syllabus



Application Development



OVERVIEW

Software development is a unique discipline that blends analysis and synthesis, formal rules and intuition. The “Computation and Problem Solving” course provided a high-level overview of this discipline, and this course builds on the concepts from that course by elaborating and extending its basic computational thinking and problem solving skills.

This course is designed for first year students. It assumes you have used a computer, but possess little or no programming experience. Successfully completing this course will prepare you to use the basic vocabulary of computing and create small standalone programs. In this course, you will:

Use interfaces, abstraction, generalization, and other crucial engineering concepts to reuse, tailor, and scale software solutions.

Learn how to use objects as a key strategy to control complexity and support reuse.

Examine basic design concepts, in parallel with programming language concepts, by discussing documentation techniques such as flowcharts, pseudo-code and simple Unified Modeling Language (UML) diagrams and implementations.

Put this knowledge to work by producing useful client-side applets and standalone applications, based on realistic requirements similar to those you would receive from an actual client or employer.

HOW TO SUCCEED IN THIS COURSE

To Pass the Course

In order to take the next course in this sequence, Application Design Choices (Java II), you must earn at least a “C” in this course. To do this, you must complete the following tasks at a bare minimum:

**Complete the assigned readings.** The bulk of the knowledge required to succeed in this course comes from the assigned reading in the textbook and other materials. When you come to class, I will assume you have completed the reading and will not repeat the information covered there—though I may ask you questions that require you to know it.

**Attend and participate in class.** Classroom time is about discipline, skill, and working toward professional performance and quality. Most classes will feature in-class discussions and exercises. While you will usually complete these in teams of two or three students, you will submit your work individually, as with all coursework.

**Produce solutions to seven programming projects.** Reading and listening to lectures will give you the basic knowledge you need, but the only way to learn programming is to do it. In this class, you will write working, stand-alone applications that meet real world needs. In-class discussion and exercises will often support the current projects, and the LMS contains a Help and Hints file for each project with additional tips for producing excellent work. For more information on projects, see the following section.

**Pass the Professional Assessments (exams).** These include:

* + Weekly Professional Assessments on selected reading and lecture material
  + Comprehensive Professional Assessments at the middle and end of the semester

To obtain a C or better in this course, you must achieve an average score of 60 percent or better on the Professional Assessments.

To Excel In the Course

Class sessions, reading assignments, and exercises cover most of what you need to meet the minimal requirements above. But if you wish to achieve a very good or exemplary grade, you will

need to go beyond that material by researching, practicing, and experimenting with the subject matter. Just as on the job, above-average performance requires initiative and taking charge of your own learning. The grading rubric at the end of this syllabus describes this requirement in more detail.

LEARNING-BY-DOING (LBD), STORY-CENTERED CURRICULUM

To be a successful software development professional, you must develop quality software. This course gives you opportunities to do just that. Learning-by-doing activities include individual and small group tasks, completed in a situated, context-rich environment that authentically mirrors the workplace.

This environment, called the “back story,” uses an approach called story-centered learning, which casts you in the role of a junior intern at a firm, iCarnegie Consulting, where you will solve problems that resemble those encountered by real companies every day. Most of the projects will begin with an email from your “manager,” whom you may also need to interview to scope out the project. You will also have access to flowcharts, requirements specifications, and other documents for the details of what is expected for each project.

As you complete the projects, I will play two roles in addition to my traditional function as an instructor. I will act as:

A **manager** who will play the role of your boss at iCarnegie. As your manager, I want (and expect) you to do well at the work I give you, and will have information to help you succeed. However, I have my own job to do, so you will need to respect my time. I will have little patience if you come to me unprepared or ask questions you could have answered yourself.

A **coach** who helps you to complete your assigned project, not by spoon-feeding you the answers, but by guiding you to find them on your own. As a coach, most of what I say to you will take the form of questions. My goal in doing this is not to frustrate you, but to make you think.

In this class, professionalism is as important as technical skill. Besides demonstrating computer know-how, you will need to work effectively in teams, examine problems from multiple viewpoints, ask questions and conduct research to ensure you understand client requirements, and articulate your solutions clearly, using standard computing terminology. Having mastered these skills, you can count on applying them regularly once you graduate and enter the workplace.

