CSCI-4448/5448: Final Report

1. **Team:** Woncheol Song
   Tae Gu Kim
   Hui Soon Kim

2. **Title:** Packet Analyzer

3. **Description:** An application program to analyze network protocols, visualize packet information and network statistics from packet data in order to assist network administrator

4. Implementation Questions

   4.1. What features were implemented?

   The following table shows the features implemented in this project. 13 out of 14 features are implemented except US-05. (Most of the functional and non-functional features are also implemented)

ID	Requirement	Topic Area	User	Implemented?
US-01	As a user, I should be able to start capturing network packet so that I can see the contents of network packet	Capture	All	YES
US-02	As a user, I should be able to stop capturing network packet so that I don't waste storage to save uninterested data	Capture	All	YES
US-03	As a user, I want to select network device to be used in capture so that I can collect network data from interested interface	Capture	All	YES
US-04	As a user, I want to analyze a specific packet in the main screen so that I can see the contents of the packet in detail	Display	All	YES
US-05	As a user, I want to select a specific layer from the selected packet so that I can check the range of content corresponding to selected layer	Display	All	***NO***

US-06	As a user, I want to filter packets so that I can see only interesting packets.	Filter	All	YES
US-07	As a user, I want to choose the range of time period in the filter screen so that I can see packets only for interested period of time	Filter	All	YES
US-08	As a user, I want to choose the type of protocol in the filter screen so that I can see packets only for interested type of packets.	Filter	All	YES
US-09	As a user, I want to see the statistics of network traffic as a visual graph so that I can the network status intuitively	Statistics	All	YES
US-10	As a user, I want to see packet traffic per second in the statistics screen so that I can see the network traffic rate	Statistics	All	YES
US-11	As a user, I want to see traffic for each protocol in the statistics screen so that I can see the protocol variation	Statistics	All	YES
US-12	As a user, I want to terminate the program whenever I don't need any use of it	Display	All	YES
US-13	As a user, I want to search specific packet so that I can see packets which contain specific string	Display	All	YES
US-14	As a user, I want to save the incoming packets so that I can use the packets later	DB	All	YES

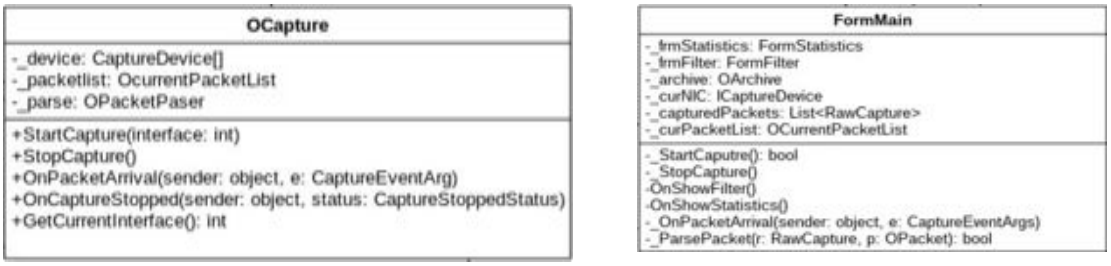
4.2. Which features were not implemented from Part 2?

The feature(US-05) to show a content of a specific protocol layer when user clicks a packet was not implemented. In the first place, we thought that Pacp library may support a detailed parsing function of network packet, but it does not. Considering the given time constraint and the purpose of this class, we thought it is not important to code all parse functions for all network protocols.

4.3. Show your Part 2 class diagram and your final class diagram. What changed? Why? If it did not change much, then discuss how doing the design up front helped in the development.

