

וְאֵת קָדְשֶׁךָ

ohad.klein@mail.huji.ac.il

# תבניות עצוב

## לתלמיד דום העצב

אלן אונן / נורמן:  


סינסינו:  


טמיון:  


# אין טו איזיון

המפעלים יוצרים מוצאים ומכירתם מושגית. המפעלים יוצרים מוצאים ומכירתם מושגית.

ה**戰略** (Strategy) מוגדרת כהנחייה והכינור של ארגון ל達 את המטרות שלו.

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 יאנו היכר לנו לאיזה גורם גורע  
הנתקה בפונקציית פוטוניותו ומי שמיין, כה אכזרי?

בכל נציגו שפנוי מפזורה נסעה ברכבת. מטרת הרכבת היא לסייע לאנשים לשוב למקומותם ולהרוויח. מטרת הרכבת היא לסייע לאנשים לשוב למקומותם ולהרוויח.

ב-**קווין** מוגדרת **המצב** (state) כ-

המצב **הנוכחי** (current state). ב-

המצב **הנוכחי** (**curr**) מוגדרת **המצבים** (states) כ-

המצבים **האפשריים** (possible states). ב-

המצבים **האפשריים** (**possible states**) מוגדרת **ה פעולה** (action) כ-

ה פעולה **האפשרית** (possible action). ב-

ה פעולה **האפשרית** (**possible action**) מוגדרת **הreward** (reward) כ-

הreward **האפשרי** (possible reward). ב-

הreward **האפשרי** (**possible reward**) מוגדרת **הpenalty** (penalty) כ-

הpenalty **האפשרי** (possible penalty).

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

הערכות הינהsingleton :  
Logger - (אחד כוננה, viele זריה)

15. מנגנון **Memento** מזכיר מה ניכר וחוץ מכך כדי לא לפגוע בכרכינט (כגון סיבת ריבוי ריבוי). מנגנון **Memento** מזכיר מה ניכר וחוץ מכך כדי לא לפגוע בכרכינט (כגון סיבת ריבוי ריבוי).

