DIS2 - Assignment3

Ridho Laksono (340985)
Zeeshan Haider Malik (340899)
Muhammad Rohan Ali Asmat (340900)

To Compile The Window Program

After you are in the root folder you can use the following commands to execute the program:
> javac -cp GES.jar *.java
to run
> java Main

NOTE:
Please extract GES.jar if you find any java.lang.NoClassDefFoundError **exception** at the time of running it after compilation success.
> `jar xf GES.jar`

DISCLAIMER

This app depends on the size of the host OS's size of title bar and its border. The mouse position was not reported correctly by GES. GES reports the mouse position with the height of the host OS's title bar and its window's border being left out.

We hardcoded the height of the title bar and the border size thus when you have different size of title bar and windows border, than the app will not work correctly. This is because of **limitation** of **GES.jar**.

The setting that we are optimizing our app to: Windows 8 with window border size = 0px and title bar = 25px.

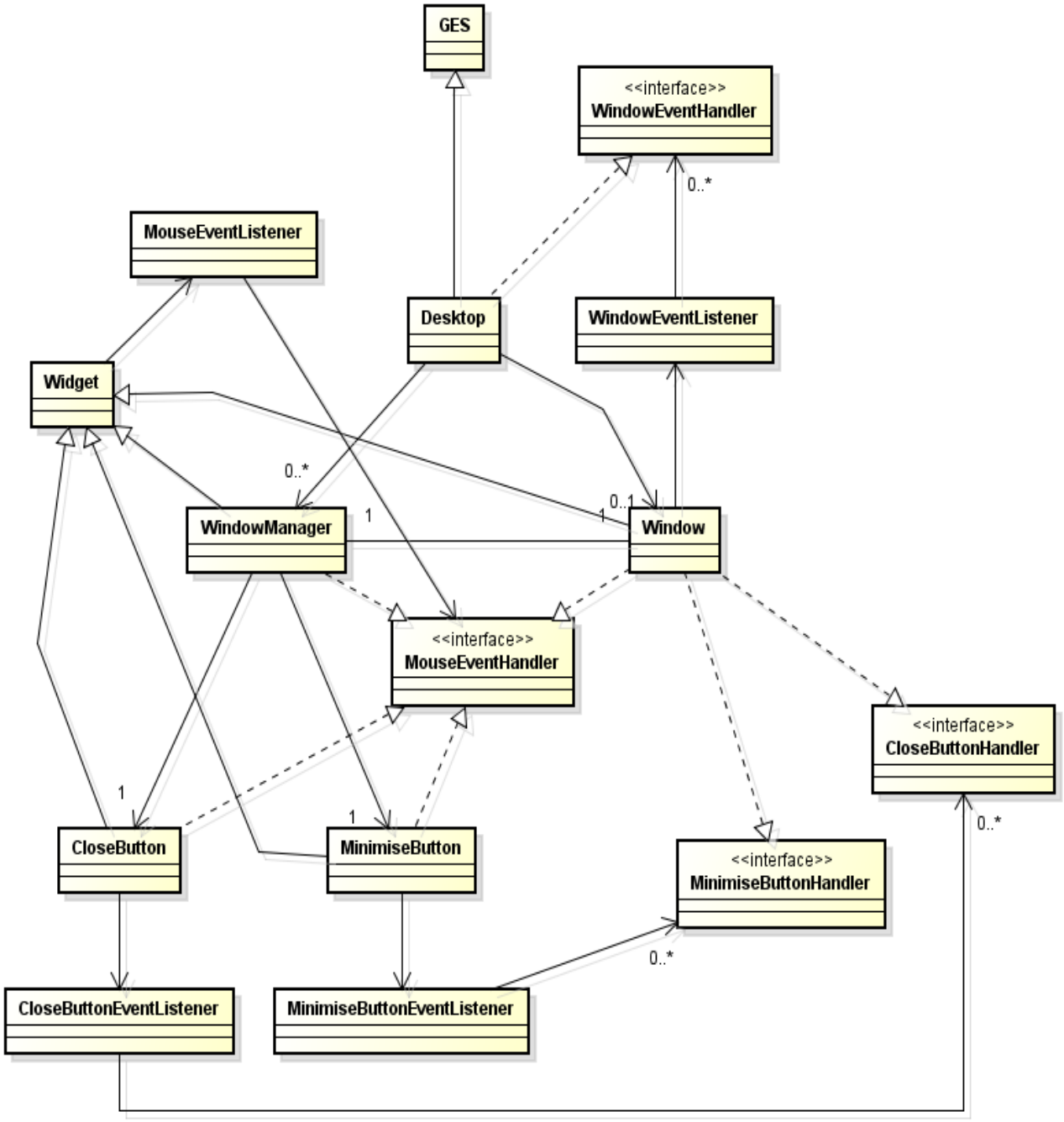
Implementation

It will show 2 windows at start.

- **Drag** and **drop** window movement for visual feedback.
- **Close** and **minimize** button also for visual feedback.

Answer 1 & 2 (Testing your understanding) Design choices

Please see the following class diagram for the complete design of our application:



- **Desktop** class that extends GES only has access to the **root Windows**.

- We wanted to create similar as the X windows system, in which they can instantiate different type of **WindowManager** without having the **Window's** consent.
- So that is why, we create the **WindowManager** and than assign them to the **Window** after it has been created.

- Then we initialize the **WindowManager** with some relevant parameters that belong to its **Window**.

- Obviously to be able to control its assigned **Window**, we have this **WindowManager** to have access to its **Window** that it manages.

- We are passing around r**eference** of **GES** to the **methods** that **draw**, and **color**.

Design pattern that we chose:

Factory Pattern:
Desktop that handles creating and removing windows.

Observer Pattern:
- We have **list of listeners** to some of our classes.

- We **implemented EventHandler Interface** that can be implemented to any of the classes that wants to listen to the Event.

- We then **add object** of those classes that **implements its handler**.

Extra Credit

1. **Border** around window
2. **Minimizing** and **Maximizing** the window:
   - Though not **into an icon** but we could say into a **"mini mode".** This will make it **smaller** and make its **background color into a bit darker**. Click again the minimize button while in mini mode to **maximize** it.