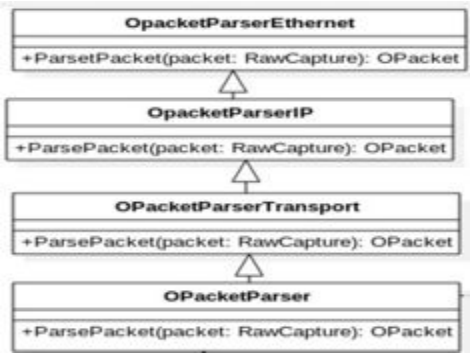
As you can see in the class diagrams in next page, there are two differences between Part 2 and the final class diagram.

First, we implemented the main functions of OCapture class in FormMain class and eliminated the class like following Fig 1. The functions of OCapture class is to start, stop capturing network packets through network interface and process a packet when it arrives but we thought that it is more efficient and simple to move that functions to FormMain class instead of creating new class OCapture even though this could violate Single Responsibility Principle.



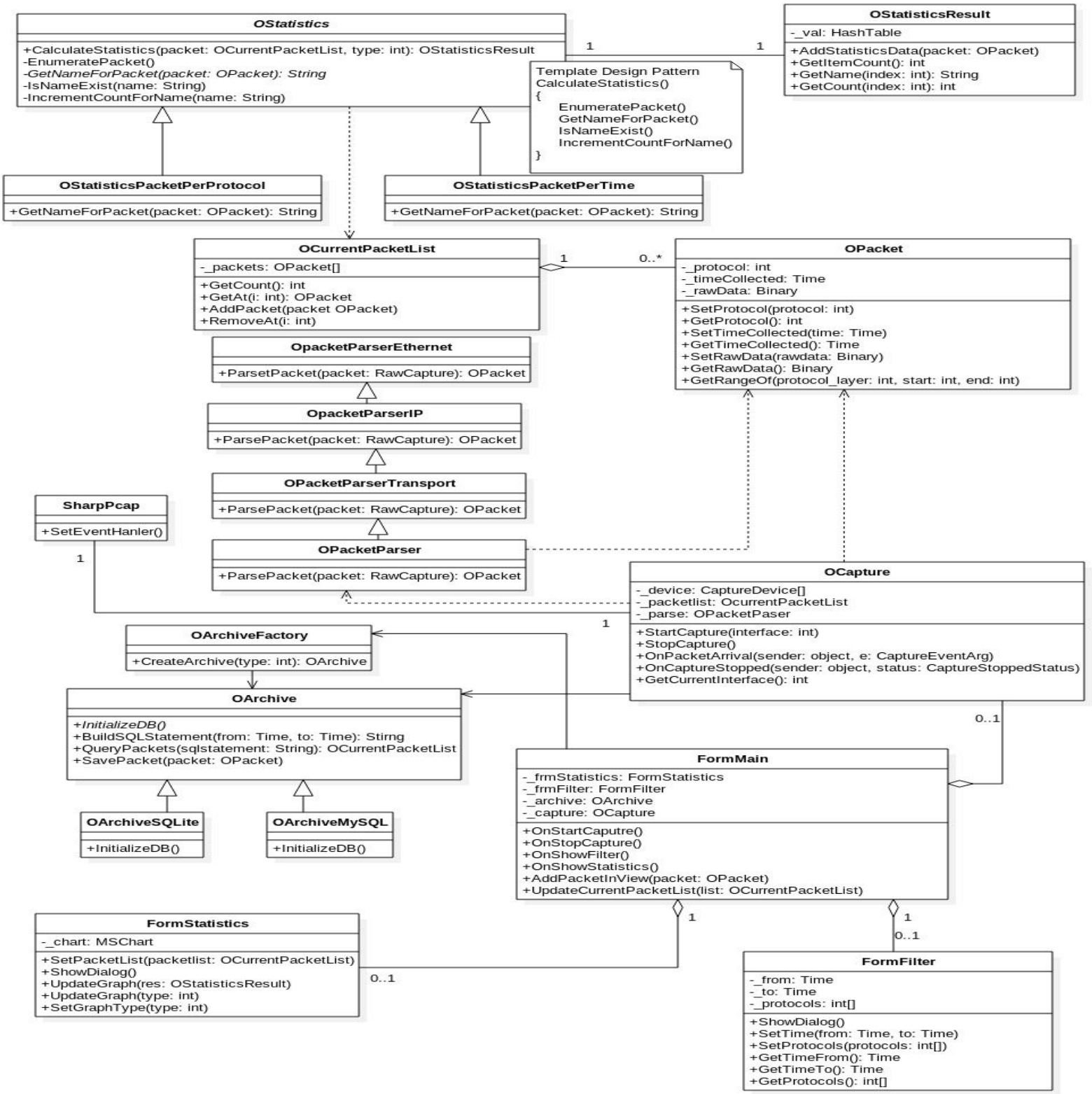
[Fig 1. OCapture functions migrated to FormMain]

