TECHNISCHE HOCHSCHULE DEGGENDORF

Application Design

Master of Applied Computer Science

Winter Semester 2022-23

Project Topic

Sensor Network for Local and Global Climate observations

Kajal Chellappan - Matriculation No: 12204385

Sonya Thomas – Matriculation No: 12203795

Date of Submission: 29.01.2023

Index

---

1. Motivation

There has been an unprecedented rise in sea level, mainly due to human-induced climate change, which is expanding the world oceans and melting the ice sheets and glaciers resulting in coastal erosion and floods. Since the beginning of the 20th century, the global mean sea level (GMSL) has risen by about 16–21 cm. Sea level has been projected to rise by 0.6–1.5 m by 2100. There will also be significant variation in regional sea-level changes due to differences in gravitational processes, ground subsidence, current dynamics, salinity, and heating effects. The impact of current and future sea-level rise would be devastating, particularly to coastal environments. Rising frequency and intensity of flooding, coastal erosion, and changes in coastal aquifer quality and groundwater are some of the physical impacts of SLR on coastal communities. Hence, we have taken this project to facilitate the prevention of this issue by monitoring the rise in sea level using sensor networks.

2. Goal of the Application

To design an application that provides an innovative approach to the rising sea level and the prediction of floods using sensor networks. The goal is for the user to be able to monitor the local and geospatial sensors and calculate the rise in sea level over a period and use that information to calculate the prediction of floods using flood calculation Algorithms.

3. Abstract requirements definition

1) Context

The Theme of the project is Sensor Network for local and global climate observations.
The problem cause address here is Rise in sea level due to the rise in temperature resulting in floods and coastal erosion. We are going to design an application that provides an innovative approach monitor one of the many problems with climate changes that is the rising sea level and the possibility of floods using sensor networks.

2) Requirements

- The estimated effort for the implementation of the project is 20 man-years and the delay is less than a year, i.e., you can work with at least 20 people in parallel. Note that the usual calculation is 100.000EUR per person-year so this is a substantial amount of effort.
- You need a more complex software architecture than a monolithic program.
- At least some part of the project requires a Human-Machine Interface that can be implemented with a touch-screen GUI interface following the guidelines on utility, usability, UX and QoE.
- At least one of project's entities is controlled by two concurrent input sources (i.e., some sensor that detects the presence of a pedestrian at a traffic light (source 1) a smart police car on duty approaching the same traffic light (source 2)).
- Your project is suitable for implementing one set of design patterns. The sets are detailed below.

3) Design Pattern Set

Choose one of the following sets of design patterns. You must implement each design pattern at least once in your project except for the one in brackets which is optional and provides bonus points.

Set 1: Composite, Memento, Builder, Adapter, Observer, (Event-based asynchronous)

Set 2: Visitor, Abstract Factory, Interpreter, Singleton, Interceptor, (Reader/Writer Lock)

Set 3: Chain of Responsibility, Decorator, Factory, Proxy, Flyweight, (Thread Pool)

Set 4: Monitor Object, Composite, Decorator, Strategy, Iterator Pattern, (Active Object)

4. Overview – Scenario and requirements

The goal is for the user to be able to monitor the local and geospatial sensors and calculate the rise in sea level over a period and use that information to calculate the prediction of floods using flood calculation Algorithm. For this we have come up with an application idea which will help the users to user the sensor network data to monitor the increasing sea level and predict the floods. We have selected 5 different scenarios each mapping with 1 design pattern of set 1.

1. **Find Sensor and Reading Sensor Details locally and geospatially** (using composite design pattern)

   This use-case is about as a user, we want to find the sensor available for sea level detection and read the data.

2. **Display details of configuration and simultaneously display the prediction of floods**(using Builder design pattern)

   This use case is about as a user, we want to display the details of configuration in graph, chart format and geo map format and simultaneously calculate the flood prediction using algorithms and display the prediction of floods.

3. **Configure frequently monitored sensors** (using Adapter design Pattern)
   This use-case is about as a user, we want to add a configuration with the set of sensors that we use frequently.

4. **Configuration Management** (using Memento design pattern)
   This use case is about as a user, we want to manage the configuration added to monitor sensors.

5. **Send warning alert to all users when the parameter limit exceeded** (using observer design pattern)
   This use-case is about as a user, we want to get notified of an alert warning, when the sea level limit exceeds.

We have developed prototype of our application such that it can be completed and become fully functional with future enhancements and work. We have described all possibilities of utilising 20 man-years in modular blocks. In Section “Work proposal for 20 Person – Separation of concern”.
We have created scribbles, wire frames and projected GUI for a use case where we have taken into consider guidelines on utility, usability, UX and Quality of Experience. In detail all this described in Section “GUI”.

We have studied all 24-design pattern to understand working of each pattern, and decided that Set 1 is most suitable set where we can fit majority of our ideas. So, for our Project we have chosen Set 1 and covered all design pattern (Composite, Memento, Builder, Adapter, Observer) in our use case scenarios.

We have implemented all application function in JAVA. As a demonstrative part we have kept some console input/output images in the section “Implementation output”.

5. Usefulness of Design Pattern:

a. Usefulness of Composite design pattern

In the Composite Pattern enables a class in high hierarchy levels to represent individual objects as well as group objects. Basically, there are three components apart from the client: the Component, the Leaf, and the Composite. In our Project we have Local Sensors act as Leaf and Geospatial Sensor act as Composite. How to get the sensors? How to add new sensors? How to find the Sensors? Composite design pattern is an easy and right solution to handle this.
This property is going to be same for individual sensors. We can have a tree structure here. So, in this case we have a hierarchy of sensors. Composite pattern is best suit here as it provides object creation in hierarchical way and each composite element or leaf have same characteristics/properties.

So here in our case we have implemented and displayed the outputs. To get, find and add Sensors, programmatically it become vital that each tree element can be called or used uniformly for implementation which is possible with composite design.

b. Usefulness of Builder Pattern in the project:

In the Sensors we have to demonstrate in different ways that are Graph representation, Map representation and Chart representation, we need to build the representations and get the results. For this purpose, builder design pattern best suits.

Builders have 3 main elements. Dashboard, Builder interface, Builder. Dashboard using specific builder call methods based on which whole representation is created. Builder interfaces have list of methods which are build and get the product result. Builders are having actual method/function for adding different kind of element for each method.

In this way we can define different concrete builders to have different types that used for fetching the result for the corresponding representation.

c. Usefulness of Adapter design pattern

In our above-described use case, there are lots of sensor and the sensor has its own data, that data can be shown in XML format and convert that into JSON format we need the adapter pattern.

Within one sensor adapter pattern is needed., in order to retrieve the results of a specific sensor, certain parameter must be passed in the constructor so that results can be calculated and convert the data in JSON Format.

d. Usefulness of the Memento pattern in our software:

The Memento pattern used in the configuration management, design pattern that enable saving and restoring the previous state (undo) of an object without revealing the details of its implementation.

Here in our software architecture, A caretaker requests a memento from originator, originator creates the memento, caretaker save the memento in a Map. A caretaker sends back a memento to originator, originator restores the internal state. Originator creates Memento/ assign memento attributes/ retrieve memento attributes.

e. Usefulness of the Observer design pattern in our project:

I used this pattern especially in this use case because the Observer pattern has two advantages; when a change in one object requires changing the others and we don't know how many objects need to be changed, also one object should be able to notify other objects without the need to know who these objects are. In our software architecture, when the sea water level exceeded alert warning message coming out, which Alarms have been triggered, and send the notification to all other observers.