Another integral part of learning by doing is seeking and absorbing the hard-earned wisdom of seasoned domain experts in your community of practice. In this course, you will be able to do this by consulting me, of course. You will also have access to a series of videos on the LMS, entitled “Ask A Guru.” In these videos, software expert Professor Lynn Carter answers some frequently asked questions on the subject of software development. I encourage you to refer to these videos as needed to complete the deliverables for this course.

Finally, just as on the job, your work products will be evaluated against standards—by you, your classmates, the instructor, and the whole class. You will need to work both quickly and effectively, and doing so will require a solid command of the basic individual skills from the reading and lectures.

LEARNING OUTCOMES

After completing this course, you will be able to demonstrate the following **knowledge**, assessed through class discussions and Professional Assessments:

Given a computer program, explain its design and how it works, using a basic vocabulary of common computing terms such as value, constant, variable, classes, objects, attributes, constructors, methods, and parameters.

Given a problem, state whether (and why) a computer can solve it, cannot solve it, or if is difficult to know for sure.

Describe the Java primitive, composite, and reference data types and explain their uses in Java programs.

You will also be able to demonstrate the following **skills**, generally assessed through programming assignments:

Implement small programs to solve simple, well-defined problems.

Research answers to questions in books and other out-of-class information sources; evaluate your findings for relevance and reliability; integrate the findings with your own knowledge to produce an appropriate solution, and properly cite sources.

Create simple programs using simple assignment statements and console terminal input and output.

Write short programs (less than 300 lines of code) that use:

* + Control structures, basic functions, and primitive data types.
  + Basic UML documentation, including activity diagrams, class diagrams, state diagrams.
  + Object-oriented design concepts.
  + Classes, data encapsulation and information hiding.
* Use a development environment to create and debug simple applications.

Identify choices that have to be made to solve a problem, and weigh the pros and cons of a given solution.

Create applications from scratch that:

* + Properly use arrays to solve searching, sorting, and other similar simple problems*.*
  + Perform simple iteration.

Given a detailed design:

* + Create applications that solve more complex problems related to searching and sorting.
  + Perform more sophisticated iteration.
  + Implement a solution that uses all standard Java control-of-flow statements.

Explain the key ideas behind sorting and search algorithms, and implement at least one algorithm of each type.

Perform basic I/O operations to the console in textual form, as well as reading and writing text files.

After developing the programming and computer science-skills described by these outcomes, you should be ready to successfully take and complete Application Design Choices: Java II.

PREREQUISITES

You should complete the following courses prior to this one:

Computing and Problem Solving

CO-REQUISITES

You should complete the following courses concurrently with this one:

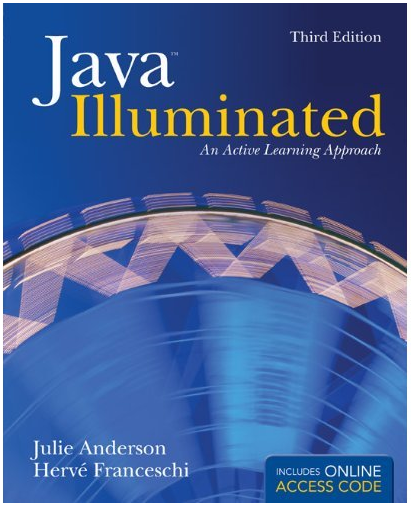
None

MATERIALS

Textbook

Java Illuminated 3rd Edition (An Active Learning Approach)

Julie Anderson and Hervé Franceschi

****

COURSE POLICIES AND EXPECTATIONS

Class Rules

Respect the learning environment:

Come to class on time and prepared to discuss the assigned reading.

Do not distract the class with excessive private conversations.

Turn off all mobile devices.

Use the lab computers in the room for exercises only.

Bring paper and pens/pencils.

Collaboration and Academic Integrity

Academic integrity is fundamentally about ethical behavior. Appropriate collaboration and research of previous work is an important part of the learning process. However, not all collaboration or use of existing work is ethical.

Different classes have different rules about collaboration. These are the standards you will be held to for this class. Unless otherwise noted on the assignment, we expect you to know and follow these rules.

You may only get help on graded assignments from designated people.

* + You are always welcome to get help on an assignment from your professors, professional school mentors, tutors, or any other faculty member. They may help you at the computer, on paper, or any way they believe will be effective.
  + Several undergraduate TAs may also be associated with this class. You may get their help with assignments outside of class, but these TAs are not allowed to help you at a computer. They may only help you work on paper.