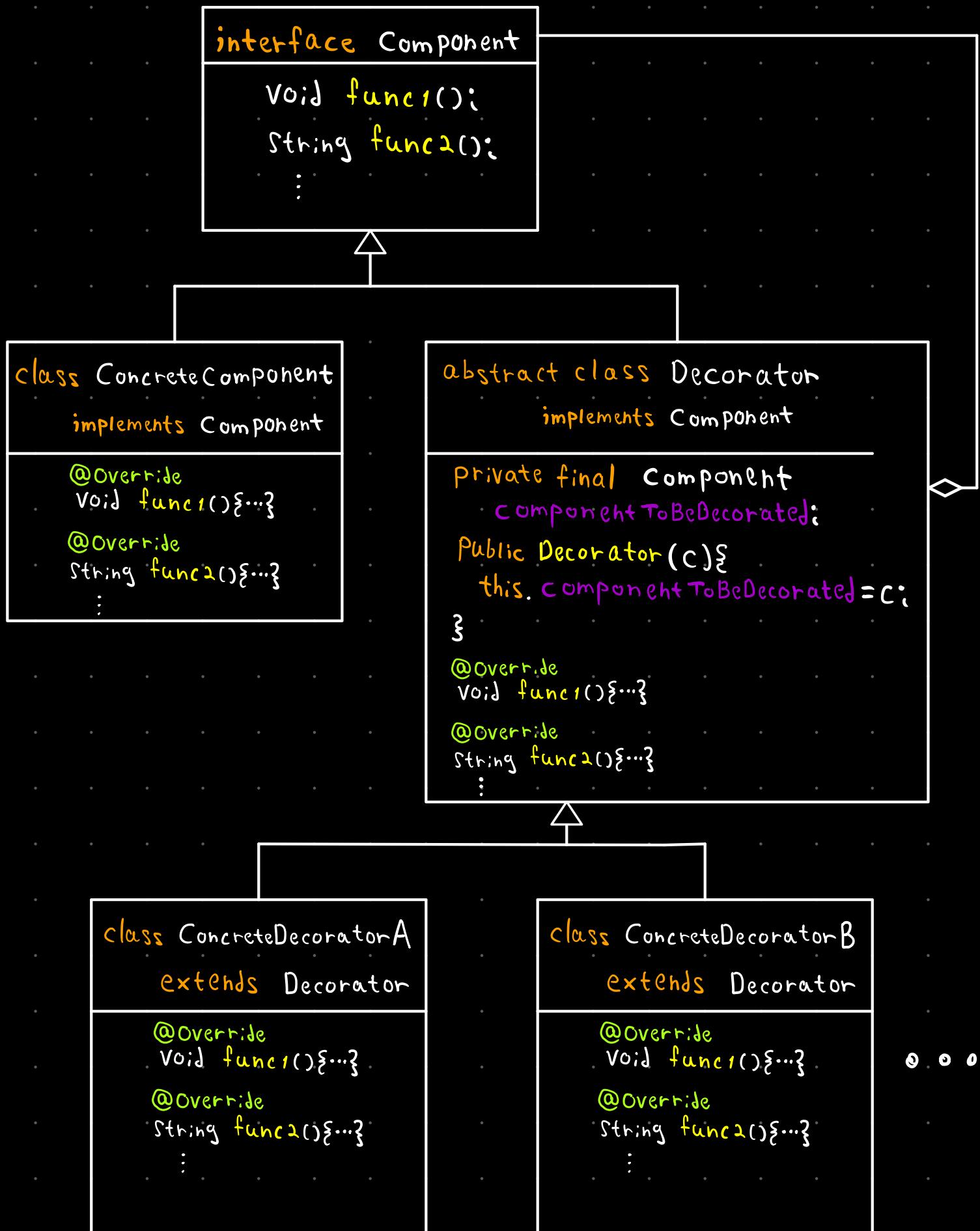


16. מנגנון **Memento** מזכיר מה ניכר וחוץ מכך כדי לא לפגוע בכרכינט (כגון סיבת ריבוי ריבוי). מנגנון **Memento** מזכיר מה ניכר וחוץ מכך כדי לא לפגוע בכרכינט (כגון סיבת ריבוי ריבוי). מנגנון **Memento** מזכיר מה ניכר וחוץ מכך כדי לא לפגוע בכרכינט (כגון סיבת ריבוי ריבוי). מנגנון **Memento** מזכיר מה ניכר וחוץ מכך כדי לא לפגוע בכרכינט (כגון סיבת ריבוי ריבוי). מנגנון **Memento** מזכיר מה ניכר וחוץ מכך כדי לא לפגוע בכרכינט (כגון סיבת ריבוי ריבוי).