6. Software Architecture

Our software architecture must be implemented as a distributed system since it is handling real time experiments and data. some of the experiment might be a critical type which doesn't tolerate any point of failure, thus the business logic layer as well as data storage must be in different places to ensure redundant behaviour in case of failure of one node, presentation layer can be set at the Institutions monitoring the sensor information. So, we recommend 3-tier architecture for our application, where the Presentation level is at the user level and business logic layer which is responsible for data validation, calculation, data modification and interfacing with the database to be separated from the data storage layer.

7. Use Case scenarios :

1. **Find Sensor and Reading Sensor Details locally and geospatially** (using composite design pattern)

Sensors have local and Geospatial sensors. In Application design some sensors activities need to perform which can be leaf or composite. Each sensors have their own parameters user can be able to find the sensors available for sea-level detection and read the sensors without any difficulties. Also, user can be able to add the sensors.

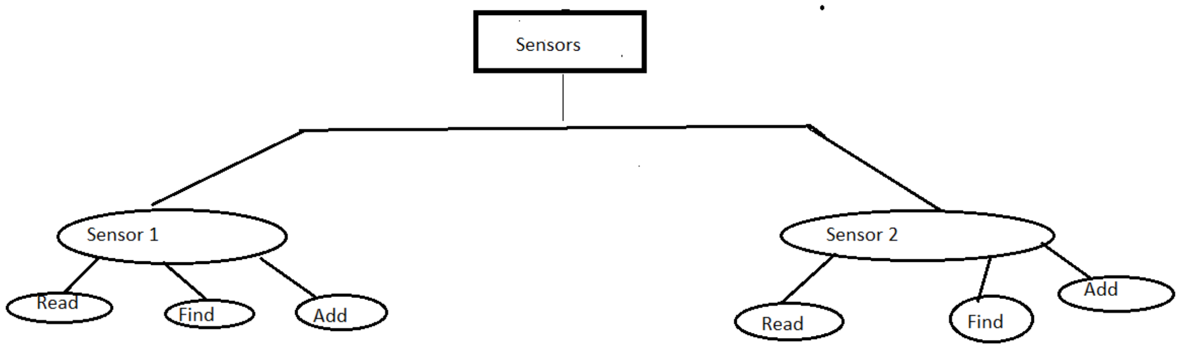


Diagram 1: finding the sensors, reading the sensor details, and adding the sensors (Composite design pattern)

As depicted in diagram sensors can make multiple sensors of variable parameters. Here example shows Sensor 1 and Sensor 2 made up of respectively.During the process the individual sensor can find, read and add the sensor details.

Group objects have same property of find, read and add sensor details. Function for composite object(group) define to represent collective find and read sensor details.

2. Display details of configuration and simultaneously display the prediction of floods (using builder design Pattern)

Each user, want to display the details of configuration in graph, chart format and geo map format and simultaneously calculate the flood prediction using algorithms and display the prediction of floods. So, fix structure design is not sufficient to fulfil each representation creation. Solution is to have some framing tool where user can pick the right entity frommultiple possibilities for different sensors.

In our use case we have made up of "graph", "chart" and map „format. Using Builder pattern, we have 1 director and multiple (3 in our case) builder. Director is responsible for creating different representation by calling different builder. Different builder represents different style and format to include within each block. In our case we named 3 types of builders as "Graph", "Chart" and "Map" to understand it easily with builder types or patterns. There are more possibilities which can be created as a future work. In code example all parts have some string indication which are supposed to be replaced with appropriate function and designing call for realization.

3. . Configure frequently monitored sensors (Adapter pattern use case)
In our scenario, the users can add the configuration with the set of sensors that are used frequently. Using this use case user can configure a set of sensors, he/she wants to monitor frequently and also use the information together simultaneously to calculate the flood prediction using a flood prediction algorithm.
Let's take the daily statistics as an example, configuration ID, Time duration, Region ID are recorded every day. So, at the end of the day user can get these statistics and can get ideas like what sensors are configured and should they provide more frequently. So overall they can get idea about how many

sensors are configured accordingly.

Also the user should be able to preview the information from the sensors they are going to configure for monitoring

4. **Configuration Management (**Memento pattern use case)

The user is able to manage the configurations added in the previous use case.
That is the user should be able to remove the configuration that the user no longer need or add a new one or edit an existing one adding newly added sensors to an existing configuration. While the removal of configs we have used Memento such that the caretaker sends back a memento to originator, originator restores the internal state through a GUI that is provided to them with a good User experience.

Users can store their Sensor details, once they stored, they are able to store these details for later use, later they can restore this to do future, or they can just update and use new details. Such that if a config has been removed the user should be able to move to any other state already stored in the memento map.

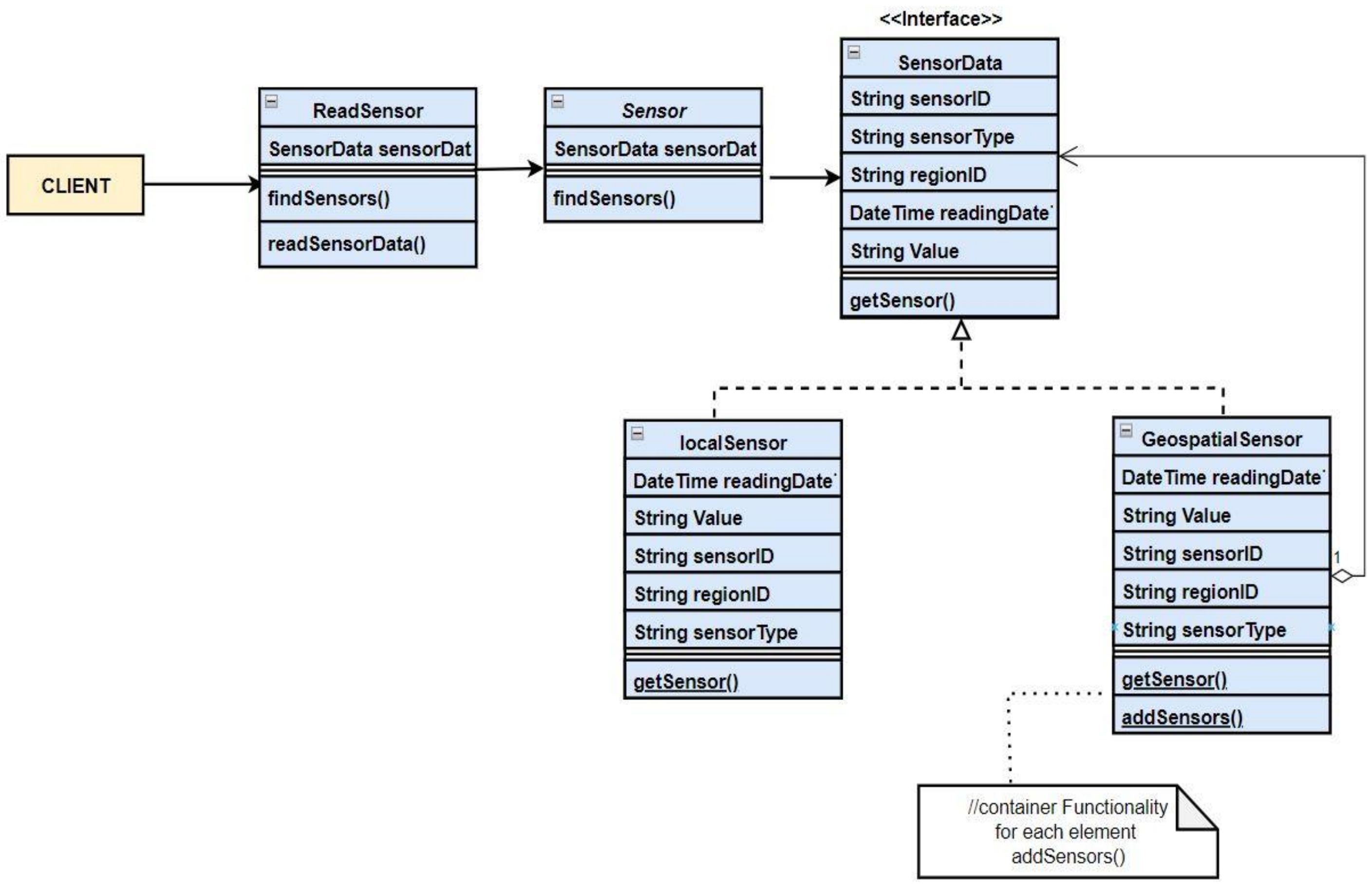
5. **Send warning alert to all users when the parameter limit exceeded (**Observer pattern use case)

General use case description:
In the use case our project supports to send the warning alert message on demand whenever an allotted sea level limit exceeds, user are notified with an alert message using a user interface that has a good user experience design, hence it is mandatory that user are able receive alarms about the sea level limit exceeded.