There is a great deal of support available to you from a number of people. If you are struggling with an assignment, by all means, please seek help from one or more of them. You are welcome (and encouraged) to talk through concepts and ideas with your fellow students and study with them.

You may not, however, share your work with another individual or get help with your assignment from any other source, including members of your family. Avoid talking about any homework assignments in any specific way. In particular, do ***not*** give direct help to, nor receive direct help from, your classmates on a graded assignment. ***Never show your work to your classmates or seek to see their work***. Homework should be completed individually. In cases where inappropriate sharing occurs, ***all students involved are at fault***, regardless of whether they are the source or recipient of shared work.

If something has your name on it, you are claiming it as your own work and academic integrity rules apply. The assignments in this class are exercises designed to help you absorb and comprehend the covered topics. Doing the work is much more important than getting the right answer. The act of practicing things and trying is critical.

We live in a time when a vast amount of information is available online, and I have no doubt you can easily find source code or answers to questions on assignments. Before using this information, ask yourself if you are misrepresenting others’ work as your own. For example:

* + Copying source code you find online and turning it in is cheating and will be treated as such.
  + Reading a Wikipedia page that helps you understand an important concept to complete an assignment is perfectly acceptable.

There is a wide grey area between the above examples. Keep in mind that it’s the instructor’s judgment that counts! If you're ever unsure about whether an action is permissible, ***ask*** before you do it.

The severity of sanctions imposed for an academic integrity violation will depend on the transgression and ascertained intent of the student. Penalties for a first offense may range from failing the assignment to failing the course and referral to an academic review board. You can find more information about the consequences of academic integrity violations from Student Affairs**.**

CLASS FORMAT AND SETUP

Three 50-minute classes will be held each week. For typical weeks:

Two classes will consist of 10 minutes of organizing discussion and 40 minutes of exercises that address issues and questions from the assigned reading and homework.

The third class will begin with a 20-minute Professional Assessment (see below) covering the reading and class exercises, followed by 30 minutes of discussion and exercises.

You will also complete additional Professional Assessments in the middle and the end of the semester. See the course schedule for the sequence of class topics.

CUSTOMER PROJECTS

Early projects will give you program shells, which you will need to complete to make functional. In addition to filling in the blanks, you should examine these shells see how they work, expand and enhance them, and reflect on how to reuse the strategies. Later projects, beginning with Project #4, will give you less and less starter code, and by the end of the course, you will be writing the entire program yourself.

Below are the projects you will complete for iCarnegie, Inc.

Project #1 Integer Calculator (Weeks 1 & 2)

For this project, you will enhance the source code for an integer calculator so that the calculator will successfully perform addition, subtraction, multiplication and division functions.

Project #2 Floating Point Calculator (Weeks 3 & 4)

For this project will build upon the code you created for Project #1 to expand the functionality of your calculator so that it:

Operates on floating point numbers.

Displays last operation (sum, product, quotient, difference)

Alters error processing to support floating point input

Project #3 Calculator with Error Terms (Weeks 5 & 6)

For this project, you will build on the code you created for Project #2 so the calculator can process error terms associated with input values.

Project #4 Hangman Game (Weeks 7, 8 & 9)

For this project, you will enhance source code for a hangman game to get the game working, and ensure that the game works correctly.

Project #5 Basic eReader (Weeks 10, 11 & 12)

For this project, you will enhance source code for a simple eReader program to determine and display the number of notes associated with an open PDF file, and to display note information based on user input. Your enhancements will take advantage of the Eclipse plug-in environment, and must be consistent with the Eclipse look and feel.

Project #6 Annotation eReader (Weeks 13 & 14)

For this project, you will further enhance the eReader program to sort and search notes (based on author, date, rank, etc.) and display the results. Your enhancements will take advantage of the Eclipse plug-in environment, and must be consistent with the Eclipse look and feel.

Project #7 Enhanced Calculator (Enhanced Terms and Units) (Week 15)

For this project, you will further enhance the calculator from Projects 1-3 so that it:

Adds user-specific units, such as degrees (C/F) and length (Metric/Imperial).

Adds rules (i.e., can't add feet to inches without a conversion)

Updates displayed units when the user multiplies or divides

Enables users to add or remove units from the calculator.

COURSE GRADING

Your course grade will be calculated based on your scores on the deliverables described above, using the percentages below.

|  |  |  |
| --- | --- | --- |
| **Task** | **Quantity** | **Percentage** |
| Weekly Assessments | 10 (drop 1) | 25% |
| Customer Projects | 7 | 35% |
| Midterm Assessment | | 10% |
| Final Assessment | | 20% |
| In-Class Exercises | | 10% |

There is no curve for this course.