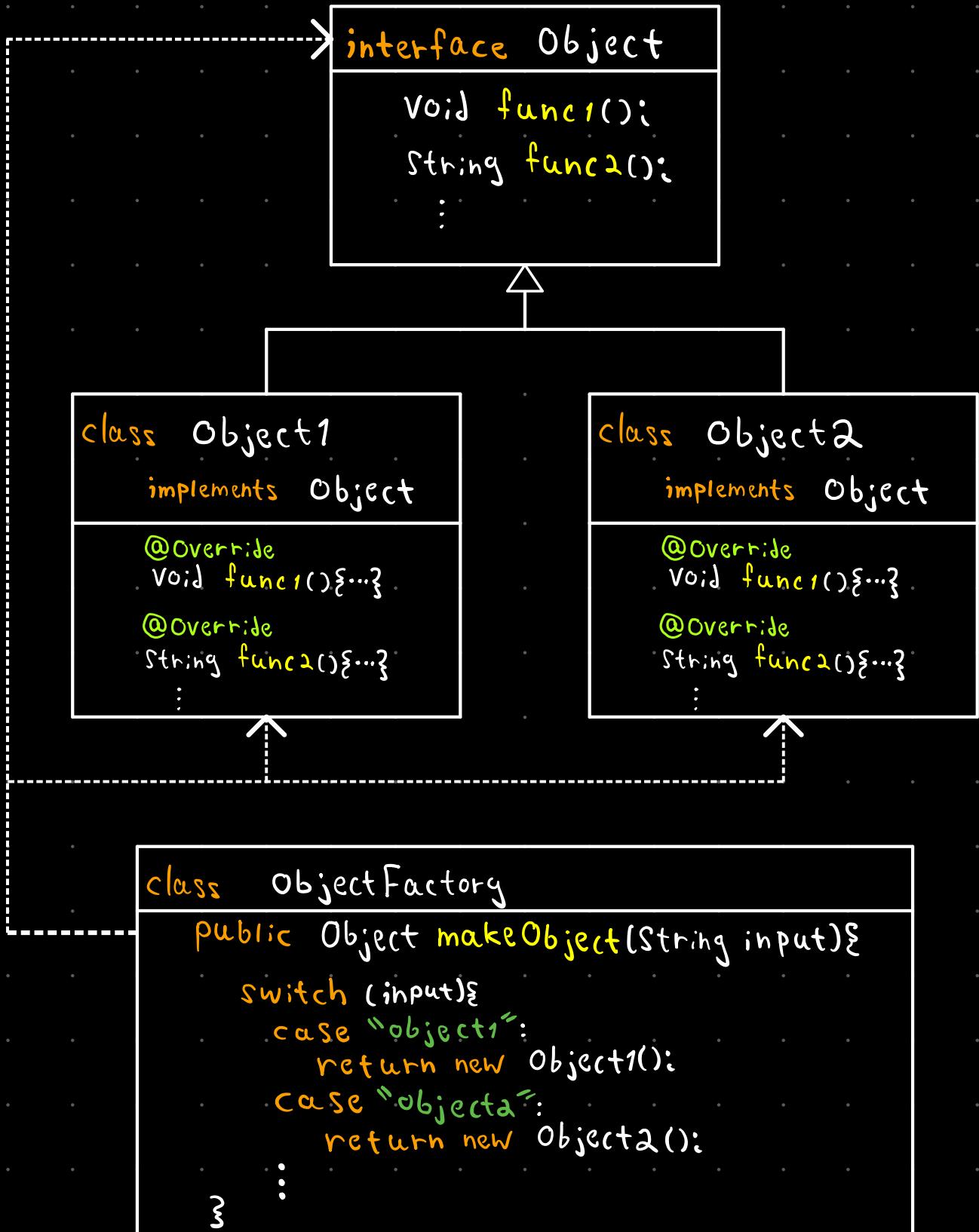


17. מנגנון **Memento** מזכיר מה ניכר וחוץ מכך כדי לא לפגוע בכרכינט (כגון סיבת ריבוי ריבוי). מנגנון **Memento** מזכיר מה ניכר וחוץ מכך כדי לא לפגוע בכרכינט (כגון סיבת ריבוי ריבוי).