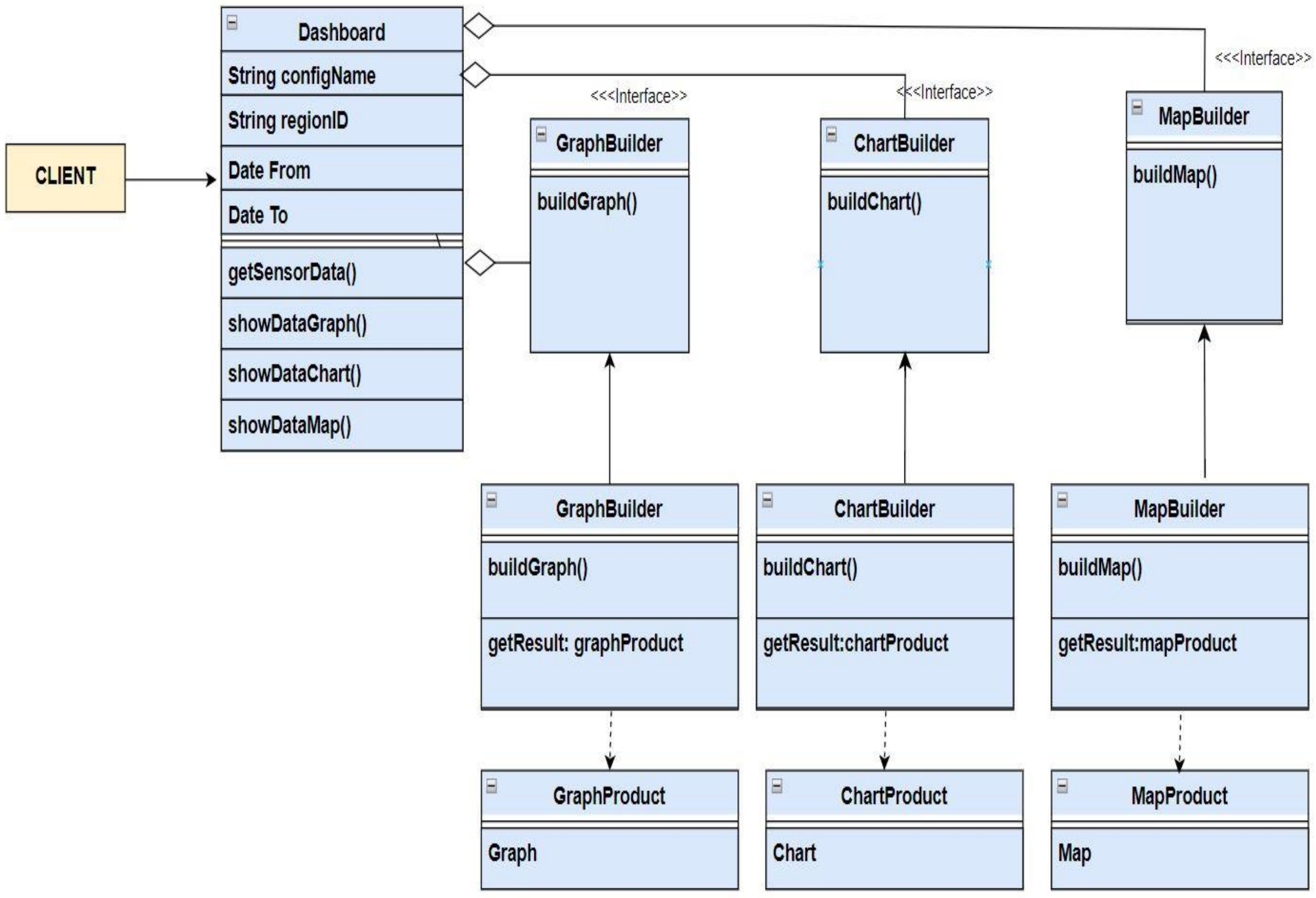
Implemented use case:
The different stakeholders and higher officials such as managers or even the users are added as observers of the alert system and then so in such a situation when the sea level rises to a level exceeding the allotted limit users are able to receive an alert message stating that "Sea Level Exceeded limit!!!" as a warning for a upcoming disaster

8. Class Diagram

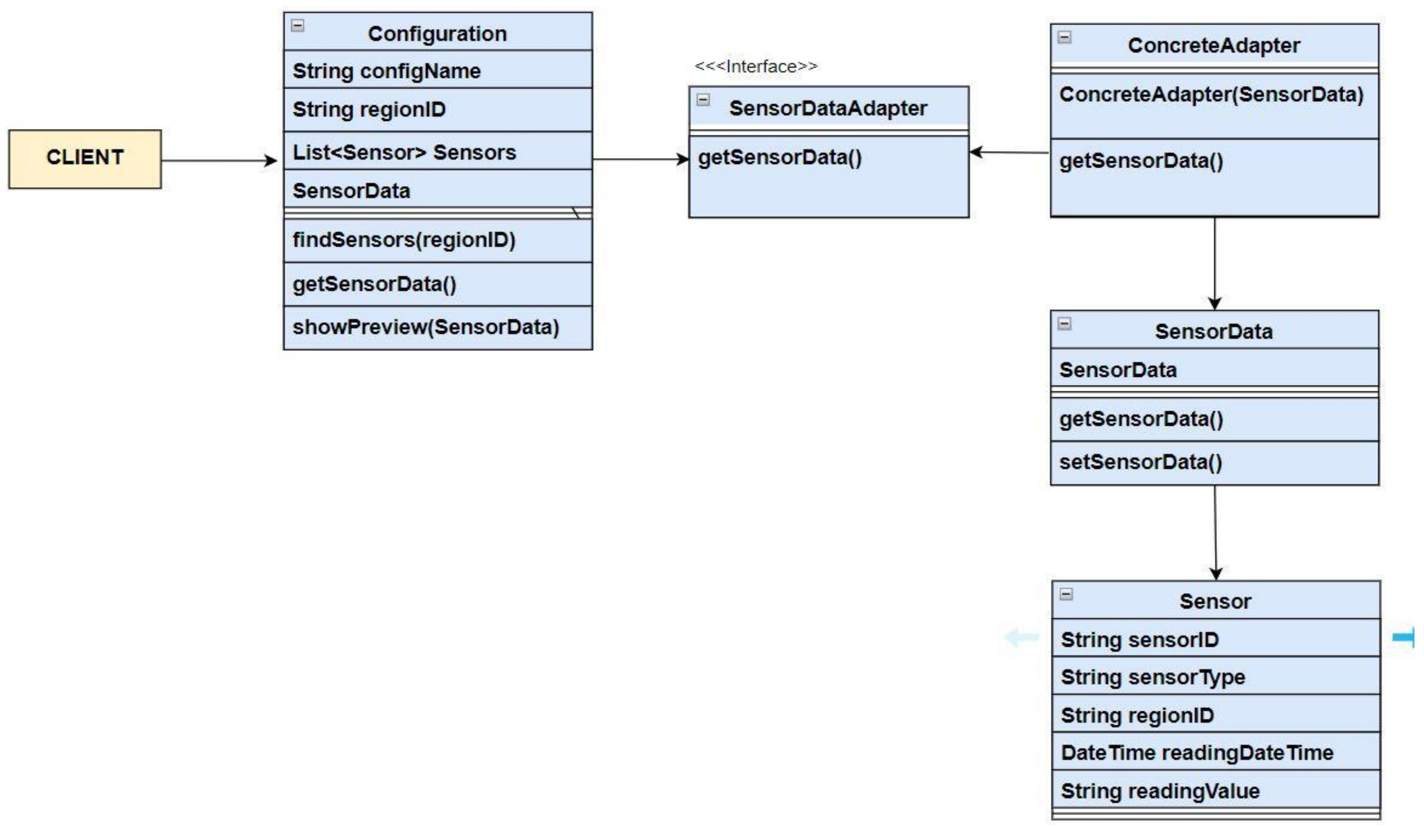
1. Reading Sensor Data (using composite design pattern)



