Design Patterns - ChainAbuse

מגישים:

אדם כיאל 211584271

עמנואל דוידוב 209122639

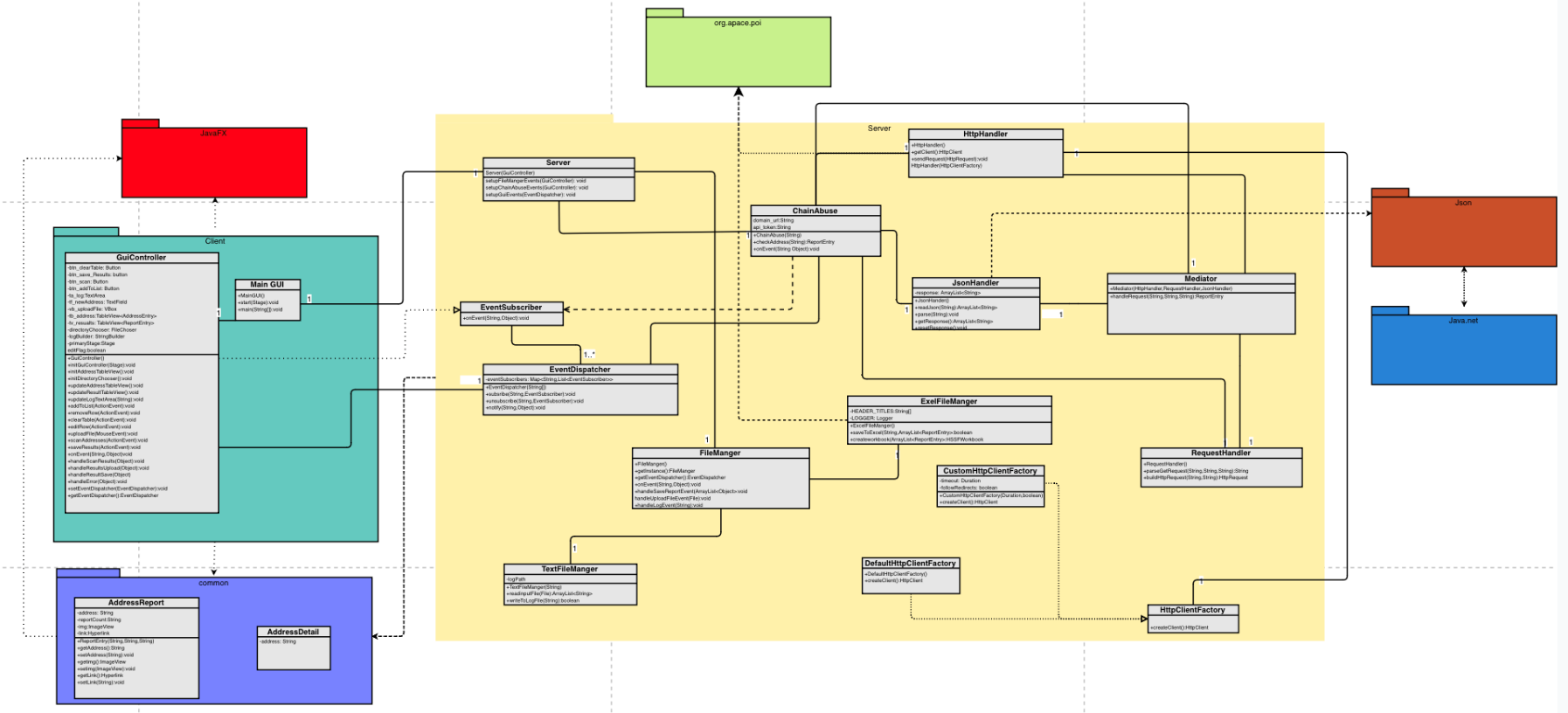
אלכס בבושין 310926415

ליטל לשצ'ינסקי **208658948**

[GIT LINK](https://github.com/emanueldavidov/ChainAbuse/)

**phone : 0533324478**

**Class Diagram**

****

**Design pattern**

**1. Mediator Pattern**

**Usage:** Centralizes communication and interaction between HttpHandler, RequestHandler, and JsonHandler into a Mediator class.