Second, we did not implement the classes to parse network packets of the following Fig 2. Instead, we adopted basic network parse functions that Pcap driver supports because the implementing the detailed parser is too time-consuming and is not necessary to complete main purpose of this project.

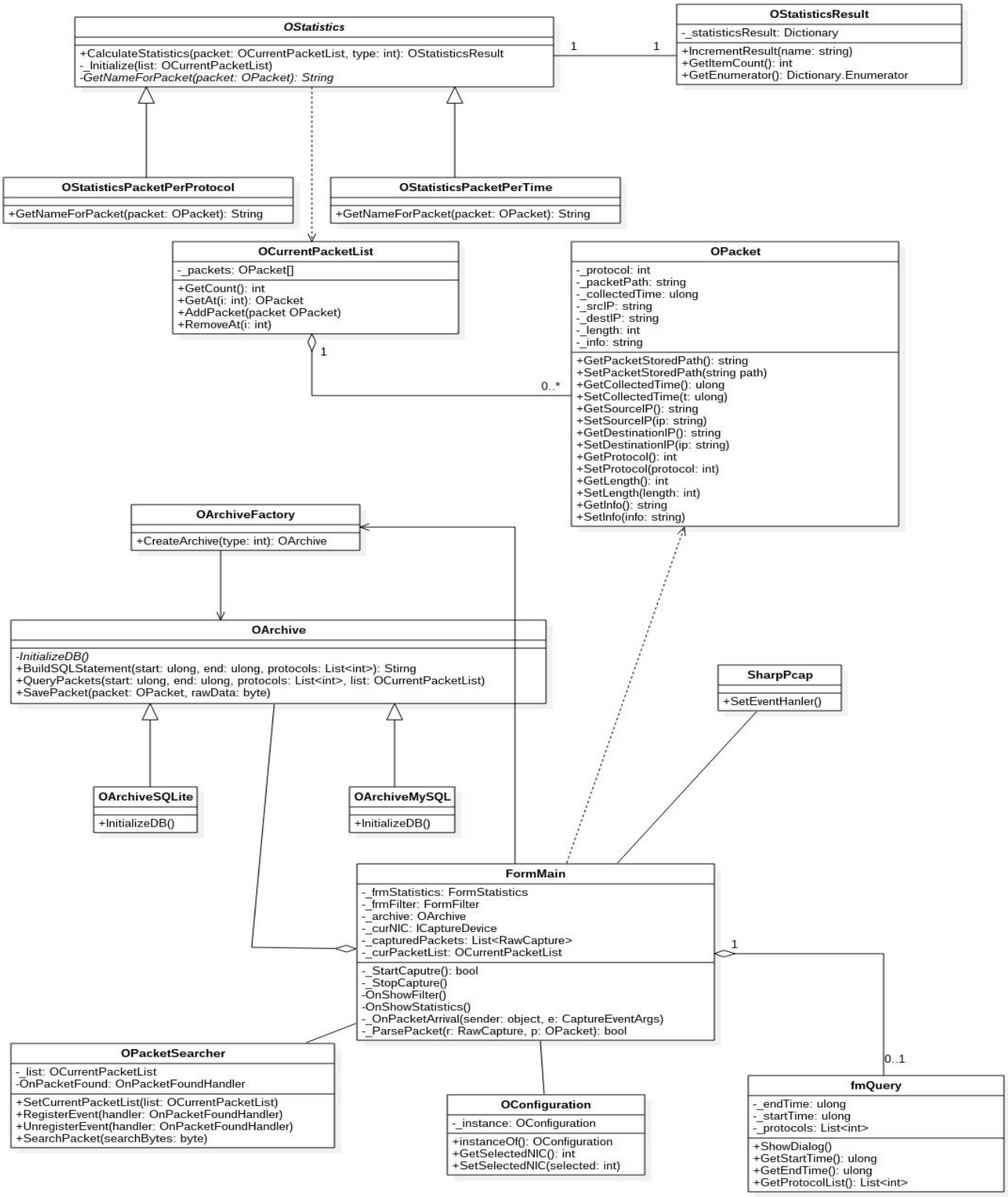


[Fig 2. Classes to parse packets]

[Part2 Class Diagram]

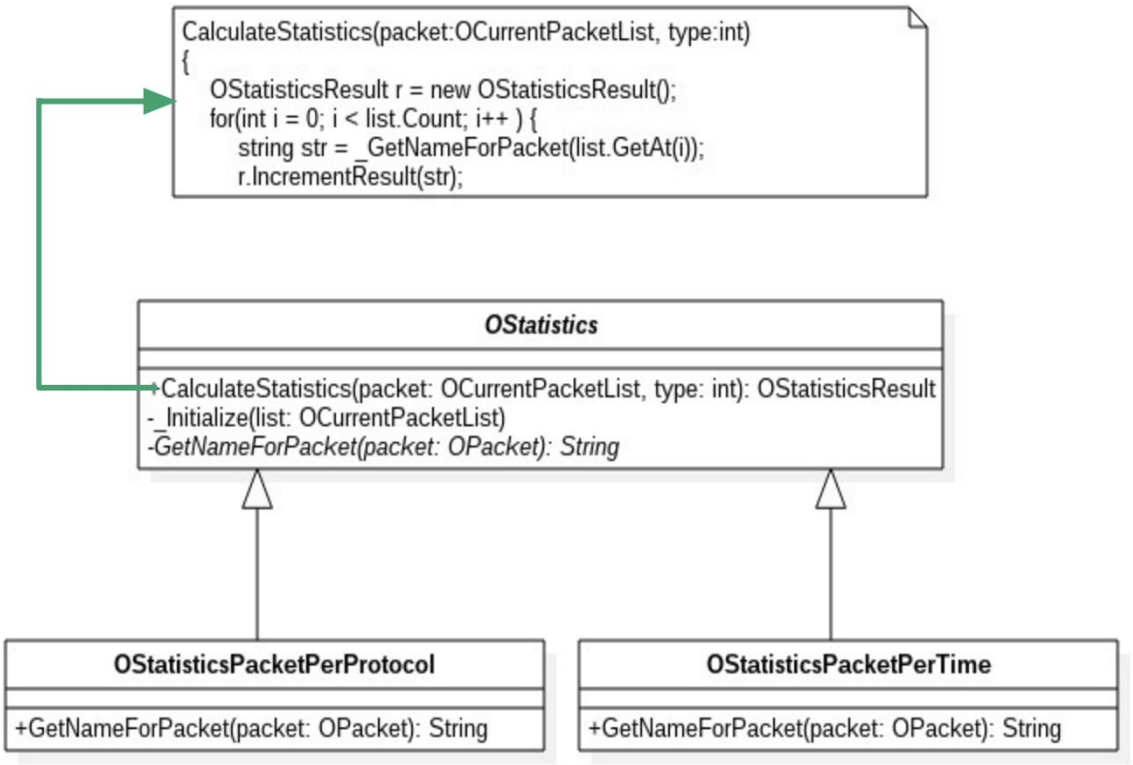


[Final Class Diagram]

