# GE M02 Classes, Objects, Methods, Decisions, Test Cases and Algorithms

**Points:** 25 (See Rubric in Canvas)

**Due Date:**  Due date listed in Canvas but some sections will be due as class participation before. Create Calendar reminders

* Final submission 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.

**Submission:** Upload files separately and do not upload a zip file.

* **Two Document Files as PDF or Word document:** This document with your answers in the highlighted areas unless otherwise stated and Technical Document
* **Two Java Files (. java files NOT CLASS FILES):** Code from
  + [2.1 Calculate Grades](#_heading=h.1evt9zrkqdqe)
  + [2.2 Concert Ticket Prices](#_heading=h.locmil69imvq)

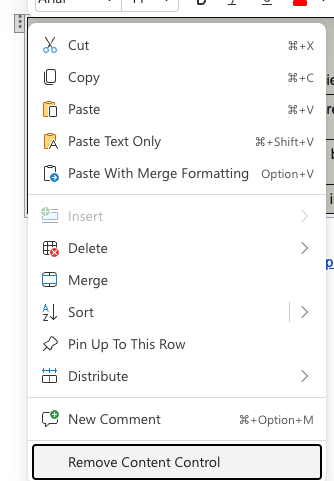
**Objectives:**

* Analyze and apply relevant theories, principles, and methods from computing, including algorithms, coding structures, and version control tools.
* Develop effective technical documentation to reflect on and solidify learning, with a focus on clarity and professional presentation.
* Apply version control and debugging practices using Eclipse and GitHub.

**Effort:**  You are encouraged to collaborate to discuss concepts and explore writing code together. Remember

* Review lectures and links in the lectures to information first.
* Follow [CS Academic Integrity and AI Policy - Harding](https://docs.google.com/document/d/1SY4-RMJ2B9GkEbTzmpRVSfKKsgQnYWYF/edit)
* If you find you don’t understand the information in the lecture reach out to **Deb, Elaine or Heriberto**

If you download this as a word document and you have any problems typing your answer in the tables you must click on 3 dots and remove content control.



|  |
| --- |
| [GE M02 Classes, Objects, Methods, Decisions, Test Cases and Algorithms](#_heading=h.gabctt8gkr9i)  [Part 1: Explore and Explain](#_heading=h.4ddd3z5g8ss4)  [Part 2 Analyze and Apply](#_heading=h.b2767o9rmjql)  [2.1 Calculate Grades](#_heading=h.1evt9zrkqdqe)  [2.2 Concert Ticket Prices](#_heading=h.locmil69imvq)  [3 Reflection](#_heading=h.unzmybdbgbo) |

# Part 1: Explore and Explain

**Purpose:** Professional communication to demonstrate your learnings by connecting different ideas from lectures or provided resources.

* Clearly explain key ideas in your own words and include supporting evidence such as code examples, images, screenshots, or charts.

Update your technical document for Module 2.

* Use headings to organize topics and create or update a table of contents to be able to quickly access information.
  + [Headings, Subheadings, and Table of Contents (Google Docs)](https://guides.lib.uni.edu/c.php?g=1243498&p=9758563)
  + [Headings, Subheadings, and Table of Contents (Microsoft Word)](https://guides.lib.uni.edu/c.php?g=1243498&p=9207893)
* Include summaries, code snippets with comments to demonstrate your understanding of
  + Standard Library Packages, Classes and Methods
  + Scanner and constructing objects to call methods to read input from keyboard
  + System - to call methods to print out to console (print, println, printf)
  + Math - pow and random
  + Characters
  + String - store array of characters as a String and use String class methods such as length and charAt.
  + Boolean, relational operators
  + If-else, multiway and nested
  + logical operators
  + Switch statements
  + Scope of variables
* Add more comments to your GuessNumber.java code from GE M01 and include in your tech doc.

# Part 2 Analyze and Apply

## 2.1 Calculate Grades

Update your calculate grades java code from GE M01

1. Remove your introduction.
2. Prompts the user and reads in the first name and last name of a student as Strings.
   1. Enter Student 1 first name: Stack
   2. Enter Student 1 last name: Terror
3. Extract the first character of both the first and last names and store in separate char variables for the student’s first initial and last initial
4. Creates the initials of each student by concatenating the first character of their first and last names with periods. Store this in a String.
5. Student 1 Initials: S.T.
6. Prompt the user to enter each grade category for this class.
7. Calculate the grade.
8. Determine the letter grade for the student
   1. A: 90 or better
   2. B: 80 or better but under 90
   3. C: 70 or better but under 80
   4. D: 60 or better but under 70
   5. F: Under 60
9. Print out the Student Name, Numeric Grade to 2 decimal places and their letter grade