**What we did:**

* Introduced a Mediator class to manage communication between HttpHandler, RequestHandler, and JsonHandler.
* The ChainAbuse class now interacts only with the Mediator, delegating the orchestration of tasks.

**Relevant Classes:**

* **Mediator:** Manages interaction between HttpHandler, RequestHandler, and JsonHandler, providing centralized methods for performing specific operations.
* **HttpHandler, RequestHandler, JsonHandler:** Sub-components coordinated by the Mediator for HTTP communication, request handling, and JSON parsing.

**2. Observer Pattern**

**Usage:** Implements event-driven communication between ChainAbuse and the EventManager.

**What we did:**

* Implemented EventManager to manage events and notify subscribers when they occur.
* ChainAbuse subscribes to events like "Scan Results" and "Error" to perform actions based on notifications.

**Relevant Classes:**

* **EventManager:** Handles event subscription, unsubscription, and notification to listeners.
* **ChainAbuse:** Subscribes to EventManager events and performs specific actions when notified.

**3. Singleton Pattern**

**Usage:** Ensures that only one instance of certain components exists across the application.

**What we did:**

* Implemented the **Singleton Pattern** in FileManager to provide a single, shared instance for managing file operations.
* Optionally, JsonHandler could also be a Singleton to ensure consistent JSON handling.

**Relevant Classes:**

* **FileManager:** Provides a single instance for handling file-related operations like reading logs and saving reports.
* **JsonHandler:** (If Singleton) Manages JSON parsing and serialization tasks with a single instance.

**4. Factory Pattern**

**Usage:**

The **Factory Pattern** is used to **create different types of HttpClient instances** dynamically. This allows flexibility in configuring HTTP clients based on specific needs, such as setting timeouts, enabling redirects, or handling authentication.

