Southern New Hampshire University

IT 145 Final Project Guidelines and Rubric

Overview

A successful career in software development depends on a thorough understanding of the fundamentals of object-oriented programming and best practices for software development. Your final project for this course will require you to apply the knowledge you have obtained prior to and during this course to the development of a simple, working program and accompanying process documentation. Professionals in software development document their process including requirements, design decisions, and defects for several different reasons as follows:

- To track what has been accomplished
- To track when items were completed in order to maintain a schedule
- To justify why a product works the way it does (verification and validation)
- To provide resources if a new team member is added so he or she can catch up
- To see where the most defects are being injected in order to prevent them
- To review what happened during the project in order to create new ways of improving the process

For your development project, you will imagine you are in charge of managing a zoo's computer infrastructure. There are many aspects of a zoo that need to be in place to keep it running. Two of those aspects are controlling data access and monitoring animal activities in exhibits. You will select which of these key components you wish to develop. Both options require at least two classes and for the design to be broken into multiple methods. Select one of the options provided in the prompt below and create your program and process documentation based on the specified requirements.

The final project for this course is the creation of an authentication or monitoring system. The final project represents an authentic demonstration of competency because it involves application of real-world Java programming. The project is divided into **one milestone** and several final project journal assignments, which will be submitted at various points throughout the course to scaffold learning and ensure quality final submissions. Milestone One will be submitted in **Module Five.** The final project will be submitted in **Module Seven.**

In this assignment, you will demonstrate your mastery of the following course outcomes:

- Implement appropriate variables, operators, methods, and classes as they are used in object-oriented programming for developing successful programs
- Utilize appropriate syntax and conventions in terms of their best practice and use in programming
- Debug coding errors by testing existing code, identifying errors, and correcting errors for improved functionality
- Assemble basic, working programs that effectively integrate essential elements of object-oriented programming

Prompt

You have assumed the role of managing the technology infrastructure at a zoo. You will develop a working program (either an authentication system or a monitoring system) for the zoo designed to follow the specifications outlined in the overview. You will also provide detailed documentation describing your development process. Select from **one** of the following options as the basis of your program.

Option 1: Authentication System

For security-minded professionals, it is important that only the appropriate people gain access to data in a computer system. This is called authentication. Once users gain entry, it is also important that they only see data related to their role in a computer system. This is called authorization. For the zoo, you will develop an authentication system that manages both authentication and authorization. You have been given a credentials file that contains credential information for authorized users. You have also been given three files, one for each role: zookeeper, veterinarian, and admin. Each role file describes the data the particular role should be authorized to access. Create an authentication system that does all of the following:

- Asks the user for a username
- Asks the user for a password
- Converts the password using a message digest five (MD5) hash
  - It is not required that you write the MD5 from scratch. Use the code located in this document and follow the comments in it to perform this operation.
- Checks the credentials against the valid credentials provided in the credentials file
  - Use the hashed passwords in the second column; the third column contains the actual passwords for testing and the fourth row contains the role of each user.
- Limits failed attempts to three before notifying the user and exiting the program
- Gives authenticated users access to the correct role file after successful authentication
  - The system information stored in the role file should be displayed. For example, if a zookeeper's credentials is successfully authenticated, then the contents from the zookeeper file will be displayed. If an admin's credentials is successfully authenticated, then the contents from the admin file will be displayed.
- Allows a user to log out
- Stays on the credential screen until either a successful attempt has been made, three unsuccessful attempts have been made, or a user chooses to exit

You are allowed to add extra roles if you would like to see another type of user added to the system, but you may not remove any of the existing roles.

Option 2: Monitoring System

As a zookeeper, it is important to know the activities of the animals in your care and to monitor their living habitats. Create a monitoring system that does all of the following:

- Asks a user if they want to monitor an animal, monitor a habitat, or exit
  Displays a list of animal/habitat options (based on the previous selection) as read from either the animals or habitats file

  - Asks the user to enter one of the options
- Displays the monitoring information by finding the appropriate section in the file
- Separates sections by the category and selection (such as “Animal - Lion” or “Habitat -Penguin”)
- Uses a dialog box to alert the zookeeper if the monitor detects something out of the normal range (These will be denoted in the files by a new line starting with *****. Do not display the asterisks in the dialog.)
- Allows a user to return to the original options

You are allowed to add extra animals, habitats, or alerts, but you may not remove the existing ones.