An “A” grade requires at least a 90%.

A “B” grade requires at least an 80%.

A” C” grade requires at least a 70%

A “D” grade requires at least 60%.

Factors for Grading Customer Projects

As you can see on the table above, the customer projects represent the largest portion of your course grade. Specific requirements for these vary by project, and you can find the detailed requirements on the grading form spreadsheet that is available for each project on the LMS. Primary grading factors include:

* Program documentation
* Proper use of Java language
* The program runs and computes as required
* The output of the program (e.g., layout, user messages)
* Correct submission of project deliverables per the instructions below
* Engineering Notebook entries

A few words are in order on the last item, the Engineering Notebook, as this assignment is likely to be unlike any you have completed before.

* In addition to adding an entry for your project work, I expect you to do so for each class you attend and exercise you complete. In fact, any time you do work on **anything** for this class, you should record it in your Engineering Notebook.
* I will review Engineering Notebooks at least each week, and more often if needed. I will assess your entries for professionalism, quality of writing, and the ability to see past the obvious to important root causes. The assessment rubrics provided as Appendix A of this syllabus describe how your work will be translated into a grade.
* I strongly encourage you to work with me to learn how to produce compelling, well-written submissions, to carefully review the feedback you receive, and strive to address the issues raised. If you have questions or are confused, let me know.

Due Dates and Times

**Readings** are due before the class period on the day the topic is covered. You will be expected to demonstrate your understanding of the material during the discussions and exercises.

**Projects** are due at 8 pm on the date specified in the course schedule. **Projects** will lose 10% off the top for each hour the assignment is submitted late. For example, a perfect paper submitted two hours late would earn a grade of just 80%. The time of submission is determined by the time stamp the course LMS assigns when the project is submitted.

Computers are unreliable, as are computer networks. Except in exceptional, extended cases, problems with the computer or the network are not reasonable excuses for being late. You may submit your work **more than once**, so if you find yourself struggling to finish an assignment, it is often better to submit a version before the deadline, even if it is not finished. This way, if the network goes down or your computer crashes; you have submitted something before the deadline. Start early and get your work done ahead of schedule and you will not have to worry!

Each of you is granted exactly **one** “I had a real bad day” 24-hour extension. You must inform your professor when you want to use this by sending me an email, with the subject “Application Development: I had a real bad day,” which specifies which assignment is to be given this extension. Do not even **think** about asking for a second one!

**Professional Assessments** are given at the start of class, generally on Fridays. There is no way to make up a Professional Assessment. You are allowed to drop one Professional Assessment, which means you can have a “real bad day” for a Professional Assessment as well.

**In-class exercises** require you to be present at the start of class and—with a few exceptions, which I will note—complete the majority of the in-class exercise before the end of class. Since there are so many exercises, and all together they only represent 10% of your grade, it does not make sense to “drop one” or allow extensions. If you come to class, actively participate, and try your best, you should do fine.

Engineering Notebook submissions made for a given week will be considered as a whole, but timely submission of each class day’s entries will be considered. If the submission for Monday’s entry (the first class day of the week) is made after 8am on Tuesday, it is considered to be late. The last class of a week’s submission is considered to be late after 8am on Monday. If one or more class day’s submissions are not on time, a 10% late penalty for the whole week will be assessed. Non-class day submissions are exempt from the on-time requirement. If your name or the date information is missing from any submission a 10% penalty for the whole week is assessed.

**Submitting Work**

Your name **must be** prominently displayed at the top of every document you submit, including each and every program source file.

**Exercises:** All exercises that require you to submit something to the LMS require you to name the exercise file or folder properly. The name must start with your school user ID. It must then be followed by a blank. The blank must then be followed by *ExerciseNN*, where *NN* is a pair of digits, ranging from "01" to "40.”  For example, Professor Lynn Carter's school ID is *LRCarter*, so upon submitting Exercise 1, he would name his submission *LRCarter Exercise01*.

**Professional Assessments:** All Professional Assessments that require you to submit something to the LMS requires you to name the file properly. The name must start with your school ID. It must then be followed by a blank. The blank must then be followed by *Professional AssessmentNN*, where *NN* is a pair of digits, ranging from "01" to "10.”  For example, Professor Lynn Carter's school ID is *LRCarter*, so upon submitting Professional Assessment 1, he would name his submission *LRCarter* Professional Assessment01.

