Assignment 5 – Test Driven Development

Name : Daniel Kim

GNumber : G01329040

Course : SWE437-001

Date : March 19, 2025

Collaboration Summary : Individual work

1. User stories

(1) As a user, I want to convert miles per hour to kilometers per hour so that I can understand speed measurements in different measurement systems.

(2) As a user, I want to convert temperatures from kelvin to celsius so that I can understand temperature measurements in everyday terms.

2. TDD tests

(1) Mph to Kph Documentation

User Story :

As a user, I want to convert miles per hour to kilometers per hour so that I can understand speed measurements in different measurement systems.

Acceptance Test:

I wrote an acceptance test that shows how a user would interact with our converter, user would choose "mph to kph", enter 65 mph, and expect to see 104.60 kph as the result. This test gives me a clear target for what I need to build.

Creating the UnitConverterApp :

Starting with the first part of the acceptance test, I needed to create the app itself. I wrote a simple test to check if I could create a UnitConverterApp object. When the test failed, I created the empty class to make it pass. This gave me the starting point for building the rest of the functionality.

Selecting Conversion Type :

Next, I needed to implement selecting a conversion type. My test sets "mph to kph" as the conversion type and checks if the app remembers this selection. This was a natural next step because after creating the app, the user's first action is selecting what conversion to perform.

Entering Value :

The next step is entering a value to convert. I wrote a test that enters 65.0 and verifies the app stores this value correctly.

Implementing Conversion Logic :

Next, I implemented the basic formula for conversion logic by adding a method that multiplies mph by 1.60934 to convert to kph.

Acceptance Test :

After building the feature step by step through TDD, I ran the acceptance test again and It passed.

Refactor & Acceptance Test :

I created a UnitConverter class for better structure and reusability. Then, I moved the conversion method into this class and extracted 1.60934 into a constant MPH_TO_KPH for improved readability and maintainability. After refactoring, I ran the acceptance test again, and it passed successfully.

(2) Kelvin to Celsius Documentation

User Story :

As a user, I want to convert temperatures from kelvin to celsius so that I can understand temperature measurements in everyday terms.

Acceptance Test :

I wrote an acceptance test that shows how a user would convert kelvin to celsius: select "kelvin to

celsius", enter 300.0 K, and expect 26.85 °C as the result.

UnitConverterApp Creation :

I first tested that I could create a UnitConverterApp instance. No new implementation was needed since the class already existed from our previous user story.

Conversion Type Selection :

Next, I tested that the app could handle selecting "kelvin to celsius" conversion. The existing selectConversionType method already worked for this new type.

Temperature Input :

Next, I verified the app could store a kelvin temperature value (300.0). The existing enterValue method handled this without any changes.

Kelvin to Celsius Logic :

Next, I wrote a test to check if 300 K correctly converts to 26.85 °C. I implemented the conversion by subtracting 273.15 from Kelvin.

Acceptance Test :

After building the feature step by step through TDD, I ran the acceptance test again and It passed.

Refactor & Acceptance Test :

I added the convertKelvinToCelsius method to the existing UnitConverter class and extracted KELVIN_CELSIUS as a constant for better readability and maintainability. After refactoring, I ran the acceptance test again, and it passed successfully.

3. Files

Canvas : Assignment_5_Daniel_Kim.zip (submitted)

Github : https://github.com/DanielKim777/Assignment_5