2. Dashboard screen (Builder Pattern):

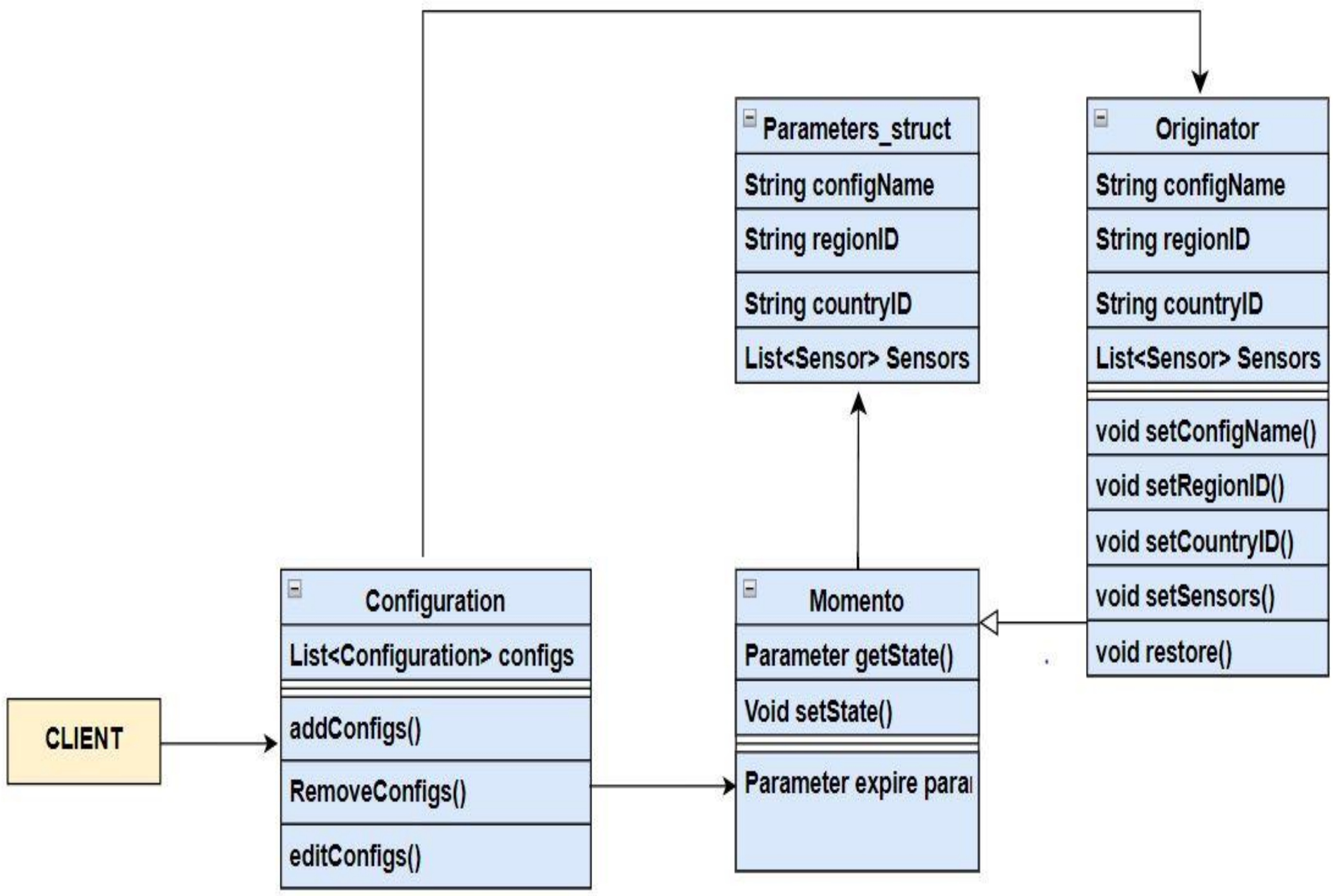


3. Configuration screen (Adapter design Pattern)

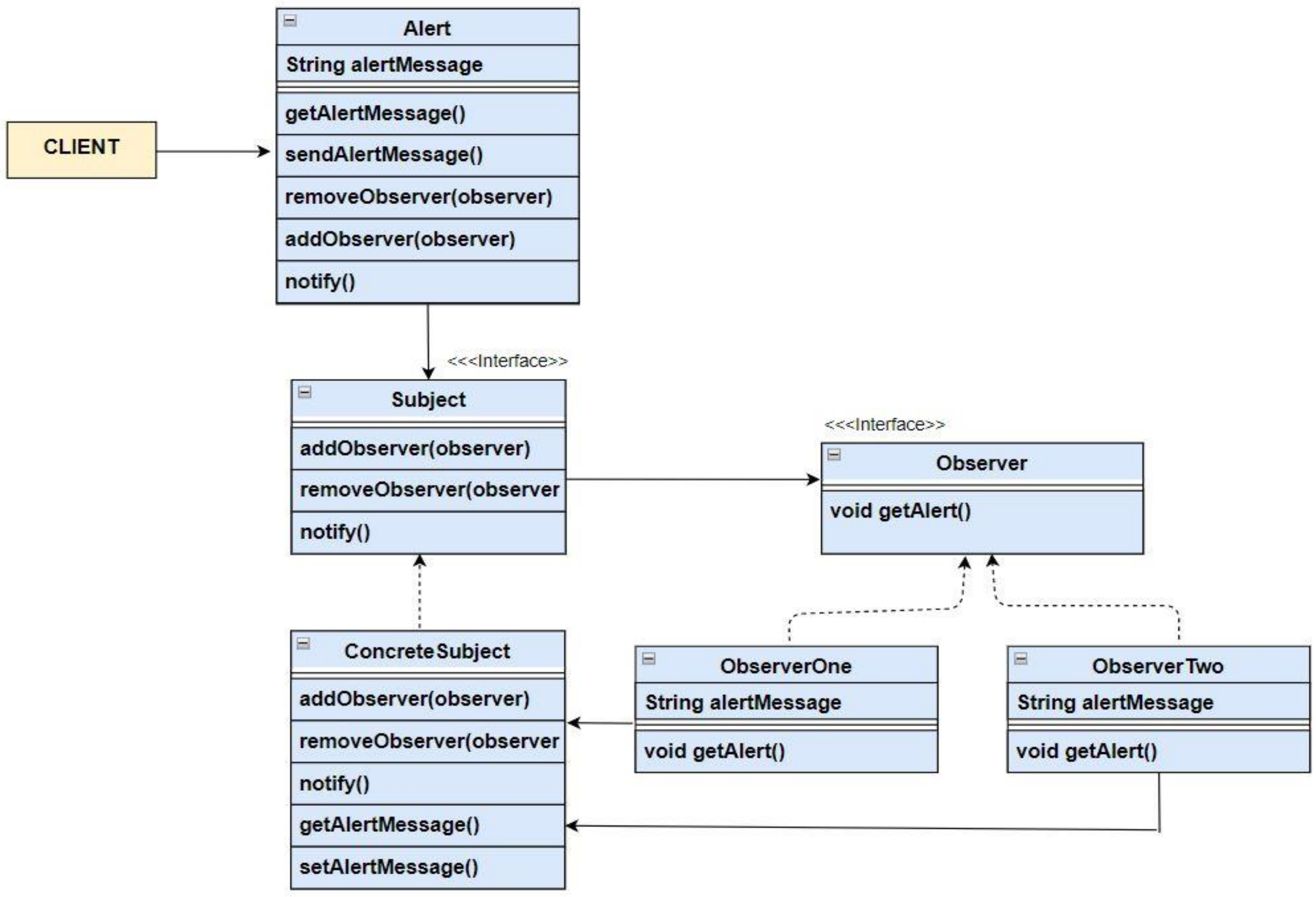


4. Configuration Management (Memento Pattern):

Collaborations: A caretaker requests a memento from originator, originator creates the memento, caretaker save the memento in a Map.
A caretaker sends back a memento to originator, originator restores the internal state.
Originator creates Memento/ assign memento attributes/ retrieve memento attributes.



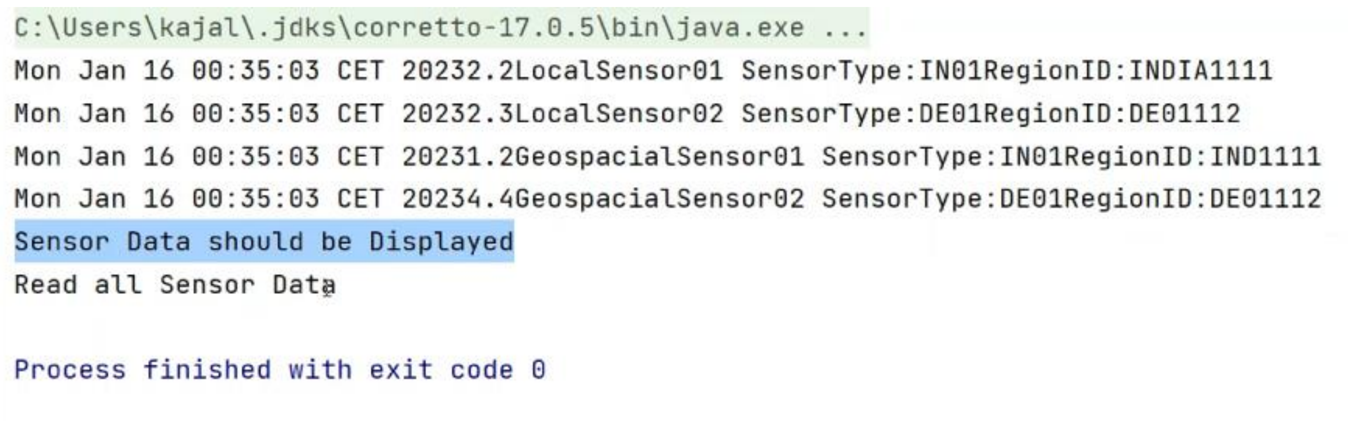
5. Class Diagram (Observer):



9. Output demonstrations

1. Read Sensor Data (using composite design pattern)

The following output demonstrate the reading of sensor information from local sensor and also from a group of geospatial sensors with a composite.
.



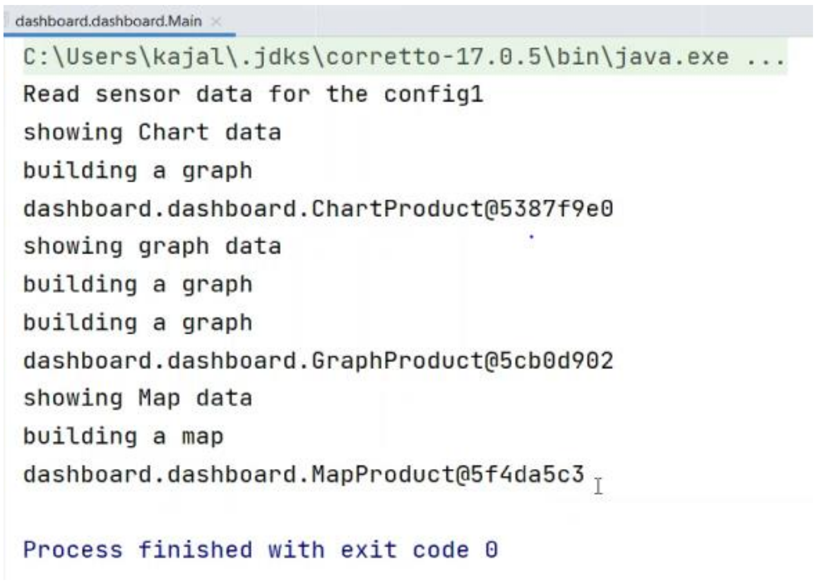
```
C:\Users\kajal\.jdks\corretto-17.0.5\bin\java.exe ...
Mon Jan 16 00:35:03 CET 20232.2LocalSensor01 SensorType:IN01RegionID:INDIA1111
Mon Jan 16 00:35:03 CET 20232.3LocalSensor02 SensorType:DE01RegionID:DE01112
