING 2021

SYLLABUS

Version: OLG v5:1:11/20

University of Northwestern – St. Paul
Office of Dual Enrollment
3003 Snelling Avenue North
St. Paul, Minnesota 55113
de@unwsp.edu

COS2071 Java Programming

University of Northwestern – St. Paul

COURSE DESCRIPTION

Course also listed as MIS2071. This course introduces event-driven computer programming using a graphical user interface and object-oriented language. Topics include classes/controls, objects, events, methods, properties, syntax, program structure, data types, functions, loops, conditional statements, and connecting to a database.

Credits: 4

Prerequisites: None

INSTRUCTOR INFORMATION

Please see "Contacting the Instructor" on the course site.

COURSE OUTCOMES

At the end of this course, a successful student will be able to

- CO-1. Explain the results and output of example computer code.
- CO-2. Apply data structures, loops, and procedures to the writing of computer code.
- CO-3. Employ modular principles in a team-working setting to develop a complete object-oriented application.
- CO-4. Test and revise computer code to run effectively and efficiently.
- CO-5. Design computer code to accomplish various tasks including: making computations, providing interaction with a user, or organizing information.
- CO-6. Design computer code to illustrate the principles of simplicity, clarity, and modularity.
- CO-7. Evaluate programming principles as illustrated in computer code.
- CO-8. Analyze the psychological and societal implications of increased human-technology interaction.
- CO-9. Identify programming principles with definitions and sample code.

MATERIALS

Required Textbooks and Materials

This course uses resources that are freely available entirely online.

A key skill of learning programming is practicing resourcefulness to teach oneself by finding necessary resources and documentation. Many resources and links are provided throughout this course to begin exposing you to the sorts of helps and documentation available to you for advanced learning beyond this course.

Additionally, question and answer forums and communities such as StackOverflow are also highly valuable for learning from others and troubleshooting difficult coding problems. You are encouraged to explore and participate in these types of communities as you develop your skills and profession in programming.

Provided by Student

For this course, students will need access to Microsoft Office (available at no cost to students through the University of Northwestern-St. Paul), a PDF reader, and a standard internet browser. Please refer to the Tech Requirements found in the Technology Help section at the top of the course site for the full requirements.

GRADING POLICIES AND PROCEDURES

Course Grade Explanation

Assignments	**Grade Weight**
Discussions (5)	10
Terminology Quizzes (6)	10
Explain/Evaluate Code (7)	10
Develop Apps (33)	50
Final Project	20
Total:	100

Grading Scale Percentages

A	≥ 93	B	≥ 83	C	≥ 73	D	≥ 63
A-	≥ 90	B-	≥ 80	C-	≥ 70	D-	≥ 60
B+	≥ 87	C+	≥ 77	D+	≥ 67	F	< 60

Late Work

All assignments are due as described in the course syllabus and the course site. Students are responsible for meeting assignment deadlines. Late assignments will be automatically deducted one letter grade. The assignments will drop an additional grade per day it is late, up to a 50% deduction in grade; late assignments will be not be accepted for a grade beyond one week past the original deadline. Forum discussion activities must be completed on time to earn points. Late forum posts will earn zero points. Students should contact the instructor via e-mail if an extenuating circumstance exists.

Feedback Expectations

Students should expect feedback for their submitted assignments within 5 days of the assignment due date or the time of their submission, whichever is later.

INSTITUTIONAL POLICIES AND SERVICES

Guidelines and Information

Students are responsible for all content of the DE Student Handbook. The most recent version of the DE Student Handbook is located on confluence.unwsp.edu and includes the following policies and procedures:

- Deadlines for Dropping or Withdrawing
- Student/instructor Communication
- Appeals, Exceptions, Disciplinary Process, & Grievances

- Assignments (late work and plagiarism)
- Examinations
- Grading System

Instructors may have course-related expectations that further detail the policies and procedures outlined in the DE Student Handbook. Any such expectations must be provided to students in writing (e.g., handout, course site posting) prior to or at the beginning of the class.

Traditional undergraduate students enrolled in DE courses are subject to the traditional undergraduate student handbook for all non-course-specific policies and procedures.

