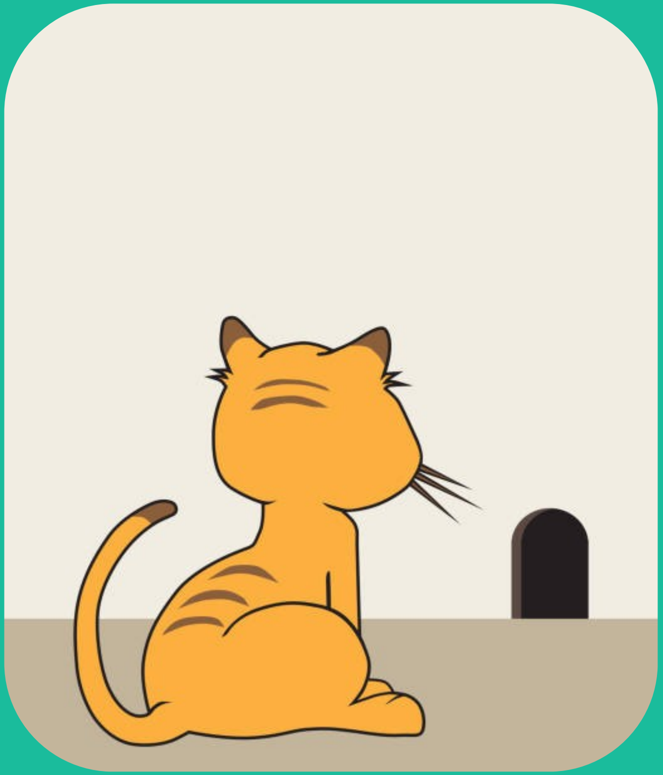
The Observer design pattern



Observer is a behavioral design pattern
that allows one objects to notify other objects about changes in their state.