Mon Jan 16 00:35:03 CET 20231.2GeospacialSensor01 SensorType:IN01RegionID:IND1111
Mon Jan 16 00:35:03 CET 20234.4GeospacialSensor02 SensorType:DE01RegionID:DE01112
Sensor Data should be Displayed
Read all Sensor Data

Process finished with exit code 0
```

Leaf Node : Local Sensor reads information from local sensor
Composite Node : Geospatial sensors read sensor information from multiple sensors

2. Dashboard Monitoring (Builder Output demonstration):

The output shows created 3 views using builder pattern, each having different kind of view forms such as graphview, chartView and mapView.



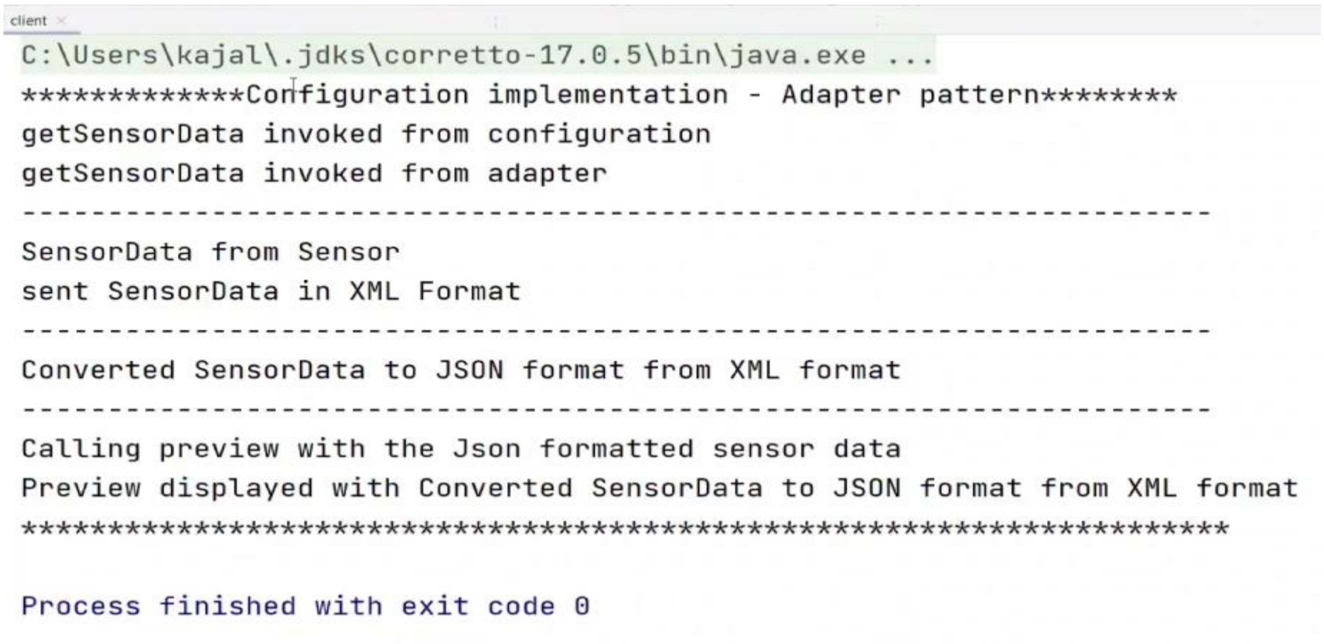
dashboard.dashboard.Main

```
C:\Users\kajal\.jdks\corretto-17.0.5\bin\java.exe ...
Read sensor data for the config1
showing Chart data
building a graph
dashboard.dashboard.ChartProduct@5387f9e0
showing graph data
building a graph
building a graph
dashboard.dashboard.GraphProduct@5cb0d902
showing Map data
building a map
dashboard.dashboard.MapProduct@5f4da5c3

Process finished with exit code 0
```