Academic Integrity

Plagiarism is theft—theft of someone else's words or ideas. It is claiming another's work as one's own. This would also include the following:

- Using the words or work of a former or current student in this class
- Recycling previously submitted assignments from a previous course attempt
- Using outside literature support sites such as, but not limited to, SparkNotes, Enotes or Schmoop that provide literary analysis of the texts we read throughout the semester

Students found plagiarizing are subject to discipline. The standard response ranges from loss of credit for the plagiarized assignment to earning an immediate "F" for the course to being placed on disciplinary probation. We should be committed to conducting ourselves with integrity in all things**.** Please refer to the DE Student Handbook for more detailed information about UNW's honesty and integrity policies.

In every course, students are required to view the Understanding Plagiarism video and complete the Understanding Plagiarism Quiz prior to completing any of the course content. These items are part of the course orientation.

Academic Achievement

UNW students requesting academic accommodations in association with the Americans with Disabilities Act (ADA) are directed to notify Disability Services to begin the application process. Academic Achievement also provides the following: Writing and Subject Tutoring, advocating, transitional skill building, Academic Coaching (organization, time management, test taking, etc.).

Contact Academic Achievement for more information: AcademicAchievement@unwsp.edu • 651-628-3316 • N4012 (Revised 06/20)

Support Services

Links to support services are available found in the Student Services section at the top of the course site.

COURSE POLICIES AND INFORMATION

Email and Announcements

Students are responsible to regularly check their Northwestern student email and the announcements in the course site in order to receive updates and information.

Attendance

Students are expected to participate in all course activities. Students must contact the faculty member in advance or as soon as possible if unable to participate in all or part of the course activities for a given week because of a medical (which includes having to quarantine or isolate due to COVID-19 exposure or confirmed illness), family, or work-related emergency. Students should refer to their course syllabus and/or faculty member for specific requirements. Students who do not participate in course activities and fail to withdraw from the course will receive a failing "F" grade.

Submission Standards

All written assignments should adhere to the following DE guidelines. Documents should be in the following format **unless directed differently by the syllabus or course instructor**:

- Submitted on the course site in Microsoft Word document format (.doc or .docx)
- Set in a traditional typeface 12-point font
- Double-spaced (unless the syllabus instructs otherwise)
- Set with one-inch margins
- Formatted in APA style for in-text citations and reference page (LIT1100 may ask for MLA documentation style)
- Labeled and submitted with the following information (APA papers require this information on a cover sheet, as detailed in A Pocket Style Manual): Student Name, Course Code and Title, Instructor Name, and Date.

Critical Response to Alternate Viewpoints

When students are reading or viewing course materials, they may encounter viewpoints, words, or images that their instructors would not use or endorse. Students should know that materials are chosen for their value in learning to read, write, and view critically, not because the materials are necessarily Christian.

ASSIGNMENTS

See the course site for complete details on the assignments.

Discussion Forums

Online discussions represent tremendous learning opportunities because classmates can share viewpoints and other ways of expressing complicated concepts that enable others with diverse learning and cognitive styles to better grasp and master ideas. Approach discussions as learning opportunities where you learn from others and they learn from you. You are not just writing for the instructor to read and grade; you are also writing to converse and help others and yourself better learn and understand at a deeper level.

For most forums, start an initial post in a new thread with 300+ words exploring one or more of the questions in the forum prompt. Respond to at least one of your peers' posts with 100+ words. For all posts, see the full grading expectations in the Discussion Rubric on the course site.

Terminology Quizzes

Purpose and Format

The purpose of terminology quizzes (about every other week) in this course is to provide you feedback on how well you have mastered and internalized basic understanding of key concepts seen in code samples. These quizzes ask you to look at snippets of code and match it to the proper definition of a key term illustrated by the code (or the bold segment of the code). Each quiz includes new concepts first, followed by review concepts drawing from all previous quizzes.

Certainty-based Scoring
Each set of matching questions also asks you to indicate your certainty about the accuracy of your submitted answers. Being very certain but incorrect will result in negative points, so be sure you are honest about your certainty so your score best reflects your actual mastery of the concepts.

Open Book with Time Limit
All quizzes are open book and open notes, but have an intentionally short timeframe requiring you to foster automatic recognition of core concepts at a glance if possible. Initial quizzes have fewer questions, shorter time limits (starting at 5 minutes), and contribute fewer points to your total course grade. Later quizzes have more cumulative review questions, and therefore longer time limits (up to 25 minutes), and contribute more heavily to your total course grade.