Specifically, the following **critical elements** must be addressed:

I. **Process Documentation**: Create process documentation to accompany your program that addresses all of the following elements:
   A. **Problem Statement/Scenario**: Identify the program you plan to develop and analyze the scenario to determine necessary consideration for building your program.
   B. **Overall Process**: Provide a short narrative that shows your progression from problem statement to breakdown to implementation strategies. In other words, describe the process you took to work from problem statement (your starting point) to the final product. Your process description should align to your end resulting program and include sufficient detail to show the step-by-step progress from your problem statement analysis.
   C. **Pseudocode**: Break down the problem statement into programming terms through creation of pseudocode. The pseudocode should demonstrate your breakdown of the program from the problem statement into programming terms. Explain whether the pseudocode differs from the submitted program and document any differences and the reason for changes.
   D. **Methods and Classes**: Your pseudocode reflects distinct methods and classes that will be called within the final program. If the pseudocode differs from the submitted program, document the differences and reason for changes.
   E. **Error Documentation**: Accurately document major errors that you encountered while developing your program.
   F. **Solution Documentation**: Document how you solved the errors and what you learned from them.

AI. **Program**: Your working program should include all of the specified requirements. The comments within your program will count toward the assessment of the documentation aspects of your submission.
   A. Functionality
      1. **Input/Output**: Your program reads input from the user and uses system output.
      2. **Control Structures**: Your program utilizes appropriate control structures for program logic.
      3. **Libraries**: Your program utilizes standard libraries to pull in predefined functionality.
      4. **Classes Breakdown**: Your program is broken down into at least two appropriate classes.
      5. **Methods**: Your program utilizes all included methods correctly within the classes.
      6. **Error Free**: Your program has been debugged to minimize errors in the final product. (Your program will be run to determine functionality.)

B. Best Practices: These best practices should be evident within your working program and process documentation.

1. **Formatting Best Practices**: Provide program code that is easy to read and follows formatting best practices as defined by the industry, such as with indentation.
2. **Documentation Best Practices**: Include comments where needed within the program in appropriate detail for communicating purpose, function, and necessary information to other information technology (IT) professionals.
3. **Coding Best Practices**: Ensure your program supports clean code through descriptive variable names.

Milestones

Milestone One: *Discussion Peer Review: Final Project Planning*
In **Module Five**, you will create and submit the pseudocode for the final project problem statement. **This milestone will be graded with the Milestone One Rubric.**

Final Submission: *Authentication or Monitoring System*
In **Module Seven**, you will submit your final project. It should be a complete, polished artifact containing **all** of the critical elements of the final project. It should reflect the incorporation of feedback gained throughout the course. **This submission will be graded with the Final Project Rubric.**

Final Project Rubric

Guidelines for Submission: Your process documentation should be approximately 2 to 4 pages double-spaced and in 12-point Times New Roman font. Any resource citations should adhere to the most current guidelines for APA formatting. You will submit the code for your working program in a separate file from your process documentation, though the two will be graded together by your instructor. Submit all files as a ZIP file to brightspace for grading.

Critical Elements	Exemplary	Proficient	Needs Improvement	Not Evident	Value
Process Documentation: Problem Statement/Scenario	Meets "Proficient" criteria and determination demonstrates a thorough understanding of necessary consideration for building the program (100%)	Identifies the program to be developed and analyzes the scenario to determine necessary consideration for building the program (85%)	Identifies the program to be developed and analyzes the scenario to determine necessary consideration for building the program, but analysis is cursory or contains inaccuracies (55%)	Does not identify the program to be developed or analyze the scenario to determine necessary consideration for building the program (0%)	4.75

