ASSIGNMENT 2
COEN 275

Pujitha Kallu
ID : W1653660
pkallu@scu.edu

---

1). Create a Factory pattern for a document generation system where the system supports generating documents of different formats, such as PDF, Word, and HTML. Each format has unique behaviors, and some documents may include advanced features like watermarks, metadata, or tables.

Implement the following features:
- A Document abstract base class with a generate() method.
- Concrete classes PDFDocument, WordDocument, and HTMLDocument, each implementing the generate() method with format-specific behaviors.
- Each concrete class should optionally add advanced features like addWatermark(), addMetadata(), or addTable() based on provided configuration.
- A DocumentFactory class with a createDocument method that takes the document type (e.g., "PDF") and configuration options (e.g., "addWatermark", "addMetadata") as arguments and returns the appropriate Document object.
- Client code to generate documents with different formats and configurations, demonstrating the use of the factory to control document type and feature inclusion.

Solution; Output

```
pujitha@Pujithas-MBP-2 Question1 % g++ -std=c++14 main.cpp -o main

pujitha@Pujithas-MBP-2 Question1 % ./main

Generating PDF document
Adding watermark to PDF: Classified
Adding metadata to PDF:
Creator: Alex Taylor
Title: Sample PDF Document
Adding table to PDF:
Name Age
Pujitha 29
kallu 34

Generating Word document
Adding watermark to Word: Preliminary
Adding metadata to Word:
Creator: Morgan Lee
Title: Word Document Example

Generating HTML document...
Adding metadata to HTML:
Creator: Jordan Smith
Description: HTML Document Sample
pujitha@Pujithas-MBP-2 Question1 %
```

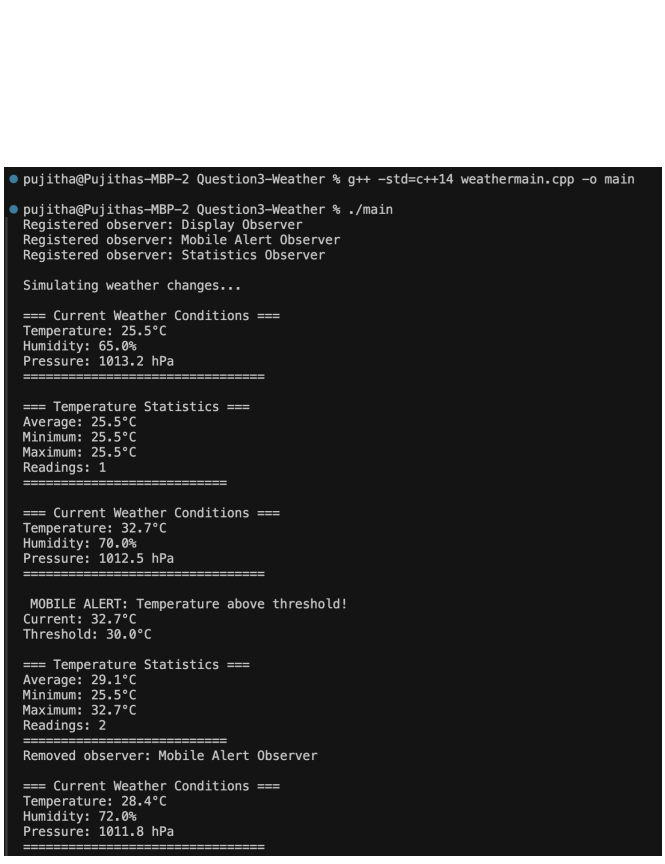
2). Singleton Pattern for a Logger Service Design a Singleton class called Logger to manage logging for an application.
The logger should:
- Have a private constructor and a getInstance() method to return the Singleton instance.
- Include methods for logging messages at various levels (e.g., info, warning, error).
- Store all log entries in memory and provide a method to retrieve all logs.
- Test the logger by creating multiple log messages froam different parts of your code.