3. **Configuration Screen** (using Adapter design Pattern)

The output shows an adapter has been created which adapts the input from a sensor and converts it into format accepted by the preview functionality to show the preview of the data.

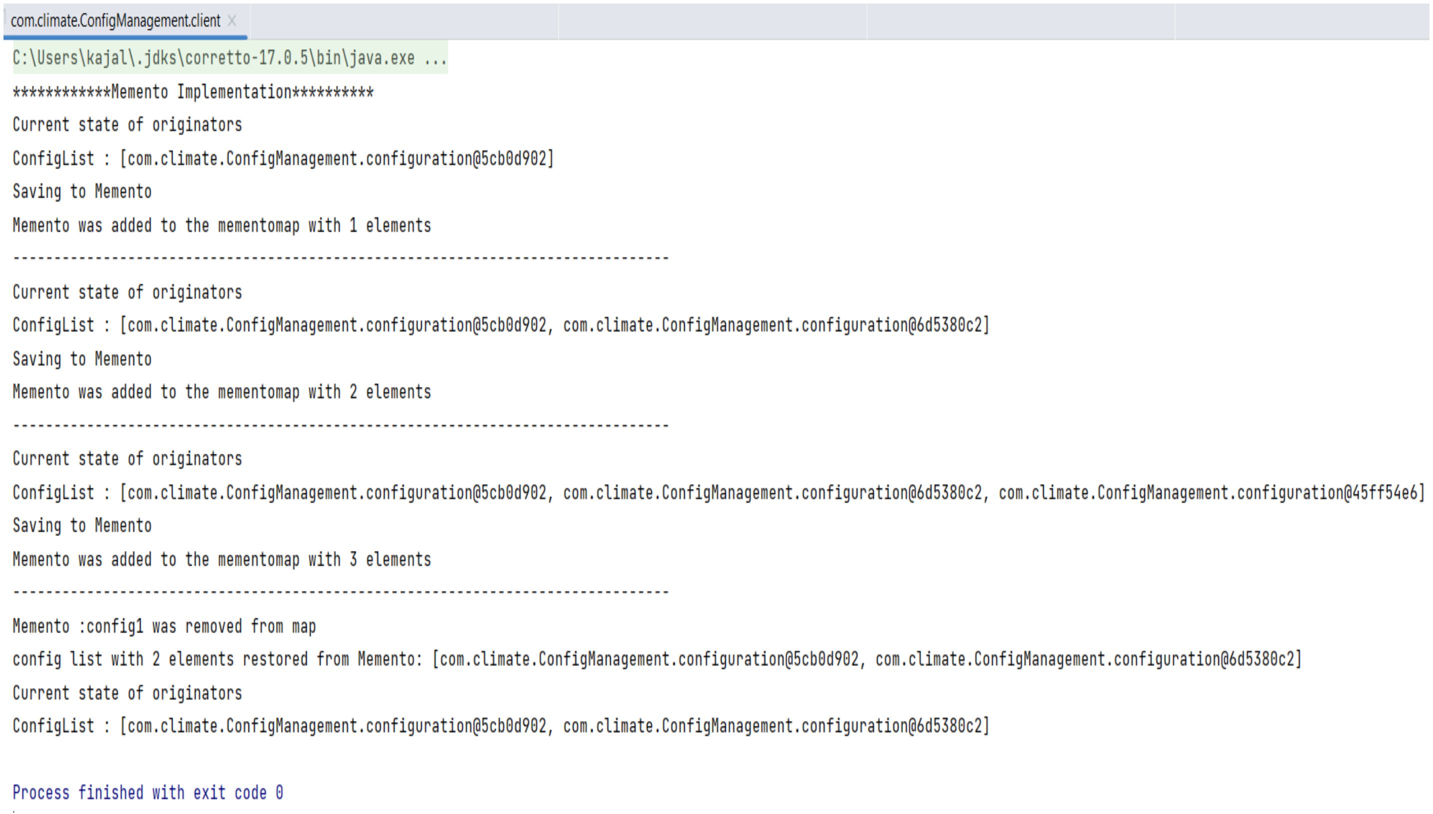


```
C:\Users\kajal\.jdks\corretto-17.0.5\bin\java.exe ...
*************Configuration implementation - Adapter pattern********
getSensorData invoked from configuration
getSensorData invoked from adapter
-----------------------------------------------------------------------
SensorData from Sensor
sent SensorData in XML Format
-----------------------------------------------------------------------
Converted SensorData to JSON format from XML format
-----------------------------------------------------------------------
Calling preview with the Json formatted sensor data
Preview displayed with Converted SensorData to JSON format from XML format
**********************************************************************

Process finished with exit code 0
```

4. Configuration Management Output (Memento):

In the following program output, memento pattern is used and on removing the configs the memento is used to change from one state to another using the originator.