Unlimited Attempts for Mastery
You are permitted to try unlimited attempts for each quiz, since the purpose of these quizzes is to promote mastery and automatic recall of core concepts. Therefore, you may take each quiz as many times as you wish until it is due until you are satisfied with your score. You are encouraged to continue to take attempts to further practice master the concepts.

Explain Code Exercises

In addition to being able to write code yourself, being able to predict and explain what a piece of code should do without having to actually run it is a critical skill in mastering programming.

For each of the Explain Code Exercises (there are two or three in the course), view the code embedded or attached in each exercise and predict what the code would do if run. In a short essay of the length described in the exercise (usually around 200 words), explain what the code is meant to do, and how and why it does what it does. Include a picture (possibly hand drawn) with each explanation showing the output you would expect to see.

Explain Code Exercises may vary in grading criteria, but will generally be assessed on the following grading criteria:

• Meets Requirements (Including Length)	20 points
• Clarity of Explanation	20 points
• Thoroughness of Explanation	30 points
• Quality of Picture or Drawing for Illustration	30 points

Evaluate Code Exercises

While Explain Code Exercises ask you to predict what code will do and what output will result, the Evaluate Code Exercises (there are two or three throughout the course) ask you to go a step further. In the Evaluate Code Exercises, you must make well-reasoned, constructive quality judgments about whether code is written well according to standard quality criteria on which your own code will be judged when developing apps in this course. Be sure to provide logical rationale and specific examples to support your positive and negative critiques. If you judge something to be inferior or insufficient, be sure to provide specific ways it could be better.

For each exercise, submit a short essay of the length described in the exercise, answering the prompt questions to assess whether the code is efficient, logical, functional, and alternatives ways it could be better written for greater efficiency, clarity, functionality, or transferability.

Evaluate Code Exercises may vary in grading criteria, but will generally be assessed on the following grading criteria:

• Meets Requirements (including Length)	30 points
• Clarity of Explanation	30 points
• Thoroughness of Explanation	40 points

Develop Apps

The bulk of your skills in this course are built and refined through hands-on practice: writing Java code to develop diverse functional apps. You are expected to compile most of the apps produced (or fixed) for assignments into your final project toward the end of the course. Therefore, every assignment is meaningful! Apps begin simple in structure, sometimes asking you to fix problematic code, and then they progress in complexity and your autonomy as you master more advanced concepts and coding practices each week.

Sometimes you will find you have to be resourceful to find an answer to a question or reinforce your understanding of difficult concepts methods. Feel free to ask questions of your instructor, your classmates, and search for answers on the internet. Programmers must be resourceful when working in the real world so start the habit of finding what you need now.

The specific output requirements for each App Development assignment varies based on the project's task. You will typically submit code (sometimes with screenshots of sample runs, if requested) in a document on the course site, either filling out a provided worksheet with starter prompts or filling an entire document with your code from scratch. While you will likely prefer to code in Eclipse or another software, submit all your code files on the course site as a Word file in .txt, .doc, or .docx format.

Grading criteria may also vary between app development assignments, but generally your code is graded on some combination of the following criteria:

- **Meets Requirements**: The program meets the minimum requirements stated in the problem.
- **Logic**: The logic of the code is readily apparent and correct.
- **Correct Output**: The correct output for each program is included.
- **Efficiency:** The code is efficient, using as few lines as necessary, and comments are effective for transferability.

Final Project

Throughout the course you will build numerous apps that accomplish various technology and computation tasks. For your final portfolio for the course, you will work with other students' methods to build what we will call your own, customized "Ultimate Computer."

The name "Ultimate Computer" may seem wildly ambitious considering what astounding feats even simple computers routinely accomplish today. Even so, imagine yourself tinkering with and building the foundation of what could eventually become, decades down the road, a robust, all-purpose technology assistant or artificial intelligence like the AI butler Jeeves from Marvel's *Iron Man* stories. Obviously, none of us are programming geniuses like Tony Stark, yet anyway, but even in this course, you can already start customizing your own personal technology assistant.

Therefore, beginning in Week 12, you will begin the design of your initial version of an Ultimate Computer. You will create a package containing various classes that allow you to accomplish any imaginable task! Well ok, not quite any task, but hopefully it will still be somewhat impressive considering this is an early programming course.

Your computer should provide an organized menu interface that allows users to select and run any of the following tasks (which you will have already completed and submitted as assignments throughout the early course). Some of the listed tasks are related and therefore you may want them to appear in meaningful categories within the menus. The Ultimate Computer program should end when the user decides he or she does not want to run any of the available tasks.

