Guided Exploration 03: Loops, Methods and TDD

**Points:** 30 (see rubric in canvas)

**Due Date:**  See Canvas.

Work will be accepted up to 24 hours after the due date with a 10% penalty. Meaning if you turn it in at 12:01 am of the next day you will be deducted 10% of the total points from your score. **If the assignment is more than 24 hours late, it will be a 0.**

Purpose: Guide your learning to meet the objectives for module 3

* **Understand and Explain Loops and Nested Control Structures**
  + - different types of loops: **for, while, do-while**
    - Sentinel values and boolean flags
    - Scope of variables in loops
    - Nested loops and conditions
* **Understand and Explain Methods**
  + Method signature: Method Name, Parameters and Return Value
  + Differentiate between methods with return values and methods without return values.
  + Modularize solutions using methods
  + Scope of variables in methods
  + Stack Frame Memory and Pass by Value
* **Implement Control Structures and Methods to Write solutions in Java**
  + Implement loops and conditions and nested control structures to control the flow of a program
  + Implement **methods** to modularize repetitive code and follow single responsibility principle
* **Follow Agile Software Development Practices**
  + Break the problem into smaller pieces
  + Follow test driven development to write test cases that include precondition and postcondition of tasks

**Effort: Collaborative and Individual**

* You can collaborate but individually **write your own code and answer the questions in your own words** . When you ask questions and explain to others you get a deeper understanding.
* **Remember: Only use concepts we have discussed in lectures. Do not use AI to answer the questions or to create code.** Handing in something that is not your own work is not beneficial to your learning and will be a violation of ([Academic Integrity](https://docs.google.com/document/d/1KsBBSZ3b227g1t8ciF3lwJBGnpKNCV9nEXRFmTLTZdY/edit#heading=h.1k4ycmhenia3)) .
* You can use it to clarify information or get help with syntax errors.

**Deliverables:** Upload each file separately and not as a zip file

* Upload this document as a pdf or word document with your answers
* Upload your part 2.2 code for grade calculator program .java file (not your class file)
* Upload your technical documentation

[Part 1: Explain and Analyze](#_4kqqreg5puc)

[1.1 Loops and Methods](#_lnqz0m7jv069)

[Part 2: Apply](#_4q84c1d4y6lu)

[2.1 Practice Writing code](#_601obpmk994h)

[2.1 Practice Test Driven Development, Pseudocode and Writing Code](#_iuyn6kbphhxe)

[Part 3: Technical documentation](#_ycma38s0k73f)

[Part 4 Reflection](#_vkb6ikjfq8q6)

# Part 1: Explain and Analyze

Go back to the lectures and the code I had you explore in class to complete the following.

## 1.1 Loops and Methods

Remember to go to the lectures where there are links to code examples.

|  |
| --- |
| 1.1.1 Explain the different types of loops. Include which loop to use in different situations and include code examples you explored in Eclipse and from the links in the lectures.  Do-while, while, and for are all loop types.  In the lecture, the do-while example code will allow you to run, and compile code to understand do-while loops.  [Code Example.](https://liveexample.pearsoncmg.com/liang/intro12e/html/TestDoWhile.html)  While – In a while loop, it will only run under the conditions that is specified in parentheses.  If the Boolean is false, the while loop will not run, the next body of code will run instead.  Do-while – The do-while is like the while loop. However, the difference is the do-while loop will run at least once. It executes the body first and then checks the conditions.  For – The for loop is like the other 2 loops, however, it is designed to run a loop a certain number of times.  All loops will need a sentinel value, this will end the loop, so it is not infinite. The loop will repeat until a certain value or condition is met. |
| 1.1.2 Explain how to define your own method. Describe parameters and return values. Include various examples for methods   * With and without parameters * With and without return value   Method Parameters - Parameters dictate numbers, types, and orders of your code. If the code that follows the method does not match the parameters, it will end in error. If a char is in the parameter, the return value needs to be a char.    In this example, the return value must be an integer, and will return the sum.  Methods without parameters – If void is used within the method, no return value is needed and therefore does not need a specified type, number, or order.    In this example, the method main is voided indicated by the keyword void. This means it needs no return value and has no specific data type or order.  Without return value – As said above, the void function in the method indicates no return value. Instead, the “return value” refers back to the main method. |
| 1.1.3 Explain different ways to exit a while loop and include examples.    To exit a loop, you will need a sentinel value. In this example, if the inputlogin or the login attempts are over 5, the loop will end, if the logpass and loguser are equal to the default value, that means the account creation wasn’t done and the login process cannot proceed. |
| 1.1.4 Explain ways you should not use to exit a loop.  Using break is generally not a good way to exit a loop, it can lead to situations that will interrupt the process of the loop. |