Stack Terror 65.35 D

1. If the grade is less than 75 print the student initials and a message to go to office hours.

S.T. Please come to office hours.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1. Create test data to make sure you test each letter grade category. Review M02L07 test cases for the BMI program.   |  |  | | --- | --- | | Pre-Condition | Post Condition | | CP = 94, GE 89.5, Q = 95.2, P= 97 F= 93  Final Grade = | A | | Final Grade = 85.4 | B | | Final Grade = 72.6 | C | | Final Grade = 69.9 | D | | Final Grade = 55.0 | F | |
| 2. Write your algorithm on paper.  What if/else algorithm did you choose to display the correct category status and why? Why can’t you use a switch statement? |
| 4. Implement a little bit of code and test. Run the debugger to show your test data will step into the correct condition. Put a screenshot showing the variables while using the debugger. |
| 5. Add comments to your code so you can use this in your technical document. Commit and push. Put a screenshot showing your code in your github remote repository. |

## 2.2 Concert Ticket Prices

Write a java program to determine the ticket price for one seat at a concert where you

* Create test cases for precondition and postconditions
* Write an algorithm on paper
* Implement a little bid of code and test after you complete sections below

The code must:

1. Prompt the user for a character that represents the seating section.  
    Use these characters and section names (store as char, not String):
   * **F**: Front Section
   * **C**: Center Section
   * **S**: Side Section
   * **U**: Upper Section

Use Character.toUpperCase(someChar); to convert the inputted value to upper case before determining if it is invalid.

If an invalid section is entered, print a message and end the program.  
Example: When user inputs X, display “Invalid section, program ending”

1. **After the section character is read and verified**, display the full section name.  
    Example: input C → output Selected Center Section
2. **Prompt the user for the row number** (integer between 1 and 60).
   * If an invalid row is entered, print a message and end the program.  
      Example: When user inputs 0 or 75, display “Invalid row, program ending”
3. Seat pricing depends only on row:  
     
   Row 1–15 cost $450

Row 16–30 cost $300

Row 31–60 cost $200

1. **Output the result** showing section, row, and price.

**Example Outputs**

**Example #1: Valid Input**

Enter ONE character for section - F: Front C: Center S: Side U: Upper

C

Selected Center Section

Enter row

22

Section C row 22 price: $300

**Example #2: Invalid Section**

Enter ONE character for section - F: Front C: Center S: Side U: Upper

Z

Invalid section, program ending

**Example #3: Invalid Row**

Enter ONE character for section - F: Front C: Center S: Side U: Upper

S

Selected Side Section

Enter row

65

Invalid row, program ending

|  |
| --- |
| 1. In **one or two sentences**, restate the problem in your own words (inputs, outputs, rules). |
| 2. Identify **data types** needed for each input and for price:  Section:  Row:  Price: |
| 3. Evaluate Section Values that are inputted by the user  Create at least **2 more** test cases covering at least   * 1 valid inputs one upper case and one lower case example * 1 invalid input  |  |  |  | | --- | --- | --- | | Precondition | Postcondition | Code Notes | | Valid Input: f | Selected Front Section | Make upper case  If F, C S or U valid input so next prompt for row | | Invalid Input: 2 | Invalid section, program ending | else not F, C S or U valid end program | |  |  |  | |  |  |  |   Write an algorithm on paper for evaluating the section based on your learnings from the test cases. Take a picture and insert below. |
| 4. Evaluate Row Values that are inputted by the user to determine price. When this is entered they will already have a correct Section entered. Assume the user always enters a number.  Create at least **4** more test cases covering valid/invalid rows:   * Valid Value Row 16–30 cost $300 * Valid Value Row 31–60 cost $200 * Invalid Value Row less than 1 * Invalid Value Row less than greater than 60  |  |  |  | | --- | --- | --- | | Precondition | Postcondition | Code Notes | | Section f  Valid row:1 | Section F row 1 price: $450 | If rows 1–15 cost $450 | |  |  |  | |  |  |  | |  |  |  | |  |  |  | |  |  |  |   Write an algorithm on paper for evaluating the row based on your learnings from the test cases. Take a picture and insert below. |
| Explain the algorithm you chose **nested if** vs **multi-way if** vs **switch** for the two tasks above  1. Explain the algorithm you chose for evaluating the section input.  2. Explain the algorithm you chose for evaluating the row to determine price. |
|  |

# 3 Reflection

|  |
| --- |
| 1. Describe something from module 1 that you were able to understand better when writing your code for this assignment. |
| 2. Describe a moment you got “stuck” and how you worked through it (on your own, with a peer, or by searching resources). |
| 3. Describe something you didn’t understand from a lecture in module 02 that you understand better from writing the code. |
| 4. Describe something you have been doing outside of class that is helping you understand the concepts better. |