**Programming tasks:** All programming assignments must be submitted via the LMS. All submissions must be named properly. The name must start with your school ID. It then must be followed by a blank. That blank must then be followed by *ProgramN*, where the *N* is a digit. For example, Professor Lynn Carter's school ID is *LRCarter*, so upon submitting Program1, he would name his submission *LRCarter* Program1. Programs are Eclipse projects, which are stored as folders on your computer. When you first create the project, name the project using this scheme so the folder will be named properly from the beginning. Before uploading the project, you will need to compress the folder into an archive file. It is this compressed (zipped) archive that you will upload to the LMS.

Do not attempt to change the name of a compressed archive after it has been created. When the archive is decompressed, the name of the original item will be restored, so the effort to change the name will be lost. If you rename the archive after it is created, you will lose credit.

Make sure your programming task is submitted via the LMS by **8 p.m. local time** on the last class day allotted for that programming task. For example, Project #1 is due by 8 p.m. on day 6 of the course.

Failure to follow these submission rules

If you do not include your name on the top of each and every document/file in the submission, you will lose 10% off the top. For example, a score that would have been a 94% will become an 84%.

If you fail to properly name the file before you upload it, or properly name a folder before you compress it and upload it, you will lose 10% off the top of your grade. For example, a project that would have been a 90% will become an 80%.

COURSE CALENDAR

Class 1

Topics

Welcome Back to iCarnegie Consulting

Your Assignments

Project 1: Integer Calculator

Exercise 1: Using the LMS

Reading Assignments

Chapter 1 (33 pages) – due Class 3

Chapter 2 (43 pages) – due Class 3

C02\_Binary\_Representation Reading - due for Class 4.

C03\_UnitsofMeasureReading - due for Class 3.

C03\_PrecisionReading - due for Class 3.

Class 2

Topics

Exercise 2: Hello World

Reflection

Reading Assignments

Chapter 1 (33 pages) – due Day 3

Chapter 2 (43 pages) – due Day 3

C02\_Binary\_Representation Reading - due for Class 4.

C03\_UnitsofMeasureReading - due for Class 3.

C03\_PrecisionReading - due for Class 3.

Class 3

Topics

Exercise 3: Compute Solid Object Volume

Reflection

Reading Assignments Due

Chapter 1 (33 pages)

Chapter 2 (43 pages)

C03\_UnitsofMeasureReading.

C03\_PrecisionReading.

Reading Assignments

C02 Binary Representation Reading – due Day 4

Class 4

Topics

Exercise 4: Binary Values

Reflection

Reading Assignments Due

C02 Binary Representation Reading

Reading Assignments

Chapter 3 (67 pages) – Due Day 9

Chapter 4 (24 pages) – Due Day 9

Class 5

Topics

Exercise 5: Data Types

Reflection

Class 6

Topics

Professional Assessment #1

Exercise 6: Computer Math Operators

Reflection

Deliverables

Project 1 - Integer Calculator due

Class 7

Topics

Project 2: Floating Point Calculator

Exercise 7: More Computer Math

Reflection

Class 8

Topics

Exercise 8: Why Does 0.1 Have Problems?

Reflection

Class 9

Topics

Exercise 9: Calculator Values

Reflection

Reading Assignments Due

Chapter 3 (67 pages)

Chapter 4 (24 pages)

Reading Assignments

Textbook Chapter 3 (67 pages) Due Day 12

Textbook Chapter 4 (24 pages) Due Day 12

Class 10

Topics

Exercise 10: Calculator Constructors

Reflection

Class 11

Topics

Exercise 11: String Class

Reflection

Class 12

Topics

Professional Assessment #3

Exercise 12: Reading In A Value With Errors

Reflection

Reading Assignments Due

Textbook Chapter 3 (67 pages)

Textbook Chapter 4 (24 pages)

Reading Assignments

Textbook Chapter 5 (55 pages) Due

Deliverables

Project 2 – Floating Point Calculator due

Class 13

Topics

Project 3 - Calculator with Error Terms

Exercise 13: Reading Data from a File

Reflection

Reading Assignments

Textbook Chapter 6.1-6.4 (30 pages) Due Day 15

Class 14

Topics

Exercise 14: Validating Input Data

Reflection

Class 15

Topics

Professional Assessment #4

Exercise 15: Leap Year

Reflection

Reading Assignments Due

Textbook Chapter 6.1-6.4 (30 pages)