Part 1.2 Algorithms

# 

"You can’t just build an algorithm, put it on a shelf and decide whether you think it’s good or bad completely in isolation. You have to think about how that algorithm actually integrates with the world that you’re embedding in." Hannah Fry

A picture containing diagram

Description automatically generated

Read [Chapter 1](https://drive.google.com/file/d/1UNweLQrXpyFJ41N1lYy2ypCPp-9F41uu/view?usp=drive_link) from Hannah Fry, “Hello World, Being Human in the Age of Algorithms” and answer the following.

|  |
| --- |
| 1.2.1 In general an algorithm is  A step-by-step process to solve and find an end to a problem.  More specifically an algorithm in computer science is  Algorithms in computer science are pieces of code that act as the cogs of the digital age. |
| 1.2.2 Summarize the four main categories of real-world tasks that algorithms are doing.  The four main categories consist of prioritization, classification, association, and filtering.  Prioritization – Making an ordered list, something like Netflix choosing the next movie you watch. Classification – Picking categories, this is something like categorizing users or people into interests or groups.  Association – Finding links, looking for correlations or connections between users or data.  Filtering – Isolating what’s important, something like AI filtering your voice from other noise. |
| 1.2.3 Explain the two main approaches taken by algorithms. Include the pros and cons.  The two main approaches are rule-based and machine learning.  Rule-based learning – The AI runs and learns based on direct human instruction.  Machine learning – The AI may have a slight bit of guidance from humans, however, most of the learning and running comes from AI’s own data. |
| 1.2.4 Share something you found interesting in this reading.  I found it interesting how the author connected the AI’s actions with human actions. It humanizes AI making it a bit easier to connect with.mv |

# Part 2: Apply

## 2.1 Practice Writing code

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2.1.1. Write 2 test cases to find the sum of the numbers 1 to a max.   |  |  | | --- | --- | | Max 100 | 1++ <= 100 | | Min 1 | 1 | |
| 2.1.2 Write code to sum the numbers from 1 to a max using a for loop. Paste code below.  public static int numAdd(int firstNum)  {  int inSum = 1;  int addi = 1;    for (addi = 1; addi <= firstNum; addi++);  {  inSum = inSum + addi;  }  return inSum; |
| 2.1.3 Write code to sum the numbers from 1 to max using a while loop. Paste code below.    public static int numIn(Scanner input2)  {  Scanner input = new Scanner(System.***in***);  int firstNum = 0;  firstNum = input.nextInt();  while (firstNum > 1 && firstNum < 100)  {  System.***out***.println("Invalid number, enter a number 1-100");  firstNum = input.nextInt();  }  return firstNum;  }//end calculateSum |
| 2.1.4 Write code to sum the odd numbers from 1 to max. Paste code below. |
| 2.1.5 Create a method for 2.1.2 where you pass the max to the method and return the sum from 1 to the max. Call the method from main and after the method is called print out the sum.  I used a method in 2.1.2’s sample code already. This time, I called upon minimum and maximum values from the main method.  What type of loop did you use and why?  I used a while loop, this means that under the circumstances that the user inputted a value outside the test case (minimum and maximum) they will then be prompted to once again re-enter a value, this time, a valid value.  Explain how the max is passed to the method.  Paste method code below  public static int numIn(Scanner input2)  {  Scanner input = new Scanner(System.***in***);  int firstNum = 0;  firstNum = input.nextInt();  while (firstNum > min && firstNum < max)  {  System.***out***.println("Invalid number, enter a number 1-100");  firstNum = input.nextInt();  }  return firstNum;  }  public static void main(String[] args) {  // **TODO** Auto-generated method stub    int firstNum = 0;  int inSum = 0;  int value = 0;  int sum = 0;  final int max = 100;  final int min = 1;  // Declaring public variables/constants    Scanner input = new Scanner(System.***in***);    System.***out***.println("Enter a value 1-100");  value = *numIn*(input);  sum = *numAdd*(value);    input.close();  } |
| Update technical documentation |

## 2.2 Practice Software Development Life Cycle

Go to your guided exploration 01 code. Review the [solution for guided exploration 01 code](https://docs.google.com/presentation/d/1eJ27Rcm24wdI_sfsQ4ZkXVmntr7eSnwA/edit#slide=id.g3340178fec4_0_9). Make updates to your code using these examples.

