The George Washington University

Advanced Software Paradigms (CSCI 6221.10)

Homework Assignment #8

Date: 26 March 2024

Submitted By:

Abhiyan Sainju (G22510509)

Submitted to:

Professor Yih-Feng Hwang

Steps for execution:
- The provided program is written in **Python**. To run it, use any online Prolog compiler, for example, https://www.online-python.com/. This is the compiler used for testing the program.
- Copy the code from the attached source code.txt file and paste it into the code area in the online compiler.
- Click the "Run" button in the compiler.
- The program will prompt you to choose between building a laptop (Facade Design Pattern) or a house (Builder Design Pattern). Enter "laptop" or "house".
- If you choose "laptop", the program will guide you through the process of selecting the laptop specifications by prompting you to enter the number of cores, RAM amount, display size, and display resolution. After entering all the specifications, it will display the chosen laptop specifications. This part of the program follows the Facade Design Pattern.
- If you choose "house", the program will guide you through the process of selecting the house specifications by prompting you to enter the number of rooms, levels, and roof type. After entering all the specifications, it will display the chosen house details. This part of the program follows the Builder Design Pattern.
- To test the code again, simply run the Python file again and make a different choice (laptop or house).

Part 1: Facade Design Pattern (Structural)
1. The Facade Design Pattern provides a simplified interface to a complex system of classes. In this implementation, the ***LaptopFacade*** class acts as the facade, providing a single entry point to interact with the *subsystem classes (Processor, RAM, and Display)*. The ***SpecsPrinter*** class is responsible for displaying the laptop specifications.
2. The ***LaptopFacade*** class has a build_laptop method that coordinates the interactions with the subsystem classes to gather the necessary information (number of processor cores, RAM amount, display size, and display resolution) and then uses the ***SpecsPrinter*** class to display the laptop specifications.

Part 2: Builder Design Pattern (Creational)
1. The Builder Design Pattern separates the construction of a complex object from its representation, allowing the same construction process to create different representations. In this implementation, the ***HouseBuilder*** class is responsible for constructing the House object, while the Director class provides methods to construct specific types of houses (*build_two_room_house, build_four_room_house, and build_house_with_user_input*).
2. The House class represents the product being built, and the HouseBuilder class has methods to set the rooms, levels, and roof of the house. The ***Director*** class uses the ***HouseBuilder*** to construct different types of houses based on predefined specifications or user input.

Input Arguments:
- The build_product() function prompts the user to input their choice of product (laptop or house) and expects a string input ("laptop" or "house").
- If the user chooses "laptop" (Facade Design Pattern), the program will prompt for the following inputs:
  - Number of processor cores (positive integer)
  - Amount of RAM in GB (positive integer)
  - Display size in inches (positive number)
  - Display resolution (e.g., "1920x1080")
- If the user chooses "house" (Builder Design Pattern), the program will prompt for the following inputs:
  - Number of rooms (2, 3, 4, or 5)
  - If the number of rooms is not 2 or 4, it will ask for the following additional inputs:
  - Number of levels (1 or 2)
  - Type of roof (flat or pointy)

Expected Result:
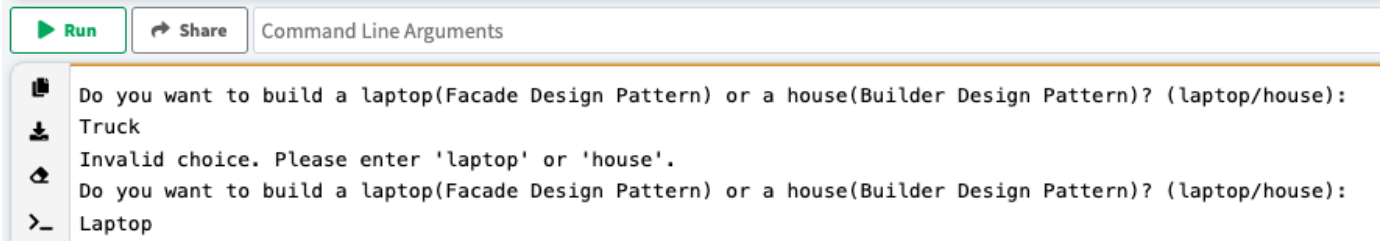- If the user chooses "laptop," the program will print the laptop specifications in the following format:

Your system has a [number]-core processor, with [RAM] GB RAM and a [display_size]-inch [display_resolution] display.