Your final project should be able to run all the following tasks. Later on, after this course you can add functionality for remote controlling a fleet of wearable superhero armor suits, but that is for another day:

1. Display Favorite Saying
2. Give Encouragement
3. Draw Text Art
4. Openglopish
5. Standard Calculator
6. Binary Calculator
7. String Processor
8. Number Converter
9. Prime Decomposition
10. Double Factorial

To assist you with the development of this code you will each share two of your methods in discussions for others to use and modify.

A note on Collaborative Coding
One of the purposes of writing code is so that you can share it with others. You can find an enormous amount of code online that has been shared for the mutual benefit of anyone. For the final project you will be coding an "Ultimate Computer" with various tasks (called methods) that it is to be able to accomplish. So for mutual benefit, you will share two of those methods (one in week 12 and the other in week 14) with others, and in doing so will have access to a method from each other student in the class. Each person has a unique "style" to coding and will write different code, you will be able to see another style and to perform the method in a different way. By doing so you will learn from what others have done, just as you have learned from code created by John Purcel (at Cave of Programming), and Oracle (the Java Tutorials) and your instructor.

Accordingly, to practice the all-important coding skill of resourcefulness and making your own code highly useable by others, you are allowed to use and modify each other’s code in this course. However, if you use someone's code or algorithm idea, please give credit in the comments section of your program. As with all professional and academic work, careful attribution is crucial to avoid plagiarism or intellectual property theft.

COURSE SCHEDULE

Format

Everything needed to successfully complete this course in fifteen weeks is explained on the course site. Each assignment has been designed to work together during each week. When studying, be sure to follow the suggested format explained for each lesson.

For this course, students will receive access to each week's work as the semester progresses. There will be due dates during the week, but most weekly assignments will be due by 11:59 p.m. on Friday. Please refer to the schedule for the due dates of assignments.

Generally, for college-level work, students should expect to have an average of 9.5 hours of homework per week.

The last official class day in Week 15 varies from semester to semester. Please refer to the Semester Calendar found in the Academic Information section at the top of the course site for the actual last day of class. All course work must be completed and submitted by that day.

Due Dates

All written assignments (outlined below) are to be submitted on the course site by 11:59 p.m. CT on Sundays at the end of each week in which they are assigned, unless otherwise noted.

For any questions regarding these assignments, contact the instructor.

Orientation

- Read the Getting Started Page
- Participate in the Introductions Forum
- View and Complete Understanding Plagiarism Presentation and Quiz
- Complete Student Responsibilities Exercise

Week 1: What is the nature of Java programming?

- Install Eclipse on your computer
- View Video: Hello World Tutorial – Cave of Programming (4:46)

Due Wednesday

- Submit Display APP

Due Friday

- Submit Text Art APP
- Complete Participation in "Java New Code" Discussion

Week 2: How do I interact with a computer?

- View Video: I/O in Java (11:20)
- View Video: Variables Tutorial – Cave of Programming (7:53)
- Read "Variables" Java Nuts and Bolts Tutorial

Due Wednesday

- Submit Encouragement APP

Due Friday

- Submit Basic Variable Types APP

- Submit Evaluate I/O Code
- Complete Week 2 Terminology Quiz

Week 3: How does my program make computations?

- View Video: Computations in Java (8:08)
- Read Operators – Java Nuts and Bolts Tutorials

Due Wednesday

- Submit Evaluate Computation Code
- Submit Calculator APP

Due Friday

- Submit Binary APP
- Submit Number Converter APP

Week 4: How does my program make decisions?

- Read Flow Control – Java Nuts and Bolts Tutorials
- View Video: If – Cave of Programming (12:26)

Due Wednesday

- Submit Evaluate Control Code
- Submit Piecewise Defined Function

Due Friday

- Submit Messages APP
- Complete Wk4 Terminology Quiz

Week 5: How does my program repeat tasks?

- Read Iteration – Java Nuts and Bolts Tutorials
- View Video: While Loop – Cave of Programming (7:15)
- View Video: For Loop – Cave of Programming (9:28)
- View Video: Nested For Loop (2:40)

Due Wednesday

- Submit Dice Roll APP
- Submit Prime Decomposition APP

Due Friday

- Submit Explain Loops Code
- Submit Times Table APP
- Complete participation in Technology and Society Discussion

Week 6: How does my program process information?