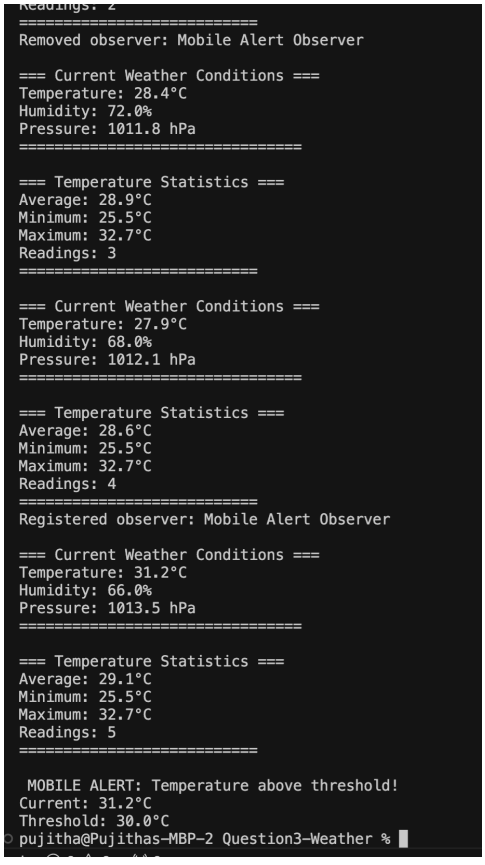
Solution: Output

```
pujitha@Pujithas-MBP-2 Question2-Logger % g++ -std=c++14 loggermain.cpp -o main

pujitha@Pujithas-MBP-2 Question2-Logger % ./main
Testing singleton behavior:
logger1 address: 0x100318000
logger2 address: 0x100318000
Instances are the same: true

Testing logging functionality:
[2024-11-17 16:43:03] INFO: Application started
[2024-11-17 16:43:03] WARNING: Memory usage is high
[2024-11-17 16:43:03] ERROR: Failed to connect to database
[2024-11-17 16:43:03] INFO: User logged in

All logs:
[2024-11-17 16:43:03] INFO: Application started
[2024-11-17 16:43:03] WARNING: Memory usage is high
[2024-11-17 16:43:03] ERROR: Failed to connect to database
[2024-11-17 16:43:03] INFO: User logged in

Warning logs only:
[2024-11-17 16:43:03] WARNING: Memory usage is high

After clearing logs:
Log count: 0
pujitha@Pujithas-MBP-2 Question2-Logger %
```

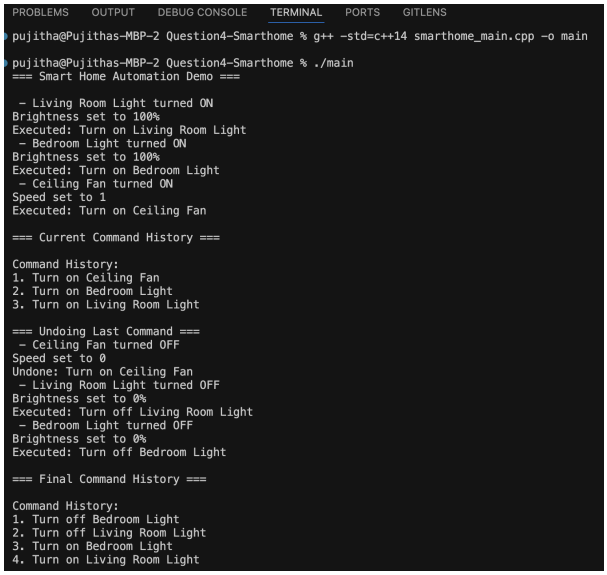
3) Implement an Observer Pattern for a Weather Monitoring System using C++
Develop an Observer pattern for a weather monitoring system with the following classes:
- WeatherData: Tracks temperature, humidity, and pressure. It should have a vector of observers, methods for adding and removing observers, and a method to notify all observers when data is updated.
- WeatherObserver (abstract base class): Includes a virtual update(WeatherData&) method.
- Implement concrete observers:
- DisplayObserver: Updates a display with the latest weather data.
- MobileAlertObserver: Sends an alert to mobile users when temperature crosses a threshold. StatisticsObserver: Logs temperature data for trend analysis.
- Write C++ code to implement these classes, register and deregister observers, and simulate data updates.

