Grading Rubric for Programming

Dr. Semih Cal
Otterbein University
Department of Engineering, Computer Science, & Physics

1. Rubric Criteria

1.1 Program Specifications & Correctness

This criterion evaluates how well the program fulfills the provided specifications and executes its intended functions accurately. For example, if the task involves creating a program to sort a list of numbers in ascending order, the evaluation centers on whether the program accomplishes this sorting task correctly. An 'Excellent' program would proficiently sort lists of all types and sizes without errors. An 'Adequate' program might effectively sort most lists, although minor issues could arise in specific cases, such as empty lists or lists with duplicate numbers. A 'Poor' program might only manage to sort certain lists correctly, while a program categorized as 'Not Met' might not achieve any sorting or might not even run.

This criterion is extremely important because every submitted program needs to function correctly and follow the provided specifications. Submitted programs should reliably exhibit the intended behavior and produce accurate results for various types of inputs.

When facing uncertainty or ambiguity about a specification, it's recommended to seek clarification from the instructor rather than making assumptions.

Program Specifications and Correctness (60%)

- **Excellent:** 100%
- **Adequate:** 80%
- **Poor:** 50%
- **Not Met:** 0%

1.2 Readability

Readability involves how easy it is to understand code. This includes how the code is structured, the names used for variables and functions, and how whitespace and indentation are used. For example, in an 'Excellent' program, the structure is clear, each function has one clear purpose, variables have descriptive names (like 'sorted_list' instead of 'sl'), and everything is consistently indented and spaced for easy reading. An 'Adequate' program is generally easy to read but might have a few small issues. A 'Poor' program is hard to understand due to unclear names, disorganization, or inconsistent formatting. A 'Not Met' program is almost impossible to understand because it lacks any structure or organization. Code should be readable to both you and a knowledgeable third party.

Readability (15%)

- **Excellent:** 100%
- **Adequate:** 80%
- **Poor:** 50%
- **Not Met:** 0%

1.3 Documentation

Documentation refers to the comments and explanations in the code that help others understand what the code does and why certain decisions were made. In an 'Excellent' program, detailed comments explain each function's purpose, any tricky parts of the code, and the overall approach. An 'Adequate' program's comments are generally good but might miss some explanations. A 'Poor' program has minimal comments, leaving much unexplained. A 'Not Met' program lacks comments or explanations entirely. Every file you submit that contains source code should start with consistent header comments.

Documentation (15%)

- **Excellent:** 100%
- **Adequate:** 80%
- **Poor:** 50%
- **Not Met:** 0%

1.4 Assignment Specifications

This criterion evaluates adherence to the specific requirements provided for the assignment, such as file format, file naming conventions, or any other specific instructions. An 'Excellent' program meticulously adheres to all these specifications, while a 'Poor' program might overlook multiple specifications, and a 'Not Met' program would completely disregard them.

Assignment Specifications (10%)

- **Excellent:** 100%
- **Poor:** 50%
- **Not Met:** 0%

2. Example File Header

JAVA

```
/*
==============================================================================
Title         : Example.java
Description   : This is an example program.
Author        : Semih CAL (student number)
Date          : 01/01/2001
Version       : 1.0
Usage         : Compile and run this program using your Java compiler
Notes         : This example program has no requirements.
Java Version  : Specify your Java version
==============================================================================
*/
```

3. Grade Calculation

Each criterion contributes to an approximate percentage of the grade given to a programming assignment, as indicated by the percentage column. Points are assigned based on the evaluations of "Excellent," "Adequate," "Poor," and "Not Met."

For example, an assignment marked as "Adequate" in the Program Specifications & Correctness criterion, "Poor" for readability, and "Excellent" for all other criteria would receive a score of:

$$(0.8 \times 0.6) + (0.5 \times 0.15) + (1 \times 0.15) + (1 \times 0.1) = 0.805 = 80.5\%$$

** As a special case, if a program does not meet the specifications at all or is entirely incorrect, no credit will be received for the other criteria either.*