- View Video: String Processing – Cave of Programming (9:21)
- View Video: String Builder – Cave of Programming (19:43)
- View Video: Tasks with String Builder (4:58)
- View Video: Substrings (3:00)
- View Video: Coded Messages (4:20)

Due Wednesday

- Submit Full Name APP

Due Friday

- Submit Random Letter APP
- Submit String Processor APP
- Be sure to get started on Openglopish APP

Week 7: How does my program store information?

- View Video: Arrays – Cave of Programming (9:46)
- View Video: Multi-dimensional Array – Cave of Programming (13:06)
- Read "Arrays" Java Nuts and Bolts Tutorial
- View Video: Sort (4:34)

Due Wednesday

- Submit Fibonacci APP
- Submit Openglopish APP

Due Friday

- Submit Pascal APP
- Submit Array Mod App
- Complete Wk7 Terminology Quiz

Week 8: How can information be stored in a way similar to humans?

- View Video: Objects and Classes – Cave of Programming (11:44)
- Read Object-Oriented Programming – Java Nuts and Bolts Tutorials
- View Video: Array of Objects (4:12)

Due Wednesday

- Submit Evaluate Classes Code
- Complete participation in Good and Bad of Technology Discussion

Due Friday

- Submit Person Info APP
- Submit GPA APP

Week 9: How do I organize related processes?

- View Video: Methods – Cave of Programming (11:05)
- View Video: Getters and Return Values – Cave of Programming (10:31)
- View Video: Method Parameters – Cave of Programming (15:00)
- Read Methods – Java Nuts and Bolts Tutorial

Due Wednesday

- Submit Person Tasks APP
- Submit Evaluate Method Code

Due Friday

- Be sure to get started on Special Project: GPS APP

Week 10: How do I get classes to run smoothly?

- View Video: Setters and 'this' - Cave of Programming (10:58)
- View Video: Constructors – Cave of Programming (10:18)
- View Video: OOP – Putting it All Together

- View Video: Recursion – Cave of Programming (17:26)

Due Wednesday

- Submit Temp Converter APP
- Submit Special Project: GPS APP

Due Friday

- Submit Person Stats APP
- Submit Recursive APP
- Complete Wk10 Terminology Quiz

Week 11: How can working code be used to develop new applications?

- View Video: Inheritance – Cave of Programming (14:09)
- Read Inheritance – Java Nuts and Bolts Tutorials
- View Video: Tic-Tac-Toe

Due Wednesday

- Submit Student APP

Due Friday

- Submit Evaluate Principles Code
- Be sure to get started on Special Project: Tic-Tac-Toe

Week 12: How are large-scale applications developed?

- View Video: Access (Public, Private, and Protected) - Cave of Programming (19:57)
- View Video: Package – Cave of Programming (14:03)
- View Video: Polymorphism – Cave of Programming (10:04)
- Read Package – Java Nuts and Bolts Tutorials

Due Wednesday

- Submit Many People APP
- Submit Special Project: Tic-Tac-Toe

Due Friday

- Complete participation in Package Discussion
- Driver: Submit "Menu with 10 Working Classes" Report to the Tester and on the course site
- Complete Wk12 Terminology Quiz
- Complete Ultimate Computer Project Menu
- Participate in Wk12 Share Sample Code Forum

Week 13: How can a computer interface with large amounts of data?

- View Video: Reading Files – Cave of Programming
- View Video: Reading a File Character by Character
- View Video: Reading Data From a File
- View Video: Geometric Facial Recognition – Computerphile (9:29)
- View Video: Intro Facial Recognition (5:40)

Due Wednesday

- Submit File Statistics APP
- Submit Total Bill APP

Due Friday

- Be sure to get started on Special Project: Facial Recognition APP

- Complete participation in Reading Files Discussion

Week 14: How can I analyze large amounts of data?

- View Video: Writing to Files – Cave of Programming
- View Video: Good Java Programming

Due Wednesday

- Submit Many People 2.0 APP
- Submit Special Project: Facial Recognition APP
- Complete Wk14 Terminology Quiz

Due Friday

- Complete Ultimate Computer Project Menu
- Participate in Wk14 Share Sample Code Forum

Week 15: How can I analyze large amounts of data?

The final week varies in length based on the semester. Please refer to the Semester Calendars found in the Academic Information section at the top of the course site for details.

Due on the Last Day of the Semester

- Submit Ultimate Computer Project Final Submission