**Implementation Details:**

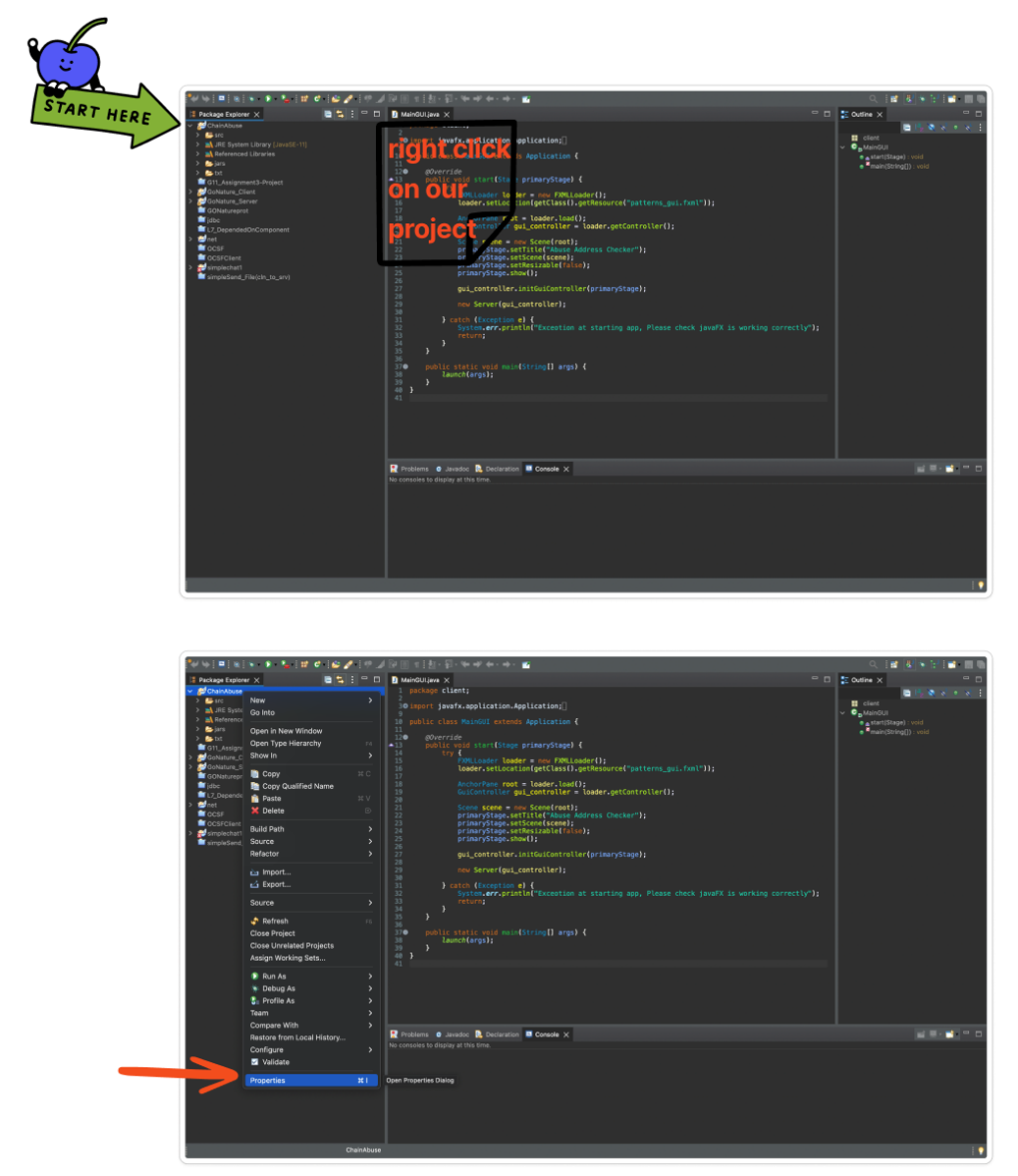
* **HttpClientFactory** defines a common interface for creating HttpClient instances.
* **DefaultHttpClientFactory** provides a simple, lightweight HttpClient without extra configurations.
* **CustomHttpClientFactory** (⚠️ **Not currently in use, but available for future needs**.) Allows an advanced HttpClient with configurable **timeouts and redirect handling**.
* **HttpHandler** uses a factory to obtain an HttpClient, making it independent of the specific client implementation.