Reading Assignments

Textbook Chapter 6.5 – 6.14 (57 pages) Due Day 18

Class 16

Topics

Exercise 16: Computing Sums and Averages

Reflection

Class 17

Topics

Exercise 17: Find Largest

Reflection

Deliverables

Project 3 - Calculator with Error Terms due

Class 18

Topics

Professional Assessment #5

Exercise 18: Hangman Documentation

Reflection

Reading Assignments Due

Textbook Chapter 6.5 – 6.14 (57 pages)

Reading Assignments

Textbook Chapter 7 (74 pages) Due Day 24

Textbook Chapter 8.1 – 8.2 (9 pages) Due Day 24

Class 19

Topics

Exercise 19: More Nested Loops

Reflection

Class 20

Topics

Exercise 20: Midterm Professional Assessment Review Puzzles

Class 21

Topics

Midterm Professional Assessment

Class 22

Topics

Exercise 22: Counting How Many Matches

Reflection

Class 23

Topics

Professional Assessment 06

Exercise 23: Displaying Matched Letters

Reflection

Class 24

Topics

Exercise 24: Get the Game Working

Reflection

Reading Assignments Due

Textbook Chapter 7 (74 pages)

Textbook Chapter 8.1 – 8.2 (9 pages)

Reading Assignments

Textbook Chapter 8.3 – 8.5 (31pages) Due Day 27

Class 25

Topics

Exercise 25: Standard Deviation

Reflection

Reading Assignments

Textbook Chapter 8.3 – 8.5 (31 pages) Due Day 27

Class 26

Topics

Exercise 26: Analysis of an Array

Reflection

Deliverables

Project 4: Hangman due

Reading Assignments

Textbook Chapter 8.3 – 8.5 (31 pages) Due Day 27

EReader Technical Overview (5 pages) Due Day 27

Reading Assignments

Textbook Chapter 8.6 – 8.10 (59 pages) Due Day 33

Class 27

Topics

Professional Assessment 7 – arrays

Exercise 27: Getting the eReader Running

Reflection

Reading Assignments Due

Textbook Chapter 8.3 – 8.5 (31 pages)

EReader Technical Overview (5 pages) Due Day 27

Class 28

Topics

Exercise 28: Display the Notes Icon

Reflection

Reading Assignments

Textbook Chapter 8.6 – 8.10 (59 pages) Due Day 33

Class 29

Topics

Exercise 29: Which paragraph has the most notes?

Reflection

Reading Assignments

Textbook Chapter 8.6 – 8.10 (59 pages) Due Day 33

Class 30

Topics

Exercise 30: Resizing Arrays

Reflection

Reading Assignments

Textbook Chapter 8.6 – 8.10 (59 pages) Due Day 33

Class 31

Topics

Professional Assessment 8 – sorting

Exercise 31: Linear Search

Reflection

Reading Assignments

Textbook Chapter 8.6 – 8.10 (59 pages) Due Day 33

Deliverables

Project 5: Basic eReader due

Class 32

Topics

Exercise 32: Search Speeds

Reflection

Reading Assignments

Textbook Chapter 8.6 – 8.10 (59 pages) Due Day 33

Class 33

Topics

Group Help session

Reading Assignments Due

Textbook Chapter 8.6 – 8.10 (59 pages)

Reading Assignments

Textbook Chapter 9 (63 pages) Due Day 36

Class 34

Topics

Professional Assessment 9 – searching

Exercise 34: Sorting Pages by Number of Notes

Reflection

Reading Assignments

Textbook Chapter 9 (63 pages) Due Day 36

Class 35

Topics

Exercise 35: Sorting Lists of Notes

Reflection

Reading Assignments

Textbook Chapter 9 (63 pages) Due Day 36

Class 36

Topics

Exercise 36: Implementing the eReader Case Study

Reflection

Reading Assignments Due

Textbook Chapter 9 (63 pages)

Reading Assignments

Textbook Chapter 10 (65 pages) Due Day 38

Class 37

Topics

Professional Assessment 10 – multidimensional arrays

Exercise 37: Multiple Key Sorting

Reflection

Reading Assignments

Textbook Chapter 10 (65 pages) Due Day 38

Deliverables

Project 6: Annotation eReader due.