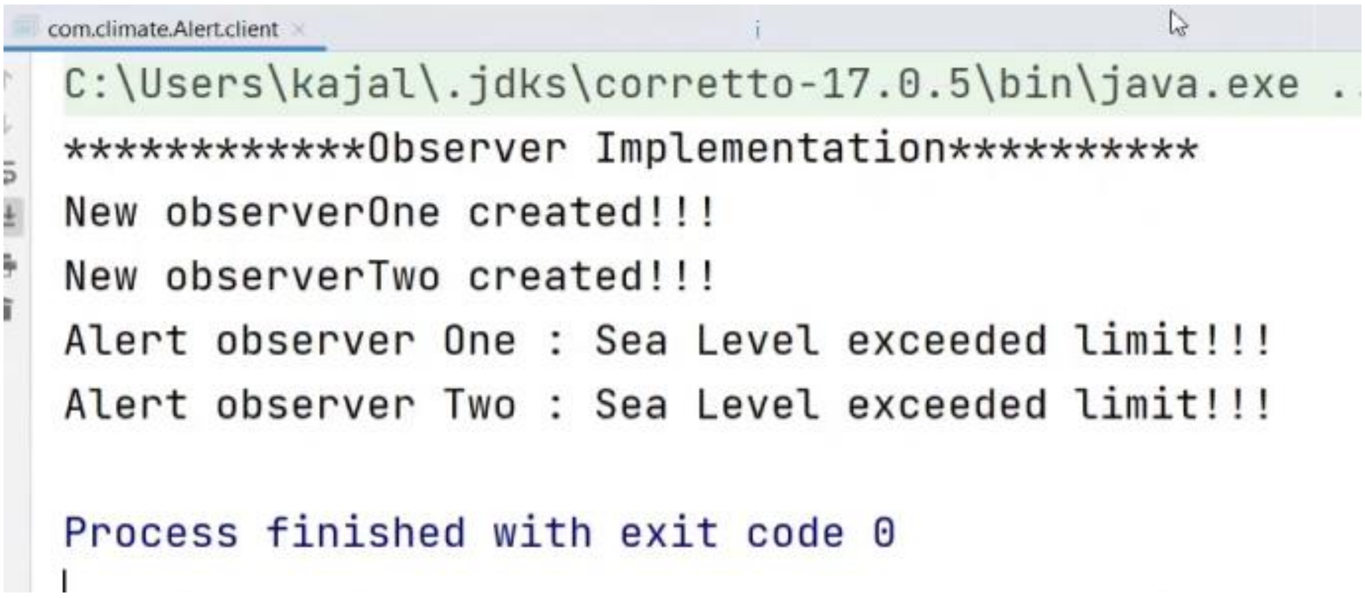


```
C:\Users\kajal\.jdks\corretto-17.0.5\bin\java.exe ...
************Memento Implementation*********
Current state of originators
ConfigList : [com.climate.ConfigManagement.configuration@5cb0d902]
Saving to Memento
Memento was added to the mementomap with 1 elements
--------------------------------------------------------------------------------
Current state of originators
ConfigList : [com.climate.ConfigManagement.configuration@5cb0d902, com.climate.ConfigManagement.configuration@6d5380c2]
Saving to Memento
Memento was added to the mementomap with 2 elements
--------------------------------------------------------------------------------
Current state of originators
ConfigList : [com.climate.ConfigManagement.configuration@5cb0d902, com.climate.ConfigManagement.configuration@6d5380c2, com.climate.ConfigManagement.configuration@45ff54e6]
Saving to Memento
Memento was added to the mementomap with 3 elements
--------------------------------------------------------------------------------
Memento :config1 was removed from map
config list with 2 elements restored from Memento: [com.climate.ConfigManagement.configuration@5cb0d902, com.climate.ConfigManagement.configuration@6d5380c2]
Current state of originators
ConfigList : [com.climate.ConfigManagement.configuration@5cb0d902, com.climate.ConfigManagement.configuration@6d5380c2]

Process finished with exit code 0
```

5. Alert Screen (Observer design pattern)

When the sea level rises exceeding the allotted limit then a alert message is sent to all the observers of the alert message



```
C:\Users\kajal\.jdks\corretto-17.0.5\bin\java.exe ..
************Observer Implementation**********
New observerOne created!!!
New observerTwo created!!!
Alert observer One : Sea Level exceeded limit!!!
Alert observer Two : Sea Level exceeded limit!!!

Process finished with exit code 0
```

7. Graphical User Interface

Utility:

Utility has been taken for, since the functionalities that are required to conduct theexperiment are available:

1. Finding the Sensors and read the sensor data frequently.
2. Provide details (Status) of the Graph/Map/Charts Representation in the Dashboard screen.
3. Setting the parameters to any desired value.
4. Start/pause/stop/restart the experiment.
5. Start the controller.
6. Flood Alert button to get the alert message.
7. List out the Sensors in the window.
8. Observing the output from the Sensors.
9. Storing the set of parameters that lead to an acceptable controller behaviour.
10. Restoring to default parameters set in case the disturbance rejection controller has anundefined destructive behaviour.
11. Removing a state from the set container once another state has proven to be a betterchoice.

Usability:

The extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use.

Learnability:
1. First time users should be able to accomplish their tasks.

Effectiveness:
2. the buttons are quite big and far from each other thus reducing the error percentage ofpressing the wrong button to the minimum.
3. Buttons are self-explanatory, Store for storing the parameters set, restore to restore toa preferred set, remove to remove a set from the container and to start the process.

Error tolerance:
A restore button has been provided to restore to a stable state in case if the tested statecaused an unwanted behavior.

Efficiency:
1. there are a few buttons that need to be pressed to achieve the required functionality; the user only needs to set the parameters by the arrows and then click start to start the process.
2. Buttons are self-explanatory, Store for storing the parameters set, restore to restore toa preferred set, remove to remove a set from the container and to start the process.So, buttons are quite intuitive thus increasing the Usability of the User interface (UI)

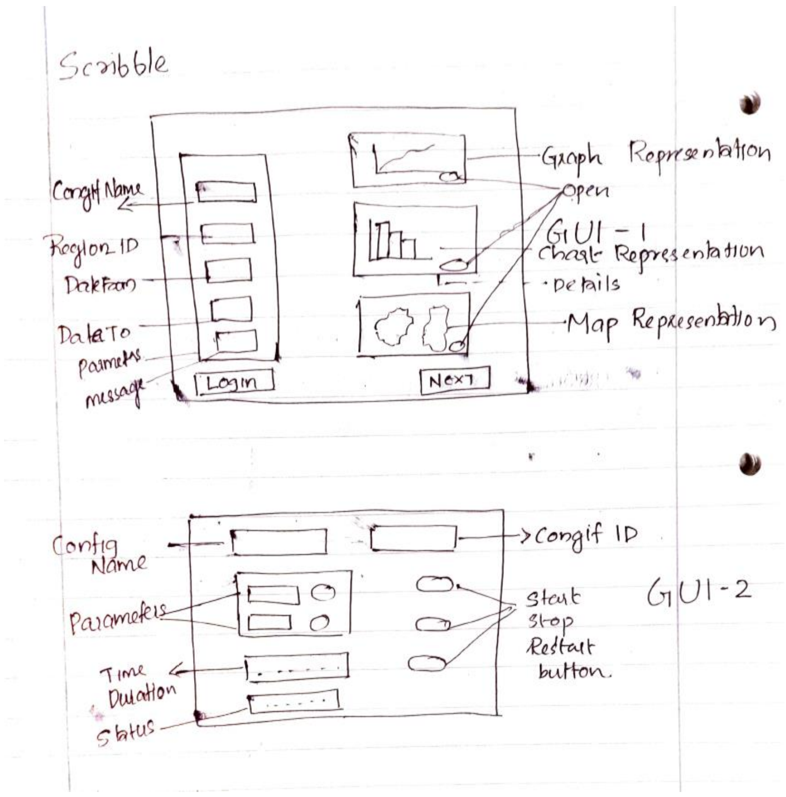
Satisfaction:
The aesthetic and proper layout and the readable typography of the GUI makes the UImore engaging and satisfying to the user.
It consists of four simple layouts: one for the input parameters,one for plotting to observe the output of the disturbance rejection controller and onefor history of acceptable parameters set.

Ease of learning:
As mentioned above there are few self-explanatory buttons and the layouts of each section are separated, so the user Knows which section to refer to if he needs to execute a specific functionality.