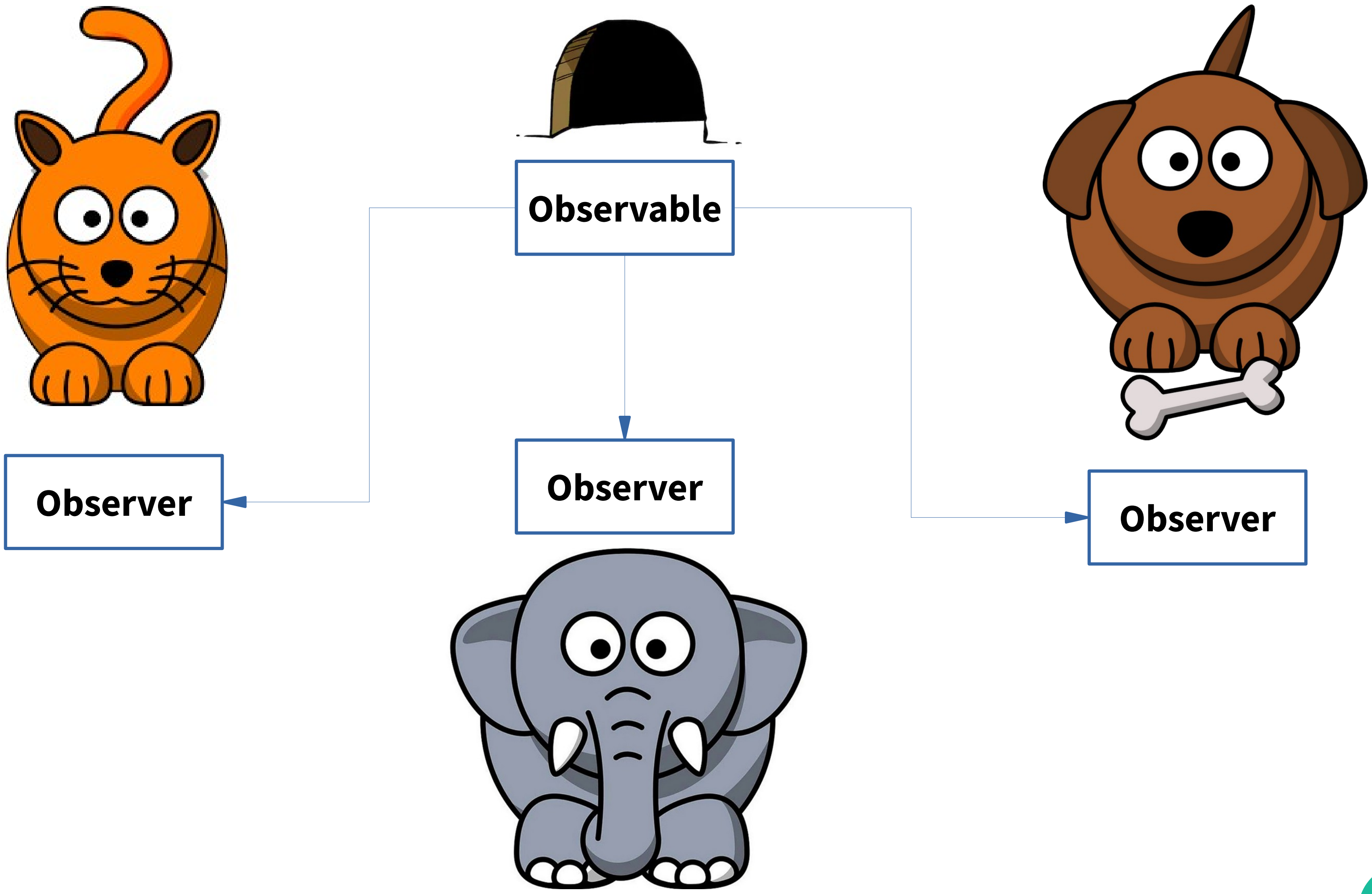
Problem

There is a mouse in the house.

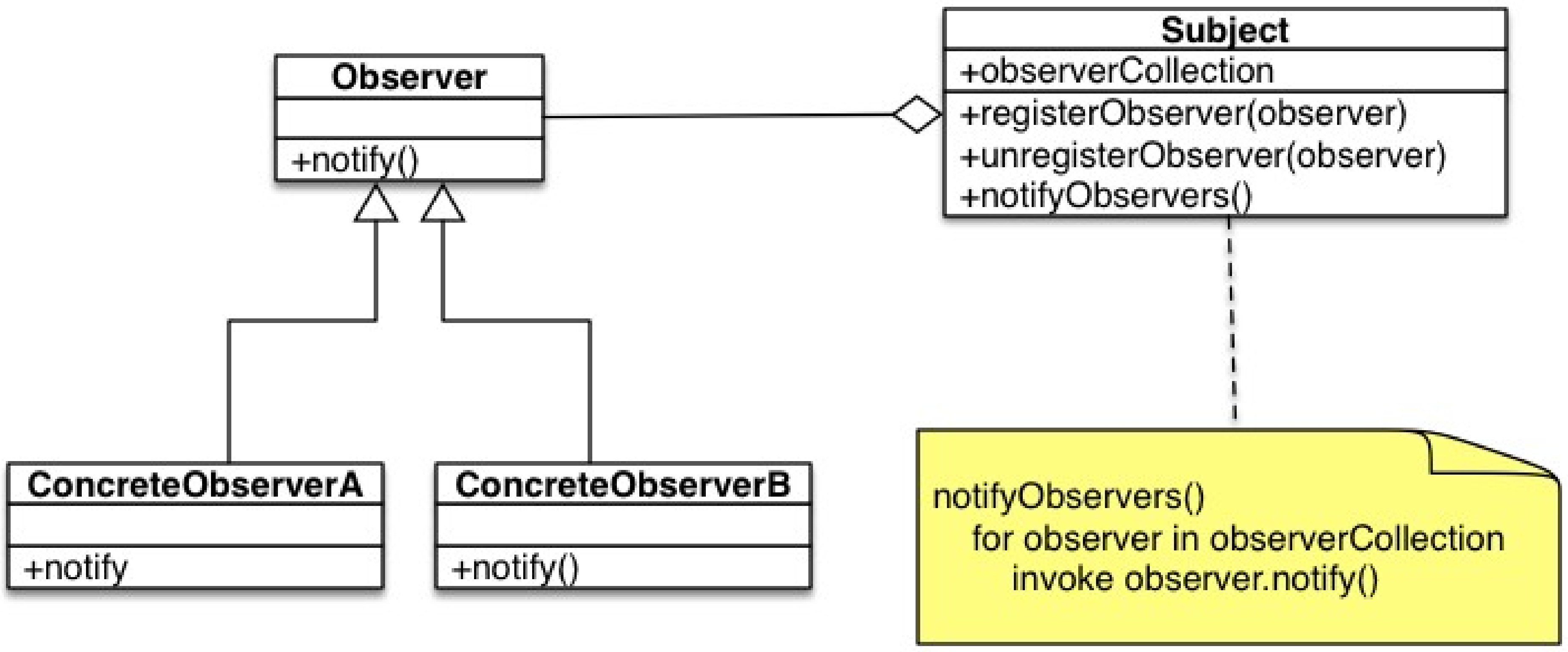
- Cat wants to catch it.
- Elephant wants to avoid it.
- Dog don’t give a f#ck.

