

וְאֵת קָדְשֶׁךָ

ohad.klein@mail.huji.ac.il

# תבניות עצוב

## לתלמיד דום העצב

אלן אונן / נורמן:  


סינסינו:  


טמיון:  


# אין טו איזיון

המפעלים יוצרים מוצרים ו שירותים המבוקשים על ידי לקוחות. המפעלים מושפעים מהתנאים הפוליטיים והכלכליים של מדינתם. המפעלים מושפעים מהתנאים הפוליטיים והכלכליים של מדינתם.

ה**戰略** (Strategy) מגדיר את ה<sup>ה</sup>גדרת ה<sup>ה</sup>מטרות וה<sup>ה</sup>הנחיות ש

```
ה
```

כ) הילדה מושגית הערתנו מושגית והילדה מושגית.

בגדרה זו ניתן לראות כי במודול `Facade` מוגדרת שכבת אבטחה על כל הפעולות המאפשרות גישה ל-

**Observer**: רני לא יתגלה ויכיר בזיהויו לאזימט פלגי  
הנתקה בפונקציית הנטגן ופונקציית הסדר, וכך אוצרת?

בכל נציגו שפנוי מפזורה נסעה ברכבת. מטרת הרכבת היא לסייע לאנשים ממקומות מרוחקים להגיע למקום העבודה שלהם. מטרת הרכבת היא לסייע לאנשים ממקומות מרוחקים להגיע למקום העבודה שלהם. מטרת הרכבת היא לסייע לאנשים ממקומות מרוחקים להגיע למקום העבודה שלהם.

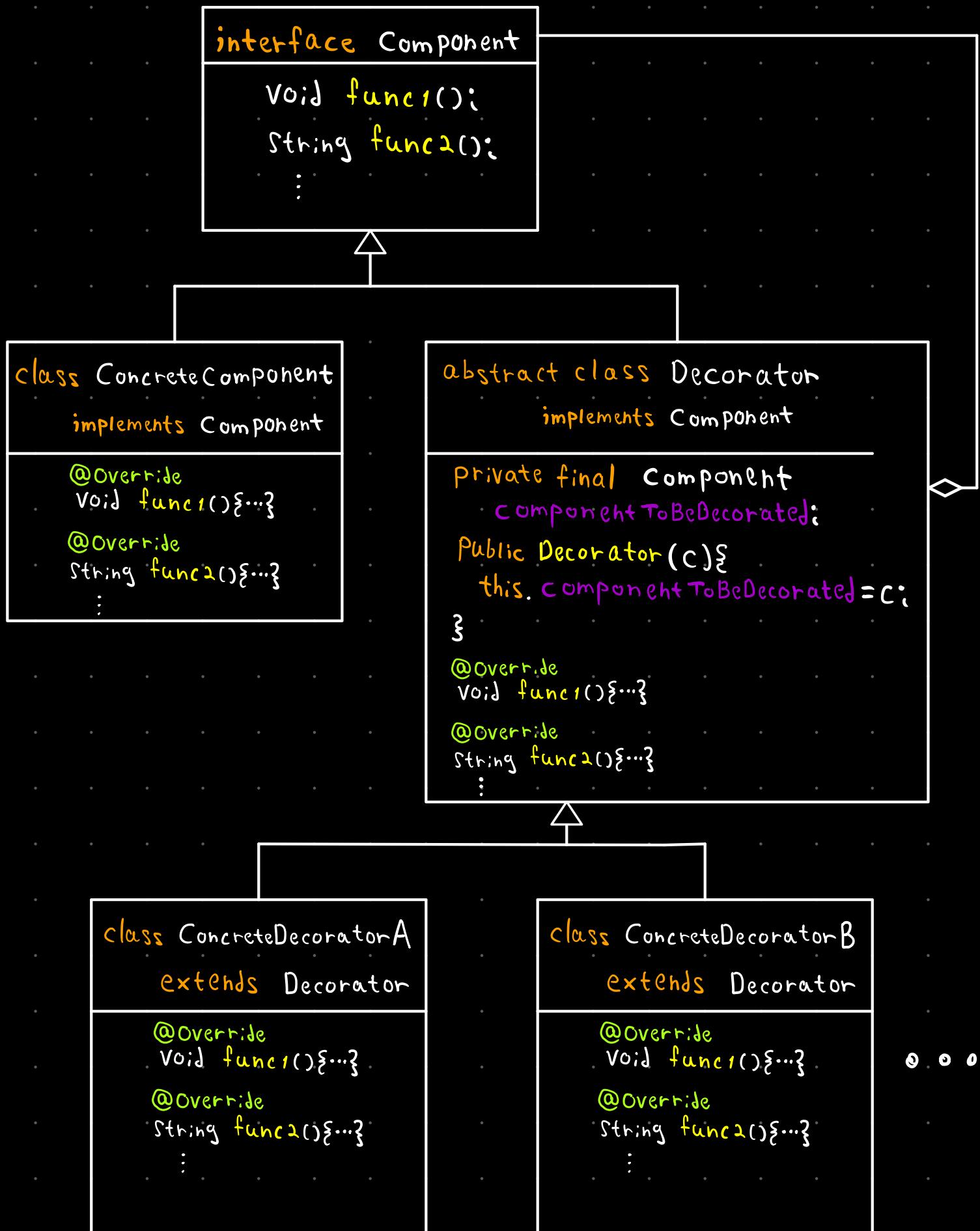
בכל סיבובו של State ישנו מצב אחד ? אשר מוגדר כ'אנו וריאנט' (variant).