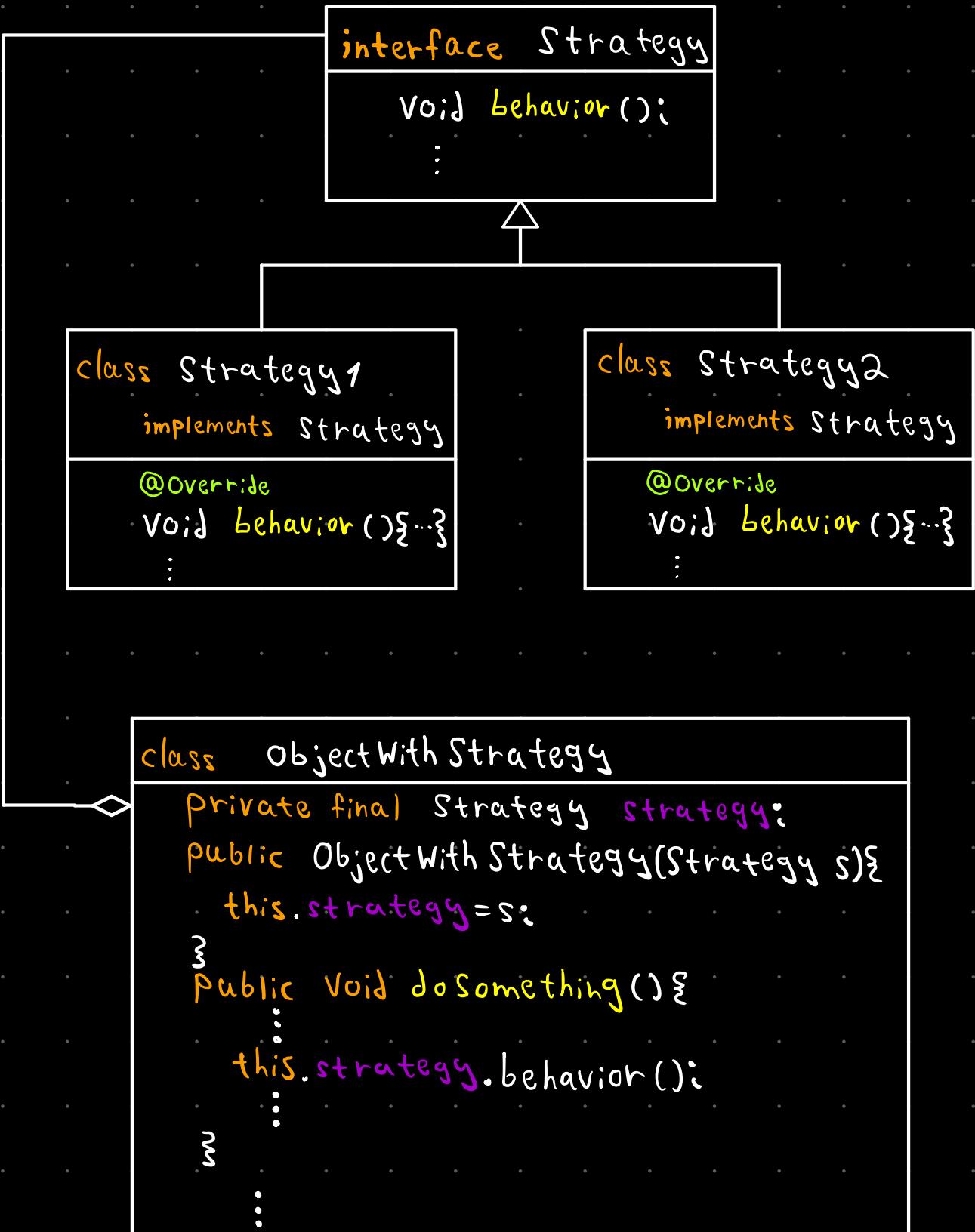
# Decorator



# 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(){...}
```

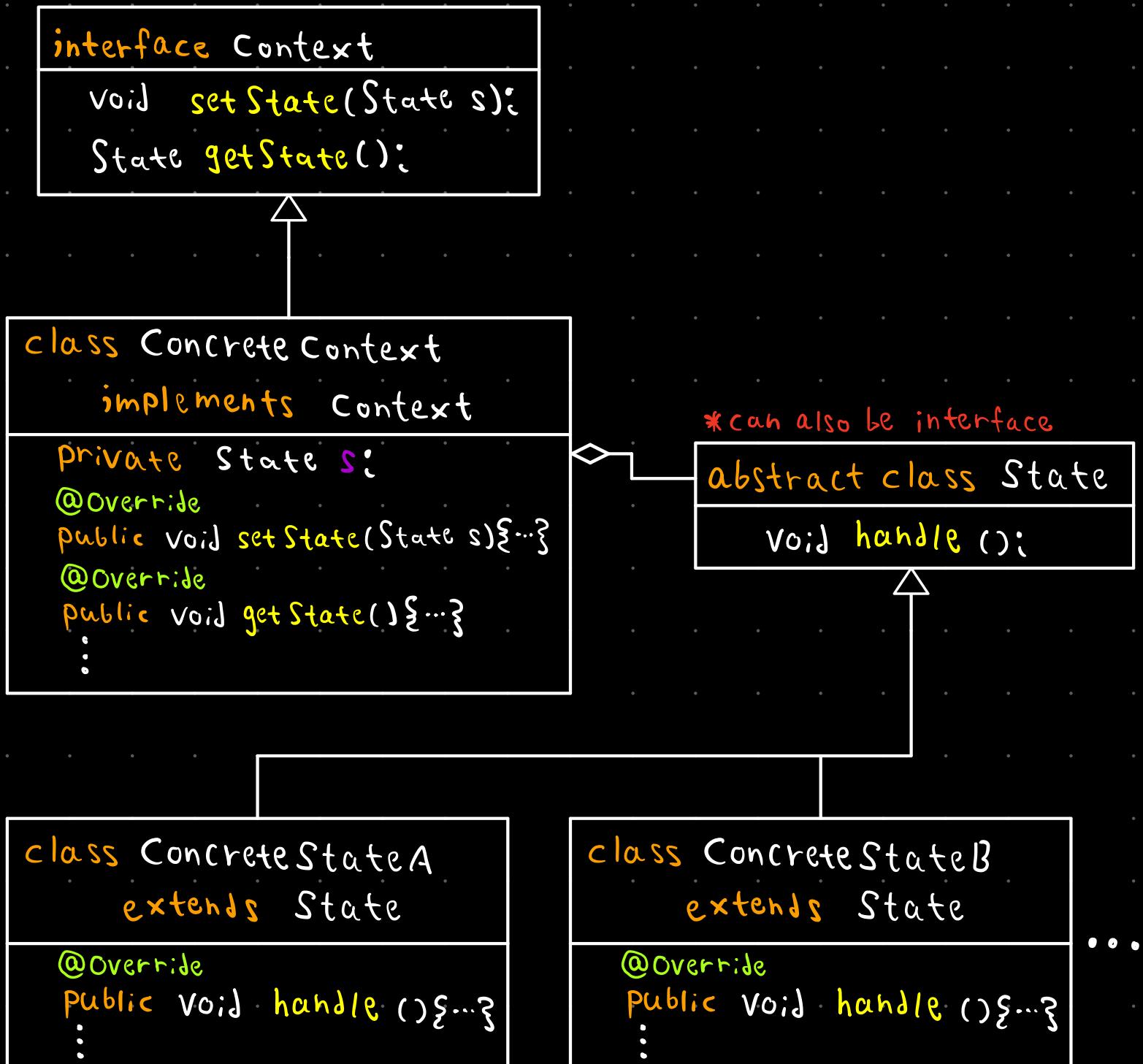
class ConcreteObserverB  
implements Observer

```
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){...}
    ...
    :
```

# 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.lang.Math;
import danogl.util.Vector2; (או)
```

