SubscribeNotify Architecture Demo

To execute the project:

1. Navigate in Terminal to the directory `Executable_File`
2. Run this command in a terminal window: `java -jar A1_Mohamed.jar`
3. The program should start in the Terminal Window

Important Notes

- When initially running the project, all Subscribers (*even, odd, prime, composite*) are subscribed to the publisher

- All Subscriber Classes are built on top of the Interface Subscriber

Introduction to Proposed System

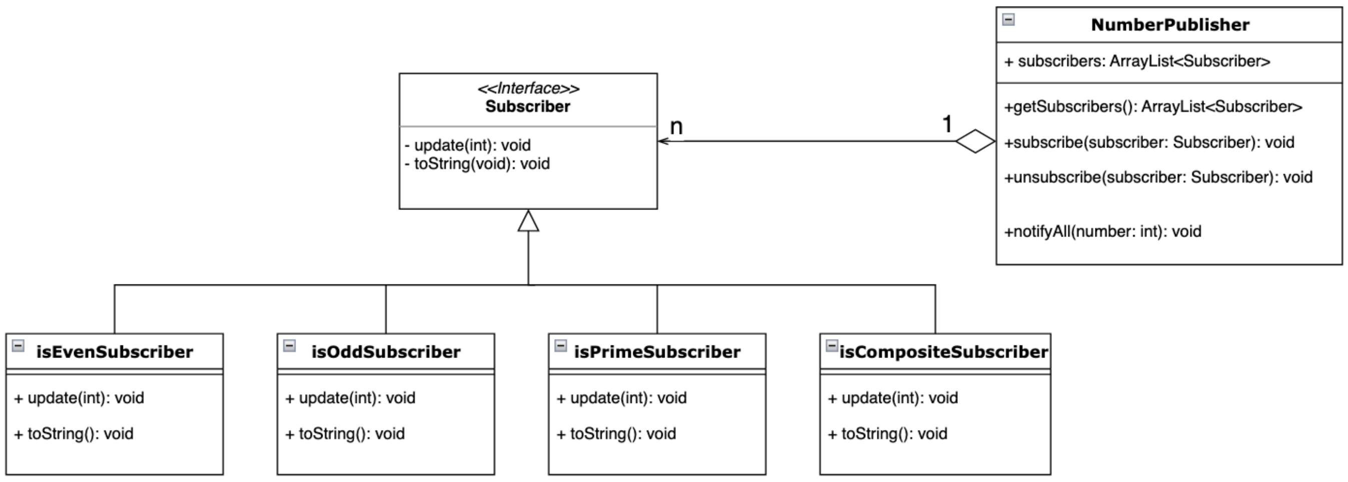
The proposed system is an architecture that utilizes the subscribe-notify pattern to analyze an input number and determine its type. The system can handles types such as even, odd, prime, and composite. Due to the design of the system, more types can be added to the system with ease, which follows the principle of *open for addition closed for modification*.

The Subscribe-Notify pattern, also known as the Observer pattern, or the Publish-Subscribe pattern is when an object (Subject) maintains a list of dependents, called Observers. The Subject notifies its Observers automatically of any state changes in the system. In the context of this project, the Main Module initiates by sending a Subscribe message expressing the interest to recieve notifications from said subscribed observers that specialize in determining the type of a number. The Subscribers wait for the Subject to invoke them and then notify the Subject of what the type of number being analyzed is. Once the type of the number is decided, the main module sends a stop request, which is described in this design as an unsubscribe message.

The advantages of the subscribe-notify design pattern is the ability of the design to be modulat and extensible. The Publisher can subscribe to many different Subscribers, and can also add or remove them the system without affecting the functionaliyu of the main module. The pattern also promotes loose coupling between the components of the system, which allows the system have have high maintainability.

Overall, the Subscribe-Notify pattern allows for the design to be flexible, mainainable, and extenisble. The pattern also promotes loose-coupling, and follows the principle of *open for addition closed for modification*.

Class Diagram



References

Microsoft Observer Design Pattern Learn

The Observer Design Pattern