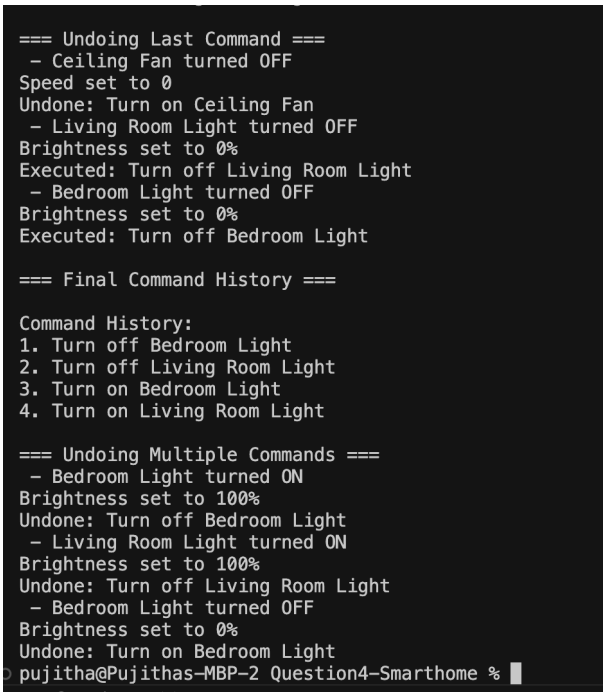
Solution: output



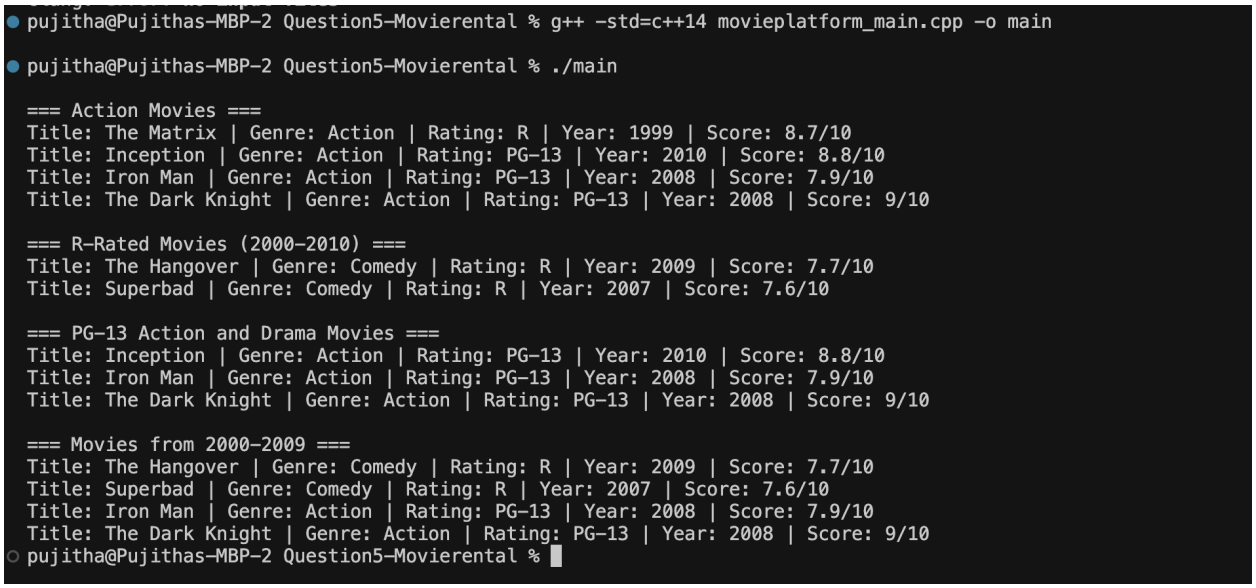


4. Command Pattern for a Smart Home Device Controller Create a Command pattern for controlling smart home devices.
Implement the following:
- A Command interface with an execute() method.
- Concrete command classes to turn LightOnCommand, LightOffCommand, FanOnCommand, and FanOffCommand.
- A SmartHomeController class that stores a command history and can execute and undo commands.
- Write a client code to create and execute commands to control light and fan devices, demonstrating the undo feature.
Solution: output





5. Design an Iterator pattern for an online movie rental platform that categorizes movies by genres, such as Action, Drama, and Comedy. Implement the following features:
- Allow users to iterate over movies within a specific genre.
- Provide an option to iterate over movies across multiple genres with specific ratings (e.g., movies rated PG-13 or R).
- Implement a filter to retrieve movies based on release year.
- Write a client code to showcase iteration over movies within a genre and across genres.
Solution: output



References:

1. https://www.geeksforgeeks.org/command-pattern/
2. https://refactoring.guru/design-patterns/cpp
3. Class material
4.