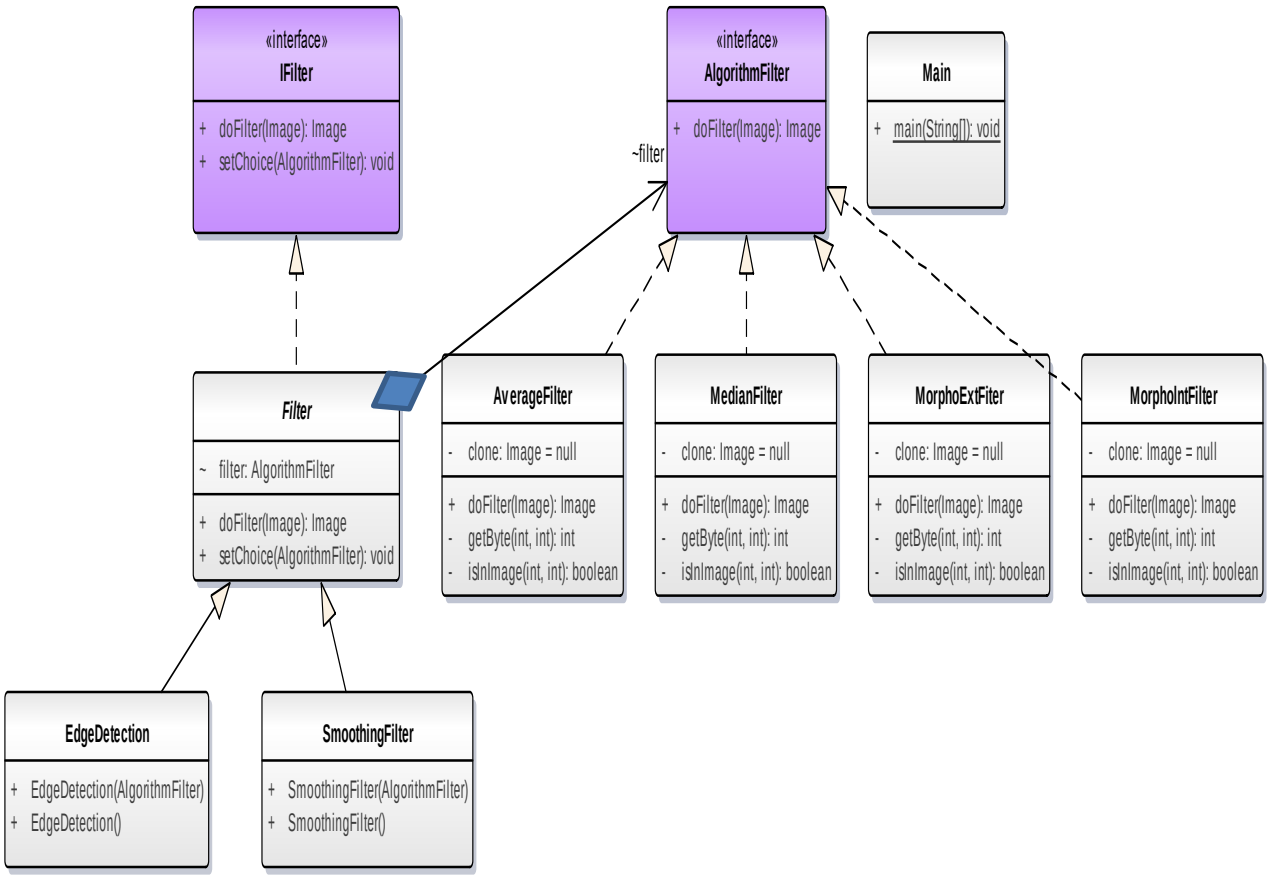
GIT Department of Computer Engineering
CSE443 - Object Oriented Analysis and Design
Fall 2016
HW1

Onur SEZER
121044074

Q1) Each design solution may not be design pattern. A design problem of the simplest and most effective way to resolve the method design pattern is called, design solutions to be design pattern, we need to use **Creatinal Patterns, Behavioral Patterns, Structural Patterns**.