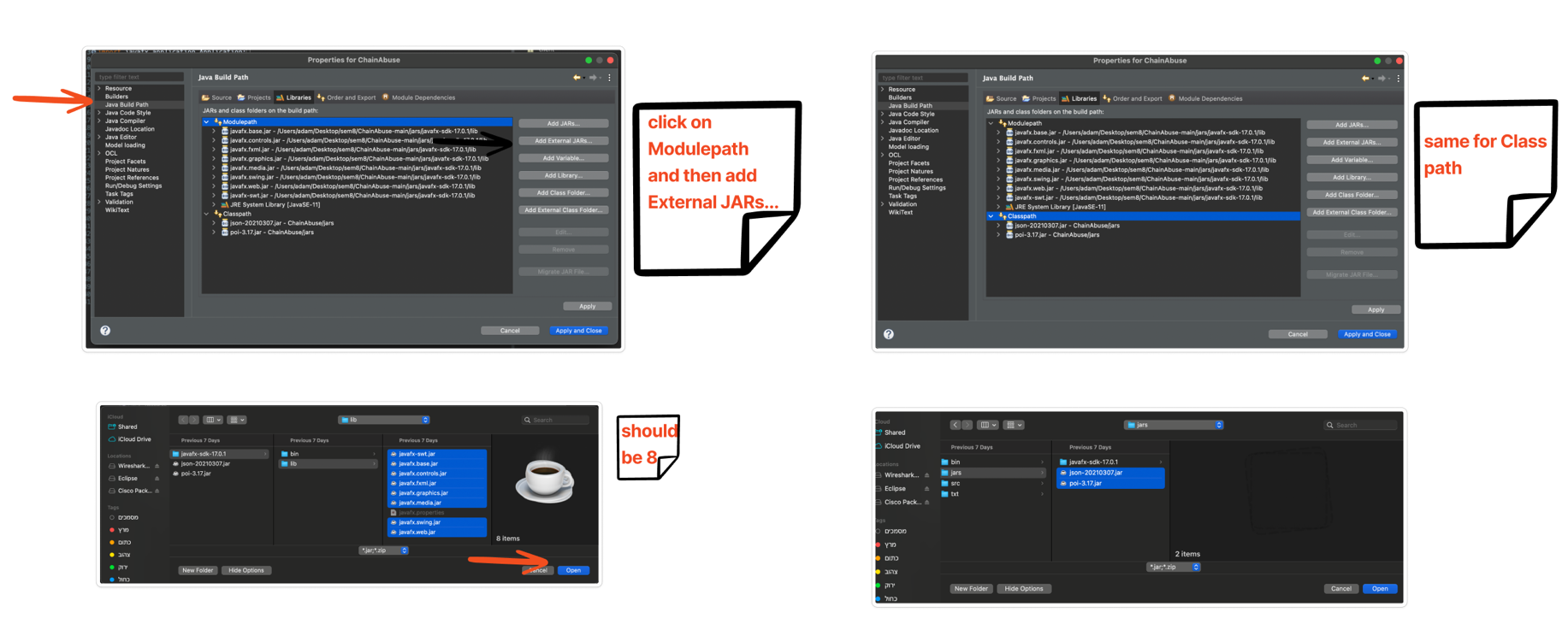
**Relevant Classes:**

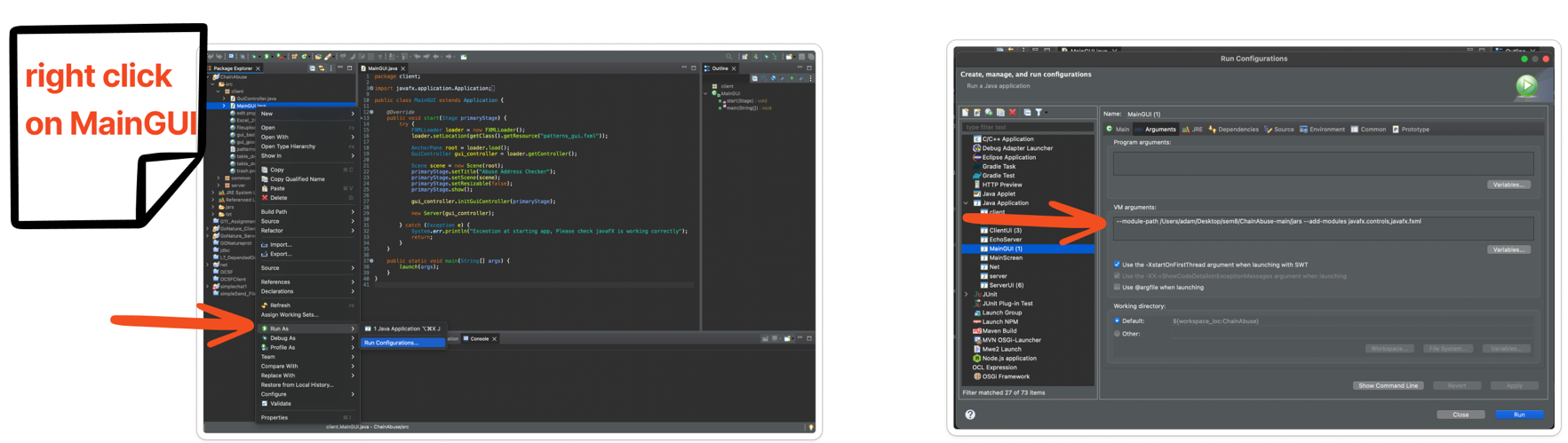
* **HttpClientFactory.java** → Defines the factory interface.
* **DefaultHttpClientFactory.java** → Creates a basic HttpClient.
* **CustomHttpClientFactory.java** → Provides a customizable HttpClient (not in use yet, but kept for scalability).
* **HttpHandler.java** → Uses a factory to create an HttpClient, allowing easy switching between different implementations.

