Decorator - Design Pattern

Présenté par Dian et Ouali

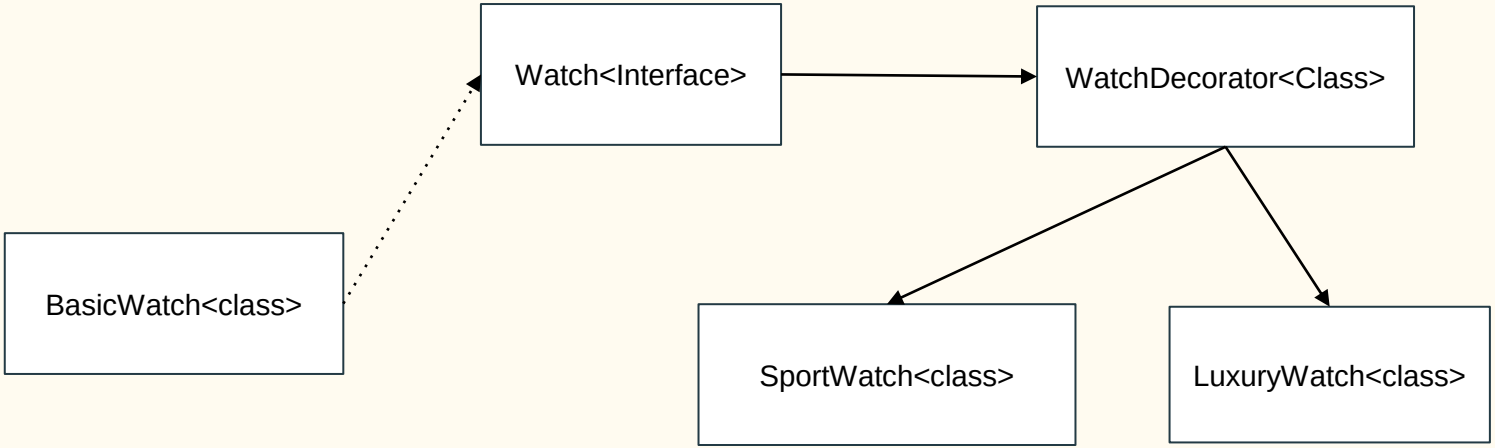
Introduction

Design pattern: Decorator

Le Design pattern decorator est un type de Design pattern Structurel. Il sert à modifier les fonctionnalités d'un objet à l'exécution. Les autres instances de la même classe ne sont pas affectées. Les objets sont modifiés individuellement.

Schéma

Structure du decorator



Conception

Interface

```
interface Watch {
    public void assemble();
}
```

Decorator

```
class WatchDecorator implements Watch {

    protected Watch watch;

    public WatchDecorator(Watch c){
        this.watch=c;
    }

    @Override
    public void assemble() {
        this.watch.assemble();
    }

}
```

SportWatch

```
class SportsWatch extends WatchDecorator {

    public SportsWatch(Watch c) {
        super(c);
    }

    @Override
    public void assemble(){
        super.assemble();
        System.out.print(" Adding features of Sports Watch.");
    }
}
```

LuxuryWatch

```
class LuxuryWatch extends WatchDecorator {

    public LuxuryWatch(Watch c) {
        super(c);
    }

    @Override
    public void assemble(){
        super.assemble();
        System.out.print(" Adding features of Luxury Watch.");
    }
}
```

Use main

```
public class decorator {
    public static void main(String[] args) {
        Watch SportsWatch = new SportsWatch(new BasicWatch());
        SportsWatch.assemble();

        System.out.println("\n*****");

        Watch LuxuryWatch = new SportsWatch(new LuxuryWatch(new BasicWatch()));

        LuxuryWatch.assemble();

        System.out.println("\n*****");
    }
}
```

Conclusion

Design pattern: Decorator

Avantage:
Le design pattern decorator est un design flexible, car il permet d'ajouter facilement des fonctionnalité à l'exécution.

Inconvénient:
Ce design alourdit le code avec de nombreux objets et décorations très similaires.