המודולsingleton:Logger - (אחד מודול אחד, כמו use)

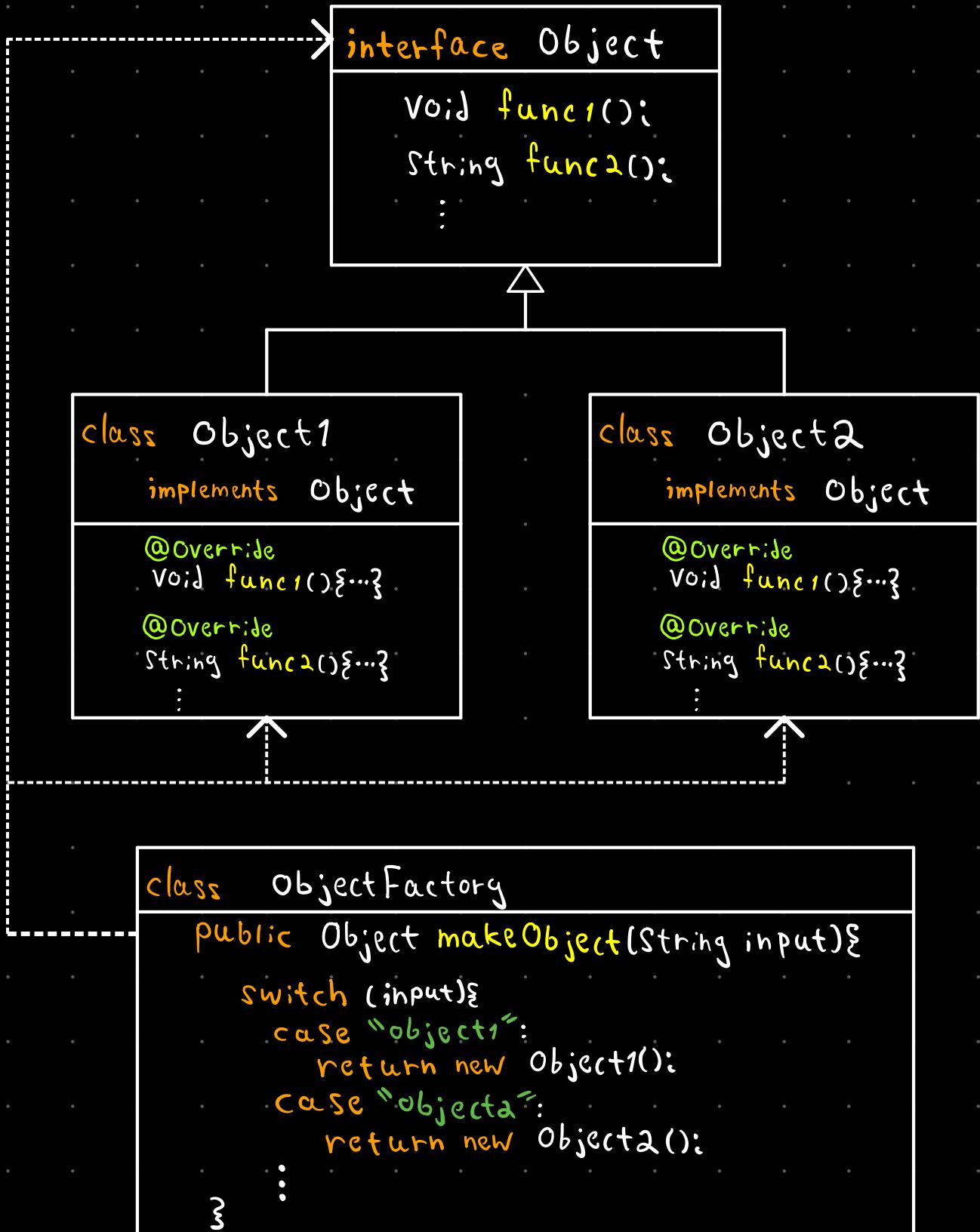
15) נעלמה סדרין מהויר מה נוכח וחיזר מכך הנטיגר  
בכחוון צוין מהויר ביצה מילוי. וק כו רכזת צויר נעלם  
יחסון מהויר יזגדה נוכחה צויר ברכזת צויר.