Class 38

Topics

Exercise 38: Pascal’s Triangle

Reflection

Reading Assignments Due

Textbook Chapter 10 (65 pages)

Reading Assignments

Textbook Chapter 12.1, 12.2 and 12.10 (19 pages) Due Day 39

Class 39

Topics

Exercise 39: Project 7 Solution Design

Reflection

Reading Assignments Due

Textbook Chapter 12.1, 12.2 and 12.10 (19 pages)

Class 40

Topics

Exercise 40: Adding Fields to the Calculator

Reflection

Class 41

Topics

Exercise 41: Are these two units the same?

Reflection

Class 42

Topics

Exercise 42: Creating New Units via Multiplication

Reflection

Class 43

Topics

Group Help Session

Class 44

Topics

Exercise 44: Break and Continue Puzzles

Class 45

Topics

Course review, evaluation and final preparation

Exercise 45: Final Assessment Puzzles

Deliverables

Project 7: Enhanced Calculator due

Appendix A: Rubric Definitions

**Engineering Notebook Assessment**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Criteria** | **Poor (< 80%)** | **Average (80-89%)** | **Good (90-99 %)** | **Excellent(100%)** | **Score (0-100)** |
| **Actions and Effort** | **Action description** | Not related to deliverables or the course or there are not enough actions to address deliverables or irrelevant actions | Actions cover the deliverables but a few major problems | Actions cover the deliverables but some minor problems | Actions cover the deliverables with excellent insight |  |
| **Results** | Relation of these results to the action and a class deliverable is unclear | Satisfactory relation to the action and a course deliverable with a few major problems | Satisfactory relation to the action and a course deliverable with a few minor problems | Satisfactory relation to the action and a course deliverable, compellingly written |  |
| **Effort and Size** | Unable to use this effort figure to compute a production rate and other major problems | Able to compute a production rate with a few major problems | Able to compute a production rate with a few minor problems | Able to compute a compelling production rate |  |
| **Notes** | **Technical Content** | A major problem and missing a few key points | A few minor problems or missing a few key points | Technically good with the majority of the key points addressed | All of the key points covered compellingly |  |
| **Summarization** | Excessive text and not well written | Some focus but potentially confusing and wordy expression | Good focus and good expression | Compelling and compact expression |  |
| **Credible references** | Incomplete reference details or no reason to believe the source is credible | Valid reference but not specific enough and potentially credible source | Valid reference and potentially credible source | Valid focused reference with compelling rationale for its credibility |  |
| **Action Item Log** | **Usage** | Little relation to any key deliverable with a few major issues | Satisfactory action items to properly cover the key course deliverables but there are minor issues | Satisfactory action items to properly cover the key course deliverables with no issues | Actions cover the deliverables with excellent insight |  |
| **Results** | Relation of these result to the action or a course deliverable is unclear | Satisfactory relation to the action and a course deliverable with a few minor problems | Satisfactory relation to the action and a course deliverable with no problems | Satisfactory relation to the action and a course deliverable, compellingly written |  |
| **Effort and Size** | Difficult to use this effort figure to compute a production rate or other major problems | Able to compute a production rate with a few major problems | Able to compute a production rate with a few minor problems | Able to compute a compelling production rate |  |
| **Plans** | **Deliverables (what)** | Little relation to any key deliverable with a few major issues | Satisfactory coverage of upcoming key deliverables with a few minor issues | Good coverage of upcoming key deliverables with no issues | Insightful coverage of upcoming key deliverables, compellingly written |  |
| **Tasks (How, who, why, and when)** | Relation of these tasks to the deliverables has a few major issues | Relation of these tasks to the deliverables is satisfactorily described with a few minor issues | Relation of these tasks to the deliverables is well described with no issues | Relation of these tasks to the deliverable is compellingly described with no issues |  |
| **Lessons learned** | **Context (expected versus actual output and surprises)** | Description of any surprises or the results (expected and actual) and the gap between them has a few major issues | Description of any surprises or the results (expected and actual) and the gap between them has a few minor issues | Description of any surprises or the results (expected and actual) and the gap between them has no issues | Compelling description of any surprises or the results (expected and actual) and the gap between them with no issues |  |
| **Lessons learned** | Lessons and the relation of the context to the lessons are poorly described | Lessons and the relation of the context to the lessons are satisfactorily described with a few minor issues | Lessons and the relation of the context to the lessons are well described with no issues | Lessons and the relation of the context to the lessons are compellingly written with no issues |  |