So how to do it without all of them staring into the wall???

Solution – the Observer pattern



UML diagram


Observer
+notify()
Subject
+observerCollection
+registerObserver(observer)
+unregisterObserver(observer)
+notifyObservers()
ConcreteObserverA
+notify
ConcreteObserverB
+notify()
notifyObservers()
for observer in observerCollection
invoke observer.notify()

Code – Observable part

```
public interface IObservable
{
    void AddObserver(IObserver observer);
    void RemoveObserver(IObserver observer);
    void NotifyObservers();
}
```

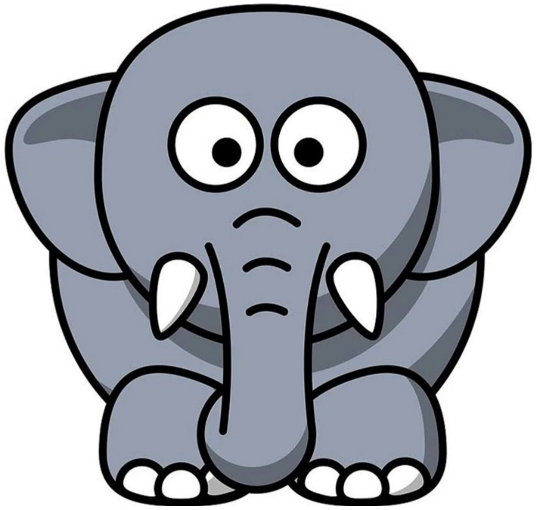
```
public class Observable : IObservable
        {
            private List<IObserver> _observers = new List<IObserver>();
            public int mouseState { get; set; }

            public void AddObserver(IObserver observer)
            {
                _observers.Add(observer);
            }
            public void RemoveObserver(IObserver observer)
            {
                _observers.Remove(observer);
            }
            public void NotifyObservers()
            {
                foreach (var observer in _observers)
                {
                    observer.Update(this);
                }
            }
            public void ShowRandomMouse()
            {
                int mouseNewState = new Random().Next(2);
                Console.WriteLine("===========================================================");
                if (mouseNewState == mouseState)
                {
                    Console.WriteLine("Nothing changed - no need to bother observers...");
                } else
                {
                    mouseState = mouseNewState;
                    if (mouseState == 1)
                    {
                        Console.WriteLine("Listen everyone! Mouse apears in the den!");
                    } else
                    {
                        Console.WriteLine("Listen everyone! There is NO mouse in the den!");
                    }
                    NotifyObservers();
                }
            }
        }
```

Code - Observers part

```
public interface IObserver
{
    void Update(Observable observable);
}
```

```
public class CatObserver : IObserver
{
    public void Update(Observable observable)
    {
        if (observable.mouseState == 1)
        {
            Console.WriteLine("Cat starts to chase the mouse.");
        } else
        {
            Console.WriteLine("Cat leaves the room.");
         }
    }
}
```



```
public class ElephantObserver : IObserver
{
    public void Update(Observable observable)
    {
        if (observable.mouseState == 1)
        {
            Console.WriteLine("Elephant storms out the room terrified.");
        }
        else
        {
            Console.WriteLine("Elephant returns into the room.");
        }
    }
}
```

```
public class DogObserver : IObserver
{
    public void Update(Observable observable)
    {
        Console.WriteLine("Dog don't give a f#ck.");
    }
}
```

Code – main and results

```
static void Main(string[] args)
{
    Observable den = new Observable();

    den.AddObserver(new CatObserver());
    var dog = new DogObserver();
    den.AddObserver(dog);
    den.AddObserver(new ElephantObserver());

    for (var i = 0; i < 10; i++)
    {
        den.ShowRandomMouse();
        if (i == 5)
        {
            den.RemoveObserver(dog);
        }
        Thread.Sleep(5000);
    }

    Console.ReadKey();
}
```

```
Listen everyone! There is NO mouse in the den!
Cat leaves the room.
Dog don't give a f#ck.
Elephant returns into the room.
=====================================================
Nothing changed - no need to bother observers...
=====================================================
Listen everyone! Mouse apears in the den!
Cat starts to chase the mouse.
Elephant storms out the room terrified.
=====================================================
Nothing changed - no need to bother observers...
=====================================================
Nothing changed - no need to bother observers...
=====================================================
Listen everyone! There is NO mouse in the den!
Cat leaves the room.
Elephant returns into the room
```

Thanks everyone!

Code aviable at:

https://github.com/SeniorSSS/DesignPatterns/blob/master/Observer/

Author: Janis Strazdins