Team 1 Project Phase 3: GRASP Patterns

|  |  |  |
| --- | --- | --- |
| **GRASP Pattern** | **Class/Line** | **Group Member** |
| Creator | Vaccines.java  line 33 | Tory Provenzano |
| Information Expert | Cases.java  lines 28 and 88 | Hiba Rehman |
| Higher Cohesion | VaccineTableDisplay.java  lines 9-11 and 17-19 | Matthew Gagnon |
| Low Coupling |  | Karanveer Sidhu |
| Controller | CovidDashboardMain.java  lines 50-56 | Matthew Muscedere |
| UML illustrations |  | Varunteja Katakam |

**Pattern: Creator**

* The purpose of this pattern is to help identify which class is responsible for instantiating another class.
* In our code (file: Vaccines.java), the Vaccines class is responsible for managing the data regarding the provincial vaccine totals and sending them to VaccineTableDisplay.java to display the data on the program.
  + Line 33 of Vaccines.java (String[][] rows = new String[provinceVaccineTotals.size()][2]) creates a new instance of provinceVaccineTotals in the form of an array, which is readable by VaccineTableDisplay.java

**UML Illustrations:**

[Class Diagram](https://lucid.app/lucidchart/87573e68-2b92-43e5-83c7-c4a99ec576e5/edit?viewport_loc=-964%2C-382%2C4002%2C1491%2C0_0&invitationId=inv_e230d2e0-9f5f-4f49-9a29-2b39b12cb092)

[Sequence Diagram](https://lucid.app/lucidchart/87573e68-2b92-43e5-83c7-c4a99ec576e5/edit?viewport_loc=-1051%2C-906%2C3920%2C1460%2C~QhuedBJ6we9&invitationId=inv_e230d2e0-9f5f-4f49-9a29-2b39b12cb092)

[UML](https://lucid.app/lucidchart/87573e68-2b92-43e5-83c7-c4a99ec576e5/edit?viewport_loc=-788%2C-149%2C2740%2C1021%2C-RhuQDqI08aJ&invitationId=inv_e230d2e0-9f5f-4f49-9a29-2b39b12cb092)

**Pattern: Low Coupling**

* The purpose of this pattern is to minimize the dependencies between classes to make the system easier to maintain, modify, and understand.
* In our code (file: CovidDashboardMain.java), the CovidDashboardMain class is responsible for managing the main graphical user interface (GUI) and displaying the various views, such as Cases, Deaths, Active upon button clicks. Instead of directly creating and managing logic for each view, CovidDashboardMain interacts with these classes through simple object creation.

**Example:** On line 50 in CovidDashboardMain.java:

addButton(menuPanel, "COVID-19 Cases", e -> new Cases());

This demonstrates low coupling because CovidDashboardMain only triggers the creation of a Cases object when the corresponding button is clicked but does not contain any logic for how Cases operates internally. The details of the Cases class, such as its data handling, UI logic, or other operations, are encapsulated within the Cases class itself.

This approach minimizes the dependency of CovidDashboardMain on the internal workings of Cases, which promotes loose coupling and makes it easier to modify or extend Cases without impacting the main dashboard class. Other views such as Deaths, Active, and Recovered follow a similar approach, enhancing maintainability throughout the project.

**UML Illustrations**

[Class Diagram](https://lucid.app/lucidchart/87573e68-2b92-43e5-83c7-c4a99ec576e5/edit?viewport_loc=-1587%2C-222%2C3599%2C1341%2C8diuj2AQv5Zt&invitationId=inv_e230d2e0-9f5f-4f49-9a29-2b39b12cb092)

[Sequence Diagram](https://lucid.app/lucidchart/87573e68-2b92-43e5-83c7-c4a99ec576e5/edit?viewport_loc=-942%2C-806%2C3652%2C1361%2COgiu~~zSSwGW&invitationId=inv_e230d2e0-9f5f-4f49-9a29-2b39b12cb092)

[Use Case Diagram](https://lucid.app/lucidchart/87573e68-2b92-43e5-83c7-c4a99ec576e5/edit?viewport_loc=-759%2C-750%2C4929%2C1836%2C8hiuxZ0ZGt2V&invitationId=inv_e230d2e0-9f5f-4f49-9a29-2b39b12cb092)

**Pattern: Information Expert**

* The purpose of this pattern is to identify which class is responsible for managing the data to perform a specific task. It is the class that already has the necessary information to fulfill a responsibility.
* In our code (file: Cases.java), the Cases class is responsible for managing the data related to COVID-19 stats. It contains the logic for reading and aggregating case data sorted by province. This makes it the most suitable place for managing this information
  + Line 28 in Cases.java: Map<String, Integer> provinceData = readAndAggregateData("Code/src/data/cases\_hr.csv"); shows that this class is responsible for gathering and organizing COVID-19 case data into a table/structure that can be used for displaying later.
  + Line 88 in Cases.java: provinceData.put(province, provinceData.getOrDefault(province, 0) + cases); aggregates the case data by province and updates the provinceData map. This method of data aggregation makes the Cases class the Information Expertfor this program.
* It’s important to note that this is just one example of an Information Expert in our prorgram. Other classes, such as Vaccines, Reported, and Deaths, might also be Information Experts for their respective data, each managing and processing information relevant to their respective areas. Multiple Information Experts allows for more flexibility and maintainability.