- If the user chooses "house," the program will print the house specifications in the following format:

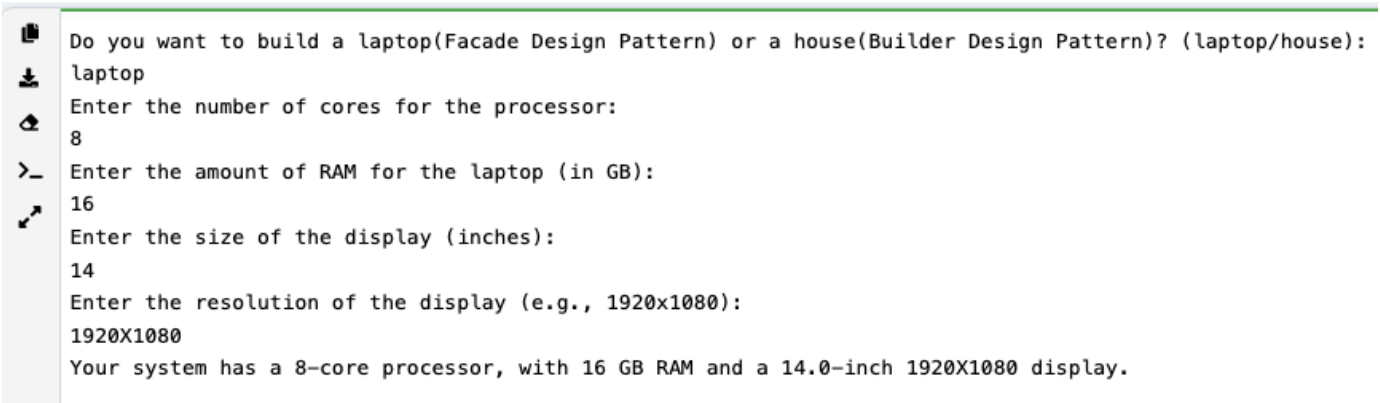House: [number] rooms, [number] levels, [roof_type] roof.

Screenshots:

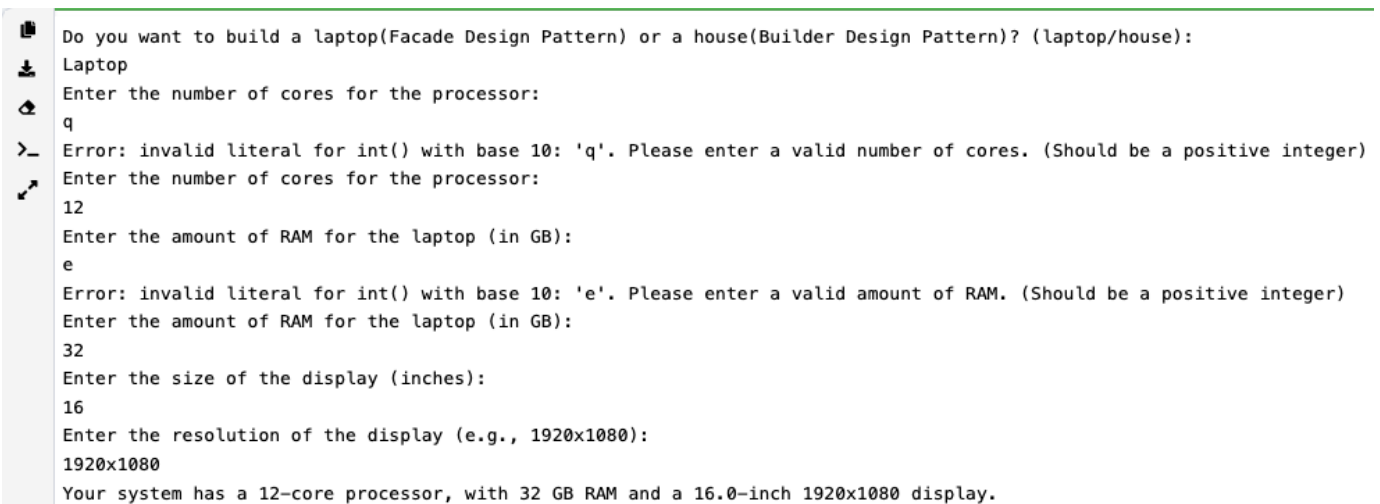1. Displaying the two design pattern options with error handling

Figure 1: Program displaying the two design patterns and an option to chode between them (with error handling)