This is to help you practice but it is ok if you don’t get everything working. You should have code showing you tried to implement all the methods.

You are adding more features to your code to calculate your grade. Here are the requirements

|  |
| --- |
| User Story:  Professors need a program to calculate a student’s final grade when entering the final grade for each category and display percentage and grade so they do not need to do the calculation.  Acceptance Criteria: Think about what tasks need to be completed   * Display summary of program including   + Grading categories and weights   + Grading Scale * The professor will enter the final percentage grade for each category.   + When grade not 0 to 105 display error and repeat until valid grade entered * Calculate final grade percentage * Determine Final Letter Grade * Displays final percentage to two decimal places and the letter grade * Ask the user if they want to enter another student’s grade. If they enter ‘y’ as first character repeat for next student otherwise display “Exiting grade calculator” and end program |

From the syllabus each category has a weighted percentage.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Grading Weights   |  |  | | --- | --- | | Class Participation | 10% | | Guided Exploration | 20% | | Module Quizzes | 25% | | Project Iterations | 25% | | Final | 20% | | Grading Scale:     |  |  | | --- | --- | | A | 90 or greater | | B | 80 or more but less than 90 | | C | 70 or more but less than 80 | | D | 60 or more but less than 70 | | F | Less than 60 | |

|  |  |
| --- | --- |
| Class Participation: 95.5  Guided Exploration: 88.4  Module Quizzes: 80.3  Project Iterations: 87.4  Final: 92.3 | 88.11 |

|  |  |
| --- | --- |
| Sample Output | Notes - Put notes to help |
| Sample output 1:  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  CS1050 Final Grade Calculator  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  ------------------------------  Category:Percent  ------------------------------  Class Participation: 10%  Guided Exploration: 20%  Module Quizzes: 25%  Project Iterations: 25%  Final Exam: 20%  -------------------------------  Letter Grade Range  -------------------------------  A: 90 to 100  B: 80 to < 90  C: 70 to < 80  D: 60 to < 70  F: < 60  Calculate Student’s Grade  Enter the Class Participation grade: 95.5  Enter the Guided Exploration grade: 88.4  Enter the Module Quizzes grade: 110  Enter a value from 0 to 105 80.3  Enter the Project Iterations grade: -1  Enter a value from 0 to 105: 87.4  Enter the Final Exam grade: 92.3  ------------------------------------------------  Final Percentage: 88.11  Final Letter Grade: B  ------------------------------------------------  Do you want to calculate another student's grade?  Enter y for yes or n for no **y**  Enter the Class Participation grade: 77  Enter the Guided Exploration grade: 56  Enter the Module Quizzes grade: 82  Enter the Project Iterations grade: 75.6  Enter the Final Exam grade: 67  ------------------------------------------------  Final Percentage: 70.97  Final Letter Grade: C  ------------------------------------------------  Do you want to calculate another student's grade?  Enter y for yes or n for no **n**  Exiting grade calculator | Task 1 Print out information about the program - this should be in a method.    DONE  Task 2: Get the grades but if not valid value from 0 to 105 repeat questions. Create a method to only return a valid grade.    DONE  Task 3: Calculate final percentage - can put in main    DONE  Task 4: Determine Letter Grade - create method where percentage is passed and letter grade character is returned    DONE  Task 5: In main put task 2, 3 and 4 into a loop  DONE |

Follow Software Development Life Cyle

* Break into smaller tasks
* Use Test Driven Development to
  + Create pre and post conditions for test cases
  + Design algorithm in pseudocode
  + Determine method signature - name, parameters and return values
* Implement a few lines of code for one task and run test cases

**Tasks**

Task 1: Print Grade Calculation Summary

Task 2: Get Valid grade

Task 3: Calculate final percentage

Task 4: Determine Letter Grade

Task 5: Repeat Calculating Grade

Here is a blank table for you to copy and paste for each task. You will have different amounts of test cases depending on the task goals and acceptance criteria.