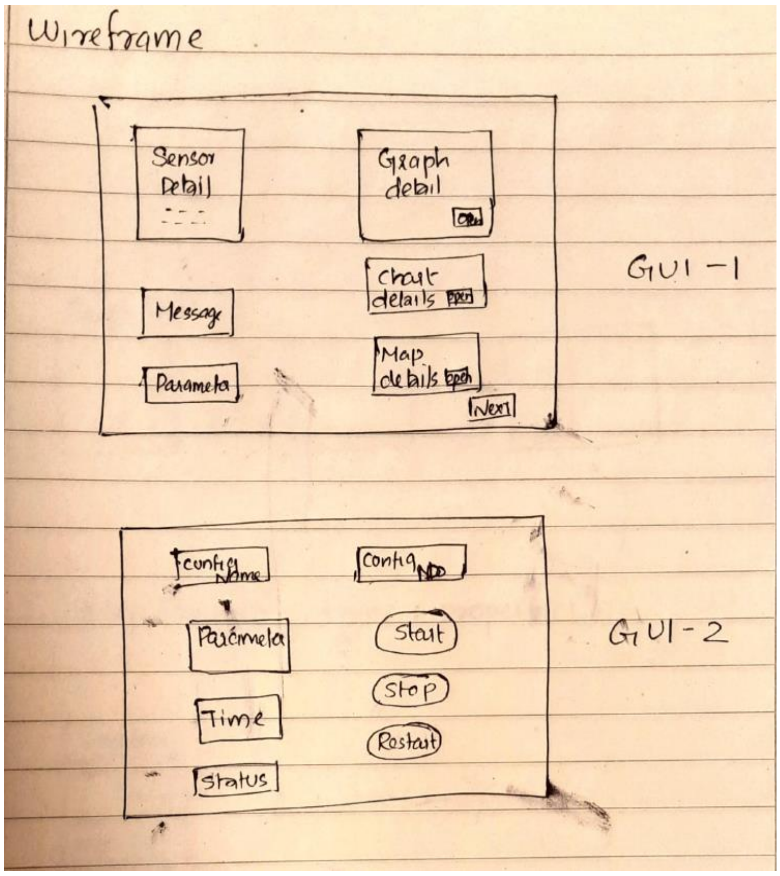
UX:
1. Analyse and understanding the context of use also checking status is included to know availability of sensors.
2. For setting time, scroll type is included to avoid misunderstanding in minutes/hours ortyping error.
3. The utility (functions required by user) as well as usability are fulfilled in the design of the GUI thus leading to a better user experience since emotional fulfilment a satisfaction has been sensed by the user.

Quality of experience is also achieved since the GUI is designed only for the user group havingno extra functionality other than the utility itself. It provides sufficient data of the status of the sensor parameters including configuration Name, Configuration ID, Region ID and Duration from and Duration To.

Sample GUI for Use Case 6:
Scribble



Wire Frame



11. Work proposal for 20 Person – Separation of concern

1 Year task distribution for 20 Person

1 Person - Design Architecture

Design architecture is going to be responsible for designing the software architecture with all design patterns set and application work. Any further advancement and modification need to be under 1 specific architecture.

2 Person - Finding sensors and data collection and then Database handling in the later phase of development.

2 Person – Read Sensor data use case (Composite pattern)

Sub-task includes:

- Leaf and composite variable and method creation and later added advancement
- Find Sensors and collect data
- Handling local sensor selected and data handling
- Handling list of sensors for geospatial sensor data handling
- Code Coverage testing and code quality testing

3 Person – Monitoring Dashboard use case (Builder pattern)

Sub-task includes:

- Builder implementation for different formats of view including graphview, chart view, map view
- Using the set of sensors for the calculation of algorithms for flood prediction and etc. and displaying it with the view
- Code Coverage testing and code quality testing
- Implementing the future advancements

2 Person – Configuration screen use case (Adapter pattern)

Sub-task includes:

- implementation for adapter and the adaptee
- Implementation of client configuration with the sensor selection and preview
- Code Coverage testing and code quality testing
- Implementing the future advancements

2 Person – Configuration Management use case (Memento pattern)

Sub-task includes:

- Implementation for memento and originator and save the states of all the originators
- Implementation of configuration manage class with management of all the configurations added, addition, removal and most importantly updation of configurations
- Code Coverage testing and code quality testing

- Implementing the future advancements

2 Person – Alert screen use case (Observer pattern)
Sub-task includes:

- handling the alert and subject classes
- handling all the observer classes
- Code Coverage testing and code quality testing
- Implementing the future advancements

2 Tester for all different sub-use cases of application

1 Graphical User Interface developer

2 Profile creator and manager responsible for all profile operation such as register, login and logout

1 person to check and maintain the health check of sensors.

12. <u>Work distribution</u>

Task List:

We divided 24 design patterns in 2 set, and we both studied 3-3 different design patterns from each set. In total, we both studied individually 12 different design patterns with focusing on idea for Project.

For the implementation divided as follows:

Sonya: Part 1: Composite, Builder
Kajal: Part 2: Memento, Adapter, Observer

The following task were done individually:

Kajal Chellappan:

1. Referred Research papers and websites for Topic Selection and content gathering.
2. Creation of class diagrams for all Use cases.
3. Implementing the use cases 3, 4 and 5
4. Work distribution for 20 man-year - Separation of concern

Sonya Thomas:

1. Implementation of use cases 1 and 2
2. Designing and creating wireframe and scribble for GUI and UI/UX
3. PPT Creation for presentation

The following task were done by both:

1. Discussion on Use cases creation for the selected topic
2. Code review – Alternate for each other.
3. Analysis on GUI and Usability, User Experience and Quality of Experience.
4. Report Documentation.

13. Difficulties and improvement of the project

Kajal Chellappan:

The First difficult part for me was choosing the project topic and the Problem cause which we were going to focus on as a selling point of our application also making sure that we are thinking in the prospect of the user's needs and requirements.
Defining 20 man-year task is another complex difficulty. since I had to put myself in the shoes of a team leader, it needs a lot of detail perspective of final output application and to make sure that the work is neither underly nor overly assigned to an individual.

The project can be improved once we have started working on the real time development and only then we can figure out the mistakes in the initial design and make enhancements. We can analyse more on the technologies and tools that would be more efficient for the real time data exchange and operations. With sensors we have to constantly make sure that the health check is working fine for all the sensors as a single faulty sensor can drastically affect all calculations it is included in.

Also, after the initial designing, I came to a conclusion that in the configuration management if the user is going to directly do the modifications, instead of the memento pattern we can use a normal list, but it can be only confirmed once we start the real time development.

Overall, a great experience and exposure to designing an application from scratch and developing a mind of a project leader. Definitely helped in developing a leadership quality and mindset.

Sonya Thomas:

Firstly, the idea behind the topic was limited on how to exactly implement it, but later on after a discussion with the professor, I understood it correctly. The most difficult thing was to analyse and decide on how to split the work between 20 members since I have to put myself in the individual responsibility of a team leader, which was a bit difficult assuming that I have no experience in the development industry.
The part which is the UX design pattern. GUI design took a lot of time and effort but finally I was able to crack it within the given deadline. In case of programming, Java was chosen and there was some gap in my java knowledge, so I had to learn some major programming concepts for the coding section.

During the presentation also, I was able to improve the design pattern because of the feedback from the professor.

Overall, it was a great exposure for me to learn and do a new project. I am also thankful to my team member for their coordination and communication.

14. ACKNOWLEDGEMENT

The Project is Guided by Prof. Dr. Marcus Barkowsky,
Faculty of Applied Computer Science
THD - Technische Hochschule Deggendorf, Deggendorf, Germany

15. REFERENCES

1. https://ieeexplore.ieee.org/document/8250467
2. https://www.google.com/url?sa=t&source=web&rct=j&url=https://www.ysi.com/parameters/level&ved=2ahUKEwjP65Ogs8P8AhWKcfEDHVVKB0MQFnoECCIQAQ&usg=AOvVaw0MYx_Hw-rNMQ9WiqmZ8yBN
3. https://www.google.com/url?sa=t&source=web&rct=j&url=https://os.copernicus.org/articles/18/997/2022/&ved=2ahUKEwjP65Ogs8P8AhWKcfEDHVVKB0MQFnoECEMQAQ&usg=AOvVaw2BbWja3Zvmr0gRmuhNxD1E
4. https://www.mssanz.org.au/modsim09/C4/ghobakhlou.pdf
5. https://www.researchgate.net/publication/228349096_Sensor_data_acquisition_for_climate_change_modelling
6. https://www.mdpi.com/2072-4292/13/18/3587