 מ'ג Memento  
'ג'ז'יקס ליס נוּן פָּרָה מִלְּבָד תְּמִימִינָה אֲנִי מִמְּלֵא  
. פָּרָה פָּונְקָנוּ מִלְּבָד caretaker סִי, Memento

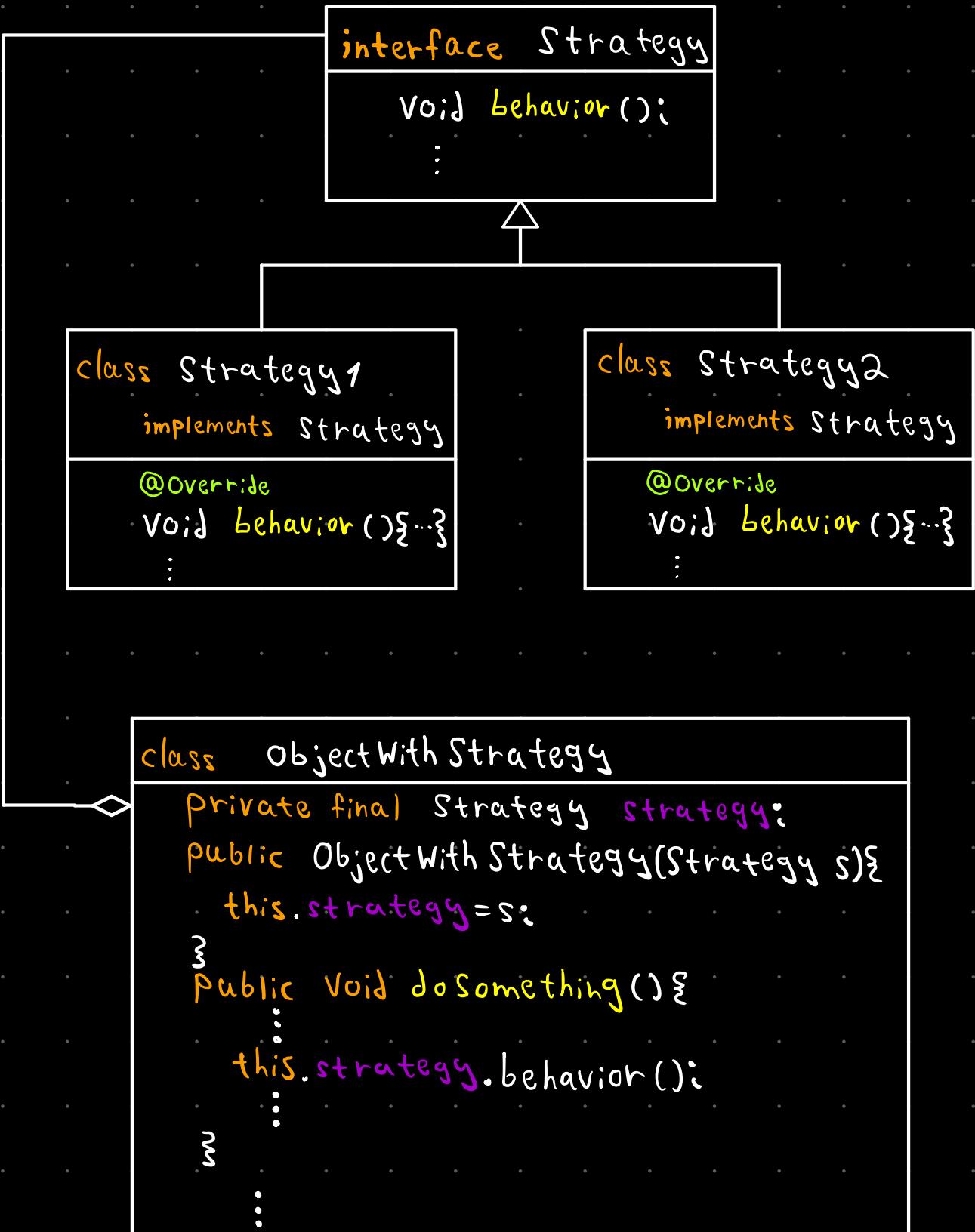
# Decorator



# Factory



# Strategy



# Facade

Complex Structure

class SubObject1

void doSomething() { ... }

:

class SubObject2

void doSomething() { ... }

:

...

class ObjectFacade

private SubObject1 s1;

private SubObject1 s2;

:

void dosomething()

this.s1.doSomething();

this.s2.doSomething();

//more complex operations

//the user doesn't care about

}

:

# Iterator

```
interface Iterator<E>
    E next();
    boolean hasNext();
    :
```

```
interface Aggregate<E>
    Iterator<E> createIterator();
    boolean add(E element);
    :
    :
```

```
class MyAggregate<E>
    implements Aggregate<E>
```

```
@Override
public boolean add(E element){ ... }
```

```
@Override
public Iterator<E> createIterator(){
    return new MyIterator<E>();
```

```
class MyIterator<E>
    implements Iterator<E>
```

```
public E next(){ ... }
```

```
public boolean hasNext(){ ... }
```

```
:
:
```

```
:
:
```

# Observer

interface Subject

```
Void attach(Observer o);  
Void detach(Observer o);  
Void notifyObservers();
```

interface Observer

```
Void update();
```

class ConcreteObserverA  
implements Observer

```
Private ConcreteSubject s;  
@Override  
Public Void update(){...}
```