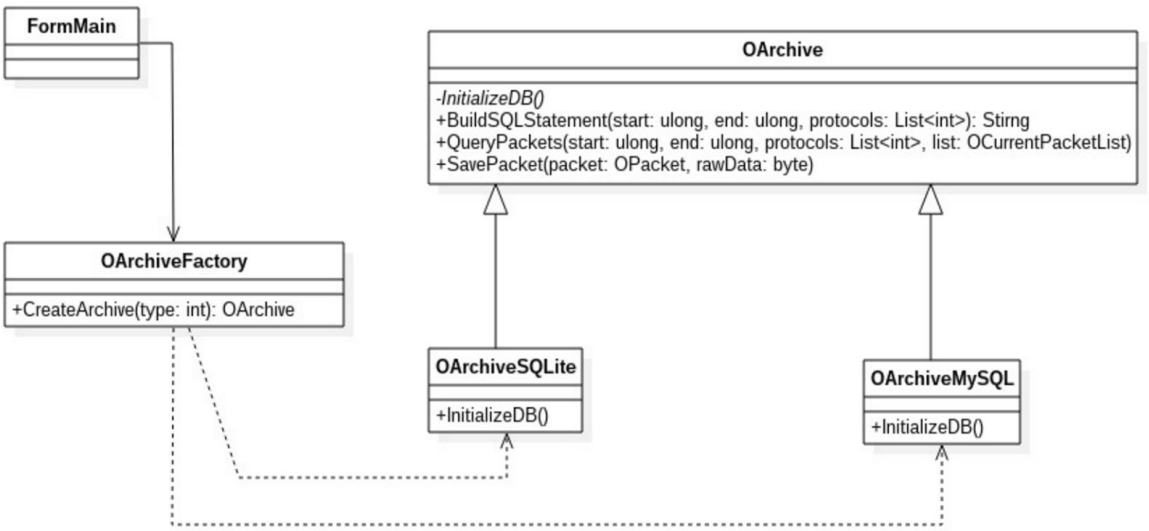


4.4. Did you make use of any design patterns in the implementation of your final prototype? If so, how? If not, where could you make use of design patterns in your system?