**CODING RUBRIC**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| SNo. | Criteria | Poor (< 80%) | Average (80-89%) | Good (90-99 %) | Excellent(100%) | Score (0-100) |
| 1. | **Requirements** | The program is producing incorrect results. | The program produces correct results but does not display them correctly. | The program works and produces the correct results and displays them correctly. It also meets most of the other specifications.. | The program works and meets all of the requirements. |  |
| 2. | **Readability** | The code is poorly organized and very difficult to read. | The code is readable only by someone who knows what it is supposed to be doing. | The code is fairly easy to read. | The code is exceptionally well organized and very easy to follow. |  |
| 3. | **Documentation** | The documentation is simply comments embedded in the code and does not help the reader understand the code. | The documentation is simply comments embedded in the code with some simple header comments separating routines. | The documentation consists of embedded comment and some simple header documentation that is somewhat useful in understanding the code. | The documentation is well written and clearly explains what the code is accomplishing and how. |  |

**WRITTEN COMMUNICATION RUBRIC**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| SNo. | Criteria | Poor (<80%) | Average(80-89%) | Good(90-99%) | Advanced(100%) | Score(0-10) |
| 1. | Purpose is clear | Lacks clarity and purpose is not mentioned. | Often lacks clarity, ambiguous and confuses the reader. | Clearly describes the main idea. | Focuses on the main idea throughout the document without digressions. |  |
| 2. | Organization and structure | Information presented as one single paragraph. | Lot of information presented. Sudden jump from one idea to another. Missing links. | Information is too detailed but the though process is in alignment with the purpose. | Information presented concisely but clearly. Each paragraph explains one idea and gives lead to the next idea that achieves the goal. |  |
| 3. | Logical Reasoning | Information presented is unrelated and has no proper reasoning. | Inappropriate and Incorrect. Confuses the reader. | Appropriate and accurate. Convinces the reader. | Appropriate and accurate with in-depth analysis. |  |
| 4. | Writing provides clear evidence and supporting examples | Information presented has no supporting examples. | Often no proper evidence or supporting examples. | Provides enough evidence with clear examples. | Provides evidence with supporting examples that explain specific details. |  |
| 5. | Language is appropriate and professional | Abusive language and inappropriate tone. | Verbose, not comprehendible and improper choice of words. | Concise and clear. Obvious choice of words. | Concise, clear with a good use of vocabulary attracting the reader’s attention. |  |
| 6. | Grammar and punctuation | Information presented doesn’t follow the rules of grammar and punctuation | Sometimes incorrect grammar confusing the reader and spelling errors. | Free of errors, easy to read. | Error free and demonstrating mastery over the language. |  |

**WRITTEN FEEDBACK RUBRIC**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| SNo. | Criteria | Poor(<80%) | Average(80-89%) | Good(90-99%) | Advanced(100%) | Score(0-10) |
|  | Understand the purpose of feedback | No feedback given or feedback is given in monosyllables like good, okay etc. | Given feedback is very generic. Doesn’t explain details of the work. | Given feedback is specific. Every lowlight is clearly mentioned with details. | Given feedback is specific. Every highlight and lowlight is clearly mentioned with details. |  |
|  | Following a process | No process if followed while giving feedback. | Given feedback is not well structured and formatted. Lot of feedback but with missing links. | Given feedback is well structured and formatted. Action items that need to be focussed have recommendations. | Given feedback is well structured and formatted. Actions items are listed priority wise. More than one recommendation is given for the critical action items. |  |
|  | Measuring success | The given feedback doesn’t measure success on any level. | Given feedback indicates the success of the deliverable in a vague manner. | Given feedback indicates the success of the deliverable clearly. | Given feedback indicates the success of the deliverable and the proposed recommendations clearly. |  |
|  | Feedback delivery: Constructive and professional | Abusive language and improper tone used in feedback. | Given feedback is very critical and hence the receiver is not open to the idea of feedback. | Given feedback is constructive and professional that the receiver is open to the idea of accepting it. | Given feedback is constructive and professional that makes the receiver evoke a positive response. Receiver sends an acknowledgement and appreciation for the received feedback. |  |
|  | Feedback data used for course improvement | Given feedback has no relation with course improvement. | Given feedback doesn’t give any major insights of the course. It just suggests whether the course is good or average. | Given feedback gives enough insights for the course designer to improve the course. | Given feedback gives enough insights and proposes approaches subtly for the course designer to improve the course. |  |