Process Documentation: Overall Process	Meets "Proficient" criteria and narrative is exceptionally clear and well contextualized (100%)	Provides a short narrative that shows progression from problem statement to breakdown to implementation strategies that aligns to end the resulting program and includes sufficient detail to show the step-by-step progress from problem statement analysis (85%)	Provides a short narrative that shows progression from problem statement to breakdown to implementation strategies, but narrative is illogical or does not align to end the resulting program or does not include sufficient detail to show the step-by-step progress from problem statement analysis (55%)	Does not provide a short narrative that shows progression from problem statement to breakdown to implementation strategies (0%)	4.75
Process Documentation: Pseudocode	Meets "Proficient" criteria and demonstrates thorough understanding of creation of pseudocode (100%)	Breaks down the problem statement into programming terms through creation of pseudocode and explains whether the pseudocode differs from the submitted program and the reasons for the changes (85%)	Breaks down the problem statement into programming terms through creation of pseudocode and explains whether the pseudocode differs from the submitted program and the reasons for the changes, but pseudocode contains inaccuracies or explanation is illogical or incomplete (55%)	Does not break down the problem statement into programming terms through creation of pseudocode or explain whether the pseudocode differs from the submitted program (0%)	4.75
Process Documentation: Methods and Classes		Reflects distinct methods and classes in the pseudocode that will be called within the final program and explains whether the pseudocode differs from the submitted program and the reasons for the changes (100%)	Reflects distinct methods and classes in the pseudocode that will be called within the final program and explains whether the pseudocode differs from the submitted program and the reasons for the changes, but explanation contains errors (55%)	Does not reflect distinct methods and classes in the pseudocode that will be called within the final program or explain whether the pseudocode differs from the submitted program (0%)	5.94
Process Documentation: Error Documentation		Accurately documents major errors encountered while developing the program (100%)	Documents major errors encountered while developing the program, but documentation lacks detail or clarity or contains inaccuracies (55%)	Does not document major errors encountered while developing the program (0%)	7.92

Process Documentation: Solution Documentation	Meets “Proficient” criteria and demonstrates an acute ability to learn from resolved errors (100%)	Accurately documents how the errors were solved and what was learned from them (85%)	Documents how the errors were solved and what was learned from them, but documentation lacks detail or clarity or contains inaccuracies (55%)	Does not document how the errors were solved and what was learned from them (0%)	7.92
Program Functionality: Input/Output		Program accurately reads input from the user and uses system output (100%)	Program reads input from the user and uses system output, but with errors (55%)	Program does not read input from the user and use system output (0%)	5.94
Program Functionality: Control Structures		Program accurately utilizes appropriate control structures for program logic (100%)	Program utilizes control structures for program logic, but with errors (55%)	Program does not utilize control structures for program logic (0%)	5.94
Program Functionality: Libraries		Program accurately utilizes standard libraries to pull in predefined functionality (100%)	Program utilizes standard libraries to pull in predefined functionality, but with errors (55%)	Program does not utilize standard libraries to pull in predefined functionality (0%)	7.92
Program Functionality: Classes Breakdown		Program accurately breaks down into at least two appropriate classes (100%)	Program breaks down into at least two classes, but with errors (55%)	Program does not break down into any classes (0%)	4.75
Program Functionality: Methods		Program accurately utilizes all included methods within the classes (100%)	Program utilizes all included methods within the classes, but with errors (55%)	Program does not utilize all included methods within the classes (0%)	4.75
Program Functionality: Error Free		Debugs program to minimize errors (100%)	Debugs program to minimize errors, but significant errors remain that impact functionality (55%)	Does not debug program to minimize errors (0%)	7.92
Program Best Practices: Formatting Best Practices	Meets “Proficient” criteria and demonstrates a sophisticated awareness of industry best practices in formatting (100%)	Provides program code that is easy to read and follows formatting best practices as defined by the industry (85%)	Provides program code that is easy to read but follows only some formatting best practices (55%)	Does not provide program code that is easy to read or follows any formatting best practices (0%)	7.92
Program Best Practices: Documentation Best Practices	Meets “Proficient” criteria and shows keen insight into documentation best practices in programming (100%)	Includes clear, detailed comments where needed within the program for communicating purpose, function, and necessary information to other IT professionals (85%)	Includes comments where needed within the program for communicating purpose, function, and necessary information to other IT professionals, but comments lack detail or clarity (55%)	Does not include comments where needed within the program for communicating purpose, function, and necessary information to other IT professionals (0%)	7.92

Program Best Practices: CodingBest Practices	Meets “Proficient” criteria and shows keen insight into coding best practices in programming (100%)	Program supports clean code through descriptive variable names (85%)	Program supports clean code through descriptive variable names, but with errors (55%)	Program does not support clean code through descriptive variable names (0%)	7.92
Articulation of Response	Submission is free of errors related to citations, grammar, spelling, syntax, and organization and is presented in a professional and easy to read format (100%)	Submission has no major errors related to citations, grammar, spelling, syntax, or organization (85%)	Submission has major errors related to citations, grammar, spelling, syntax, or organization that negatively impact readability and articulation of main ideas (55%)	Submission has critical errors related to citations, grammar, spelling, syntax, or organization that prevent understanding of ideas (0%)	2.99
				Total	**100%**