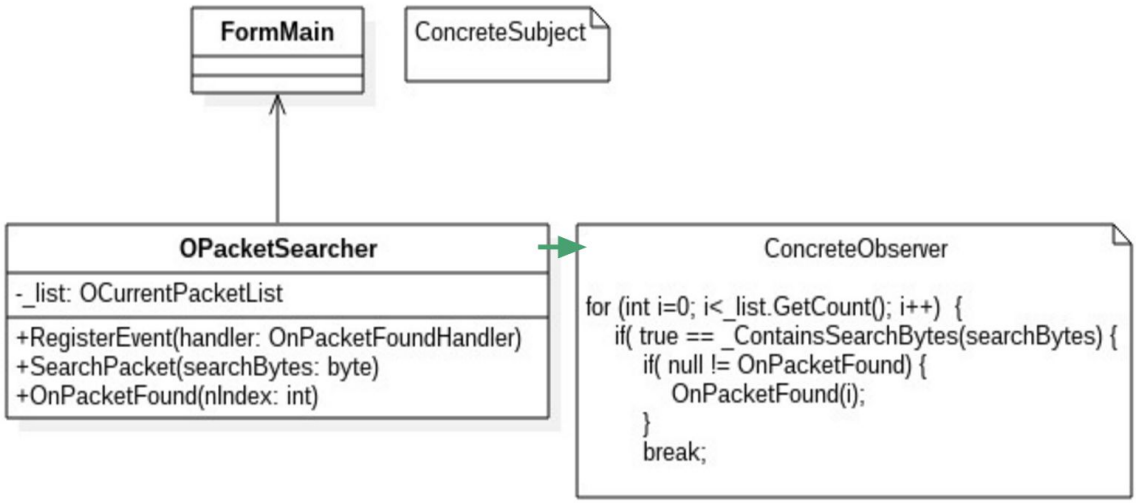
4.4.1. Template



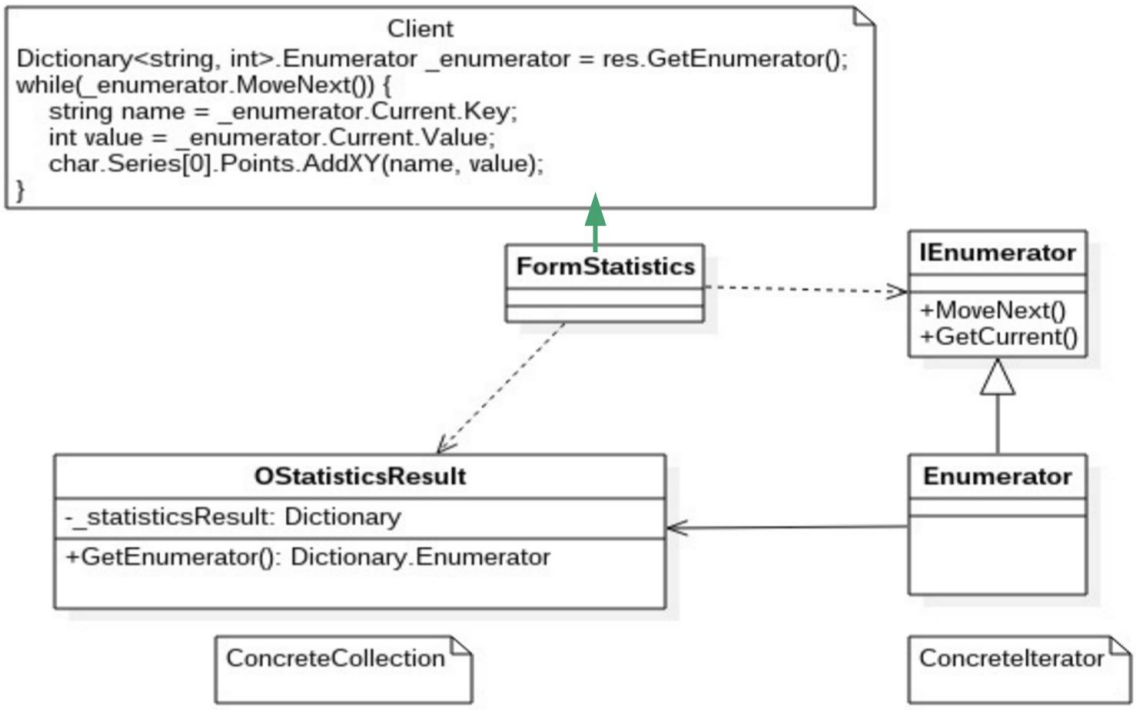
4.4.2. Simple Factory



4.4.3. Observer



4.4.4. Iterator



4.4.5. Singleton

```
public class OConfiguration
{
    private static OConfiguration _instance = null;
    private OConfiguration() { }
    public static OConfiguration instanceOf()
    {
        if (null == _instance)
        {
            _instance = new OConfiguration();
        }
        return _instance;
    }
}
```

4.5. What have you learned about the process of analysis and design now that you have stepped through the process to create, design and implement a system?

1) A thorough and clear documentation of software design is very useful when actually implementing a program because we can just follow the documentation.
2) Applying design patterns on software development project helps exploit expertise of professional programmers and improve communication among team members.
3) When some tests fails or features stop working, changing one thing at a time makes it much easier to find out problems. In other words, it is better to do one thing at a time and make sure it works and repeat.
4) OOP helps team members separate the complex tasks of software development easily. Team member don't need to understand other team member's work thoroughly because OOP adopts abstraction.
5) As the software development project goes, changes in the original design is inevitable so we need to prepare them in advance.