**Configuration Manual**

**1) Add JAR files to build path**

****



**2) Modify the run configuration to include the correct VM arguments with the appropriate path**

**(The path to the jars on your own computer here is an example)**

**--module-path /Users/adam/Desktop/sem8/ChainAbuse-main/jars --add-modules javafx.controls,javafx.fxml**

**Final step**

**Click run:**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

הסבר התקנה בסרטון + הרצות עם\בלי מקרי קצה:

[ChainAbuse Guide](https://www.youtube.com/watch?v=alrE4E7z82s&ab_channel=EmanuelDavid)

הסבר במלל:

**1. Import the Project**

1. Open **Eclipse IDE**.
2. Navigate to File > Open Projects from File System.
3. In the dialog box, select the project folder:
   * Browse to the ChainAbuse project directory.
   * Click **Finish** to import the project into Eclipse.

**2. Configure the Project's Build Path**

1. Right-click on the project ChainAbuse in the Project Explorer and select **Properties**.
2. Navigate to **Java Build Path > Libraries**.
3. Perform the following steps to add the required JAR files:
   * **Modulepath**:
     1. Select the **Modulepath** entry and click **Add External JARs**.
     2. Navigate to ChainAbuse-master/jars/javafx-sdk-17.0.1/lib.
     3. Select all the .jar files in this folder and click **Open** to add them.
   * **Classpath**:
     1. Select the **Classpath** entry and click **Add External JARs**.
     2. Navigate to ChainAbuse-master/jars.
     3. Select the following JAR files and click **Open**:
        + json-20210307.jar
        + poi-3.17.jar
4. Click **Apply and Close** to save the build path changes.

**3. Configure the API Key**

1. **Register and Obtain an API Key**:
   * Visit [ChainAbuse](https://docs.chainabuse.com/reference/reports-1) to register and obtain an API key.
2. **Generate the Basic Authorization Header**:
   * In the **Basic** section of the ChainAbuse documentation:
     + Use your API key as the **username**.
     + Use your API key as the **password**.
   * Copy the **Basic Authorization Header** string generated in the documentation.
3. **Update the api\_json.json File**:
   * Navigate to the file at ChainAbuse-master\txt\api\_json.json.
   * Open the file in a text editor or IDE.
   * Paste the copied **Basic Authorization Header** string into the appropriate field in the JSON file.

**4. Set VM Arguments**

1. In the Project Explorer, navigate to the src > client folder.
2. Right-click on the MainGui class and select **Run As > Run Configurations**.
3. In the **Run Configurations** dialog:
   * Select the MainGui configuration.
   * Navigate to the **Arguments** tab.
   * In the **VM arguments** section, add the following:
   * --module-path "path-to-ChainAbuse-master/jars" --add-modules javafx.controls,javafx.fxml

Replace path-to-ChainAbuse-master/jars with the actual path to the jars directory on your system. For example:

--module-path "C:\Users\emanu\Desktop\ChainAbuse-master\jars" --add-modules javafx.controls,javafx.fxml

1. Press **Apply** to save the configuration.

**5. Run the Application**

1. In the same **Run Configurations** dialog, click **Run**.
2. The application will launch, and the GUI will be displayed.

**Troubleshooting**

* **Missing JAR Files**: Ensure all required JAR files are correctly added to the Modulepath and Classpath.
* **Invalid API Key**: Double-check the Basic Authorization Header in api\_json.json.
* **JavaFX Errors**: Verify the --module-path points to the correct jars directory.

**Design Patterns in Use**

* **Mediator**: Manages communication between components, reducing direct dependencies.
* **Observer**: Implements a subscription mechanism for events and notifications.
* **Singleton**: Ensures that a class has a single instance and provides a global access point.
* **Factory Method**: Abstracts the creation of objects for flexibility and scalability.