**UML Illustrations**

[Class Diagram](https://lucid.app/lucidchart/27904971-2236-4dcd-8389-b4374baf515e/edit?viewport_loc=-2361%2C-431%2C5325%2C1984%2C0_0&invitationId=inv_714acc0f-7526-48ce-b246-4f4483a02f0b)

[Sequence Diagram](https://lucid.app/lucidchart/27904971-2236-4dcd-8389-b4374baf515e/edit?viewport_loc=-1481%2C-220%2C3187%2C1187%2C5wiuwnHdbV3N&invitationId=inv_714acc0f-7526-48ce-b246-4f4483a02f0b)

[Use Case Diagram](https://lucid.app/lucidchart/27904971-2236-4dcd-8389-b4374baf515e/edit?viewport_loc=-1698%2C-660%2C5360%2C1997%2CHyiuewG62MeD&invitationId=inv_714acc0f-7526-48ce-b246-4f4483a02f0b)

**Pattern: Controller**

* Purpose: This pattern delegates responsibilities for handling system events to a controller class, often representing the use case or session of an interaction.
* Implementation: In the provided codebase, the CovidDashboardMain class in CovidDashboardMain.java acts as a controller. It orchestrates the initialization and interaction of various UI components like Cases, Active, and Deaths.

Example in Code:  
  
**File**: CovidDashboardMain.java  
  
// Add buttons for different views

Line 50: addButton(menuPanel, "COVID-19 Cases", e -> new Cases());

Line 51: addButton(menuPanel, "Active Cases", e -> new Active());

Line 52: addButton(menuPanel, "Total Deaths", e -> new Deaths());

Line 53: addButton(menuPanel, "Total Recovered", e -> new Recovered());

Line 54: addButton(menuPanel, "Reported Cases", e -> new Reported());

Line 55: addButton(menuPanel, "Vaccine Administration Data", e -> new Vaccines());

Line 56: addButton(menuPanel, "Exit", e -> System.exit(0));

Explanation: The CovidDashboardMain class initializes and listens for user interactions via buttons, where each button click leads to the instantiation of relevant UI components like Cases or Active. This aligns with the Controller pattern as it mediates between user actions and system behavior, handling user input and coordinating the creation and display of other UI components. By abstracting away, the logic for individual views from the main program execution, the CovidDashboardMain class adheres to the Controller responsibilities in GRASP, ensuring a clear separation of concerns and effective delegation of tasks.

**UML Illustrations**

[Class Diagram](https://lucid.app/lucidchart/27904971-2236-4dcd-8389-b4374baf515e/edit?viewport_loc=-1869%2C-745%2C5823%2C2170%2CmBiuhK08BYmJ&invitationId=inv_714acc0f-7526-48ce-b246-4f4483a02f0b)

[Sequence Diagram](https://lucid.app/lucidchart/27904971-2236-4dcd-8389-b4374baf515e/edit?viewport_loc=-1004%2C-888%2C3759%2C1400%2C~Ciuprnn-sJo&invitationId=inv_714acc0f-7526-48ce-b246-4f4483a02f0b)

[Use case Diagram](https://lucid.app/lucidchart/27904971-2236-4dcd-8389-b4374baf515e/edit?viewport_loc=-3051%2C-1047%2C7440%2C2772%2CXDiu5PtI7Ap3&invitationId=inv_714acc0f-7526-48ce-b246-4f4483a02f0b)

**Pattern: High-Cohesion**

* **Purpose:** The High Cohesion pattern encourages each class to have a specific, focused responsibility. This is to mitigate overloading classes with unrelated tasks.
* **In Our Code:** The *VaccineTableDisplay* class (in VaccineTableDisplay.java) is considered highly cohesive because of the way it focuses only on rendering a table that displays vaccine totals by province. It performs this task by creating and configuring the JFrame, setting up the JTable for data, populating the table, and then displaying it. It does not stray into unrelated tasks such as fetching data from a database, complex computations, event handling, etc. Therefore, the class maintains the single responsibility of constructing and displaying this specific window.

**Examples:**

* Lines 9-11 in VaccineTableDisplay.java
* JFrame frame = new JFrame("Vaccine Totals by Province");
* frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);
* frame.setSize(400, 300);
* **In these lines, we can see that the class is responsible for one thing and one thing only, defining the window that will display the table. And we can see that all actions concerning the window are kept together for even better cohesion.**
* Lines 17-19
* for (Map.Entry<String, Integer> entry: provinceVaccineTotals.entrySet()) {
* model.addRow(new Object[]{entry.getKey(), entry.getValue()});
* }
* **This loop iterates through the provinceVaccineTotals map and adds rows to the DefaultTableModel. The class directly uses the provided data to populate the table while keeping the logic for rendering the data separate from its retrieval or computation. The data population logic is part of the core responsibility of rendering the table. It doesn’t concern itself with where the data came from or what to do with it after.**

These are just a few of the lines that show that the entire class is focused solely on rendering a table with the given data. Each line contributes to this single task.

**UML Illustrations**

[Class Diagram](https://lucid.app/lucidchart/6988ddf6-5ef5-45e5-9ecd-42ab9c907845/edit?viewport_loc=-856%2C-364%2C4091%2C1524%2C0_0&invitationId=inv_e0380ef1-836b-4420-8e3f-ab0a70c2be11)

[Sequence Diagram](https://lucid.app/lucidchart/6988ddf6-5ef5-45e5-9ecd-42ab9c907845/edit?viewport_loc=-1618%2C-391%2C4134%2C1540%2CnRiu16zlrnNs&invitationId=inv_e0380ef1-836b-4420-8e3f-ab0a70c2be11)

[Use case Diagram](https://lucid.app/lucidchart/6988ddf6-5ef5-45e5-9ecd-42ab9c907845/edit?viewport_loc=-1193%2C-385%2C3896%2C1451%2CrSiu~MVCd-Jl&invitationId=inv_e0380ef1-836b-4420-8e3f-ab0a70c2be11)