2. When the Laptop option is chosen (Facade Design Pattern)

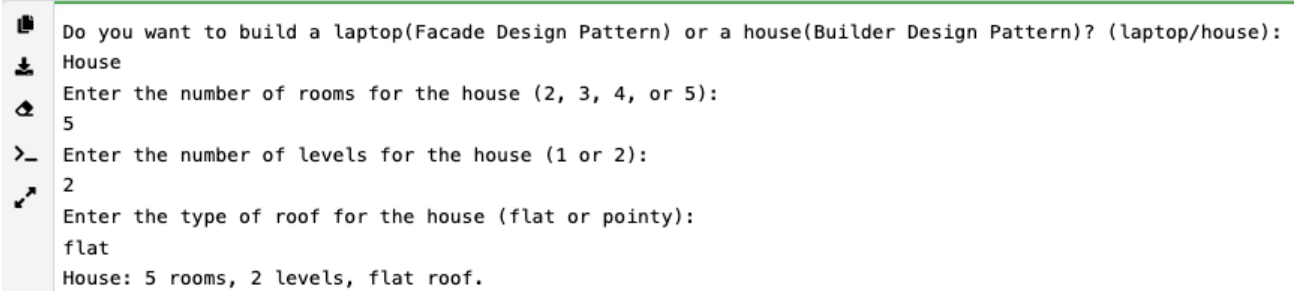*Figure 2: Result of the program when the laptop (Facade Design Pattern) option is chosen*

Figure 3: Result of the program when the laptop (Facade Design Pattern) option is chosen with error handling

3. When the House option is chosen (Builder Design Pattern)

```
Do you want to build a laptop(Facade Design Pattern) or a house(Builder Design Pattern)? (laptop/house):
House
Enter the number of rooms for the house (2, 3, 4, or 5):
4
House: 4 rooms, 2 levels, pointy roof.
```

Figure 4: Result of the program when the House (Builder Design Pattern) option is chosen with predefined specification
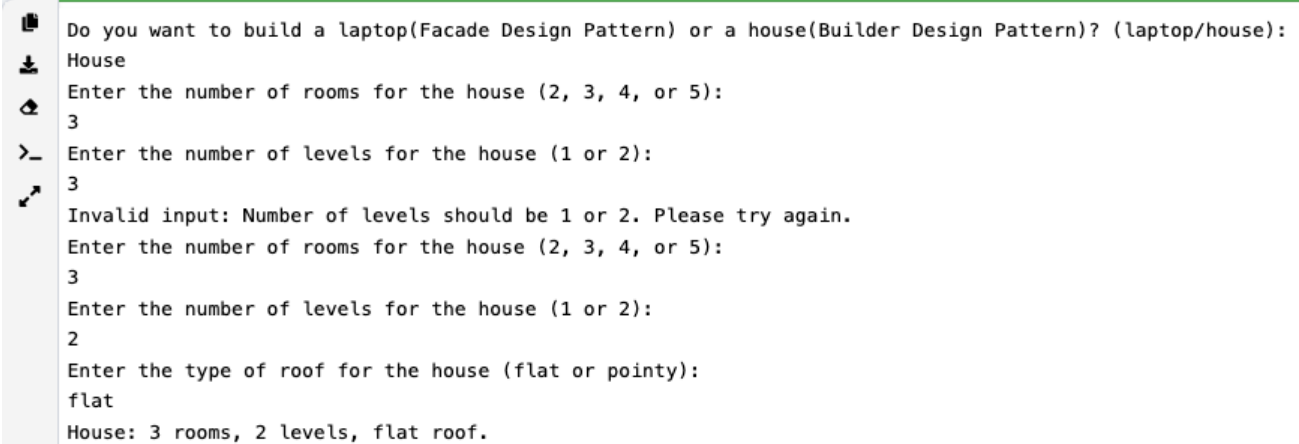
```
Do you want to build a laptop(Facade Design Pattern) or a house(Builder Design Pattern)? (laptop/house):
House
Enter the number of rooms for the house (2, 3, 4, or 5):
5
Enter the number of levels for the house (1 or 2):
2
Enter the type of roof for the house (flat or pointy):
flat
House: 5 rooms, 2 levels, flat roof.
```

Figure 5: Result of the program when the House (Builder Design Pattern) option is chosen with user-chosen specification

```
Do you want to build a laptop(Facade Design Pattern) or a house(Builder Design Pattern)? (laptop/house):
House
Enter the number of rooms for the house (2, 3, 4, or 5):
3
Enter the number of levels for the house (1 or 2):
3
Invalid input: Number of levels should be 1 or 2. Please try again.
Enter the number of rooms for the house (2, 3, 4, or 5):
3
Enter the number of levels for the house (1 or 2):
2
Enter the type of roof for the house (flat or pointy):
flat
House: 3 rooms, 2 levels, flat roof.
```

Figure 6: Result of the program when the House (Builder Design Pattern) option is chosen with error handling