Q2)

Uml Diagram:



The Strategy Pattern defnes a family of algorithms, encapsulates each one, and makes them interchangeable.Strategy lets the algorithm vary independently from clients that use it.MedianFilter, AverageFilter, MorphoExtFiter and MorpholntFilter classes was implemented AlgorithmFilter.Then, implemented IFilter interface which Filter class used AlgorithmFilter interface with composition.Next, SmoothingFilter and EdgeDetection was extended from abstract Filter class.
Strategy pattern gives us the following:

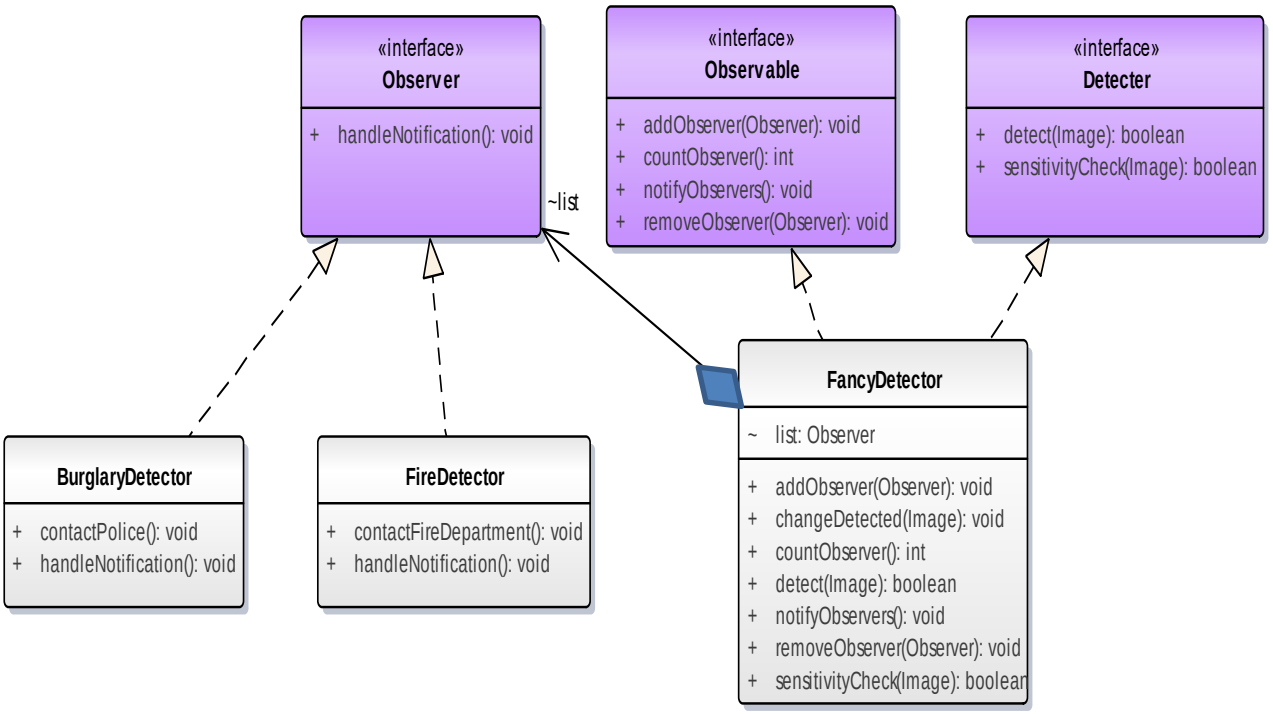
- The algorithm provides a very comfortable way to use writing everywhere at once.
- It was provided the ability to adapt to changes made design.
- We don't need redundant interface and derive class.

Q3)

- Two classes were created to make smoothing and edge from the given image.
- For smoothing and edge in the generated class the doFilter() method takes as a parameter and will return a handle formal customer wants.
- AverageFilter and MedianFilter classes were created for smoothing.And, MorphoExtFiter and MorpholntFilter classes were created for edge detection. As the user wants, my application can be able to switch dynamically between algorithms for every filter. The ability of Java's polymorphism was used for this.

Q4) My codes is in the Codes_OnurSEZER folder.

Q5)


«interface»
Observer
+ handleNotification(): void
«interface»
Observable
+ addObserver(Observer): void
+ countObserver(): int
+ notifyObservers(): void
+ removeObserver(Observer): void
«interface»
Detecter
+ detect(Image): boolean
+ sensitivityCheck(Image): boolean
~list
BurglaryDetector
+ contactPolice(): void
+ handleNotification(): void
FireDetector
+ contactFireDepartment(): void
+ handleNotification(): void
FancyDetector
~ list: Observer
+ addObserver(Observer): void
+ changeDetected(Image): void
+ countObserver(): int
+ detect(Image): boolean
+ notifyObservers(): void
+ removeObserver(Observer): void
+ sensitivityCheck(Image): boolean