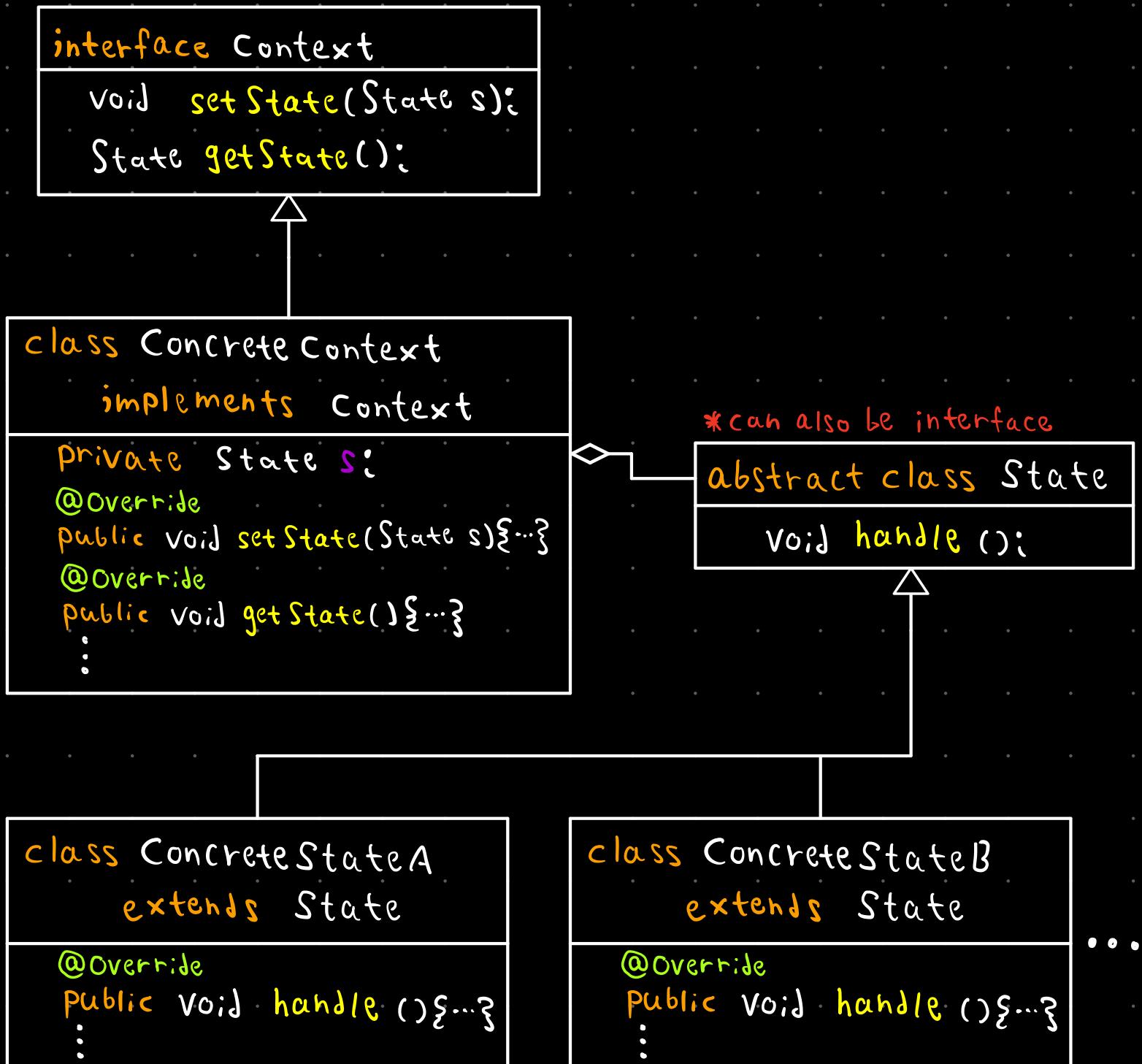
class ConcreteObserverB  
implements Observer

```
Private ConcreteSubject s;  
@Override  
Public Void update(){...}
```

class ConcreteSubject  
implements Subject

```
Private List<Observer> observers;  
@Override  
Public Void attach(Observer o){...}  
@Override  
Public Void detach(Observer o){...}  
@Override  
Public Void notifyObservers(){  
    for(Observer o: this.observers){  
        o.update();  
    }  
}
```

# State



# Singleton

```
class Singleton  
{  
    private Singleton instance;  
    private Singleton(){}  
  
    public Singleton getInstance()  
    {  
        if (instance==null){  
            instance=new Singleton();  
        }  
        return instance;  
    }  
    ...  
}
```

# Memento

```
class Memento
    private String* state;
    Public Memento(String state){...}
    public String getState(){...}
```

\* Doesn't have to be a String

```
class Originator
    private String state;
    public void setState(String state){...}
    public void getState(){...}
    public Memento saveStateToMemento(){
        return new Memento(state);
    }
    public void getStateFromMemento(Memento m){
        state=m.getState();
    }
    ...
    :
```

```
class CareTaker
    private List<Memento> mementoList=...
    Public void add(Memento m){...}
    public Memento get(int index){...}
    ...
    :
```

# Dependency injection

```
class Service  
public void doSomething(){...}  
⋮
```

## Constructor injection:

```
class Client  
private Service service;  
public Client(Service s){  
    this.service=s;  
⋮  
}
```

## Property injection:

```
class Client  
private Service service;  
public void setService(Service s){  
    this.service=s;  
⋮
```

## Method injection:

```
class Client  
public void useService(Service s){  
    s.doSomething();  
⋮
```

# דברים נוספים לזכור

```
import java.io.File;
import java.util.Scanner;
import java.util.Arrays;
import java.util.Random;
import java.util.function; (.Consumer\Predicate\Supplier...)
import java.util.ArrayList; (also List\Map\Set...)
import java.util.regex.Pattern;
import java.util.regex.Matcher;
import java.lang.Math;
import danogl.util.Vector2; (או)
```