**Task: 1 Program Summary**

Display summary of program including

* Grading categories and weights
* Grading Scale

No need for test cases or method.

Task 2: Get Valid Grades

The professor will enter the final percentage grade for each category.

* When grade not 0 to 105 display error and repeat until valid grade entered

2.1 Create test cases and include edge cases such as, -.1, 0, 88.5, 105, 105.1

|  |  |
| --- | --- |
| **Precondition** | **Postcondition** |
| -.1 |  |
| 0 |  |
| 88.5 |  |
| 105.1 |  |
| 105 |  |

2.2 Determine Method signature

|  |
| --- |
| Think about what the method needs and what will be returned. For example, for this method the signature would be   * Name for the method: getValidGrade * Parameters: the max and min values as integers for the grade * Return the grade as a double   double getValidGrade (integer min, integer max ) |

2.3 Design an algorithm in pseudocode on paper, tablet or whiteboard. Think about what type of loop.

|  |
| --- |
| Paste image of pseudocode |

2.4 Implement and Test Code

|  |
| --- |
| Add a little bit of code at a time to your java file and use unit test cases above to test if your method works. Remember to commit and push.  Start with creating the method and putting it below the main method but before the end of the class. You will pass the input scanner from main  /\*\*  \* Repeats prompting for double number until number with in range of min and max  \* @param int min for valid grade range  \* @param int max for valid grade range  \* @param inputReader Scanner object to read from keyboard  \* @return double valid grade value  \*/  public static double getValidGrade(int min, int max, Scanner inputKeyboard)  {  }  In main you will ask the question and then call the method as follows:  final int MIN\_GRADE = 0  final int MAX\_GRADE = 105  Scanner input = new Scanner(System.in);  System.out.print("Enter Class Participation grade: ");  double classParticipation = getValidGrade(MIN, Max, input);  System.out.print("Enter Guided Exploration grade: ");  double guidedExploration = getValidGrade(MIN, Max, input);  … |

Task 3: Calculate final percentage

|  |  |
| --- | --- |
| Class Participation | 10% |
| Guided Exploration | 20% |
| Module Quizzes | 25% |
| Project Iterations | 25% |
| Final | 20% |

Calculate final grade percentage that you calculated in GE01 and keep this code in main.

Task 4: Determine Letter Grade

|  |  |
| --- | --- |
| A | 90 or greater |
| B | 80 or more but less than 90 |
| C | 70 or more but less than 80 |
| D | 60 or more but less than 70 |
| F | Less than 60 |

2.1 Create test cases

|  |  |
| --- | --- |
| **Precondition** | **Postcondition** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

2.2 Determine Method signature

|  |
| --- |
| Think about what the method needs for parameters and what will be returned.  Method signature: Public Static Char letterGrade (Double finalG)  This means that our data types have to match what we will be using, in this case, a char will be returned, and double values will be used in reference. |

2.3 Design an algorithm in pseudocode on paper, tablet or whiteboard. Think about what type of control structure should be used.

|  |
| --- |
| Paste image of pseudocode |

2.4 Implement and Test Code

|  |
| --- |
| Add a little bit of code and test. Remember to commit and push. |

Task 5: Repeat Calculating Grade

Ask the user if they want to enter another student’s grade. If they enter ‘y’ as first character repeat for next student otherwise display “Exiting grade calculator” and end program

2.3 Think about what should be implemented to repeat.

|  |
| --- |
| Share what loop structure you chose to implement and why.  For this method, I chose to implement a do-while loop. This way, it will initially run, however, it will only continue to do the code while the user has selected yes, otherwise, any other word, including no, will end the loop. |

2.4 Implement and Test Code

|  |
| --- |
| Think about what goes in the loop. Implement the loop and test. Remember to commit and push. You will upload this code for your GE03. |

# Part 3: Technical documentation

|  |
| --- |
| Look at the objectives for module 3, the lectures and your work above to add module 3 to your technical document. Create headings and update the table of contents to make it easy to navigate. |

Read what the deliverables are for this assignment and submit them all to canvas. You can submit at different times but pay attention to the due date and the late date.