

## **COMP3520 – Software Engineering**

### **Assignment 4: Clean Code & Refactoring**

**Due Date:** November 16, 2025

#### **Objective**

This assignment will help you practice professional **clean coding** and **refactoring** skills used in real software teams.

You will refactor a poorly written Java program to:

- Improve readability & maintainability
- Follow coding standards & naming conventions
- Remove duplication and code smells
- Apply clean code principles
- Optionally add unit tests (TDD practice)

#### **Program Description**

A file named TempConvProgram.java is given to you. This program is a simple **temperature conversion console application**. It allows the user to convert temperatures between three different units:

- Celsius to Fahrenheit
- Fahrenheit to Celsius
- Celsius to Kelvin

The program:

1. Displays a menu of conversion options.
2. Accepts user input from the keyboard.
3. Performs the selected temperature conversion using mathematical formulas.
4. Prints the conversion result to the console.
5. Repeats until the user chooses to exit the program.

Currently, the code works, but it has **poor naming, formatting, repeated code, inconsistent styles, and weak input handling**, making it difficult to read and maintain.

Your task is to **refactor this code** to follow clean code and software engineering best practices **without changing the application's behavior**.

Note: Do **not** change the program's behavior. Only improve its structure & quality.

## Your Tasks

Refactor this program to follow clean code principles:

- Improve **method & variable names**
- Apply **Java naming conventions**
- Remove **hard-coded values (magic numbers)**
- Eliminate **duplicated prompts & printing**
- Improve **menu structure**
- Use **consistent formatting & indentation**
- Validate input & handle errors politely (It means **do not assume the user will always enter correct values**. Your program should check input before using it, and if something is wrong, respond nicely instead of crashing or printing confusing messages.)

**Example:** Check if input is a valid number

- ✓ Check if the menu choice is 1–4
- ✓ If input is invalid, show a clear message like:
- ✓ "Invalid choice. Please enter a number between 1 and 4."
- Add comments only where **necessary**

**Your refactored code must behave the same as the original!**

## Hints

- CtoF → convertCelsiusToFahrenheit
- x → userChoice
- Avoid repeating System.out.println("enter t")
- Use a **switch-case** instead of long if-else
- Extract menus / prompts into constants or methods
- Celsius can be negative — don't label it "wrong"
- Only comment **why**, not what

**Note: These are just some hints. You have to make sure that all the tasks listed above are fulfilled.**

**Bonus Marks :** Write 2–3 JUnit tests for conversion methods.

#### **Submission Instructions:**

Create a folder with your name and upload to Moodle. The following items must be part of this submission.

<b>Items to be submitted</b>
Refactored .java file
Explanation of changes (please see the format of report below)
(Optional) JUnit test file
GitHub repo link which includes both original and refactored code ( <b>must</b> )
Screenshot of the program running

#### **What to Explain in Your Report**

In one page:

- What changes you made & **WHY**
- Which **clean code principles** you apply  
(naming, DRY, readability, error handling, etc.)
- (Bonus) How you tested your methods