Hemtenta Exempel III

Mikael Svahnberg*

2020-05-15

Contents

1 Uppgifter **1**
1.1 Uppgift 1 1
1.2 Uppgift 2 2

2 Systembeskrivning **2**

3 Klassdiagram **2**

4 Klassbeskrivning **3**

5 Pseudokod **3**
5.1 create() 3
5.2 cast() 3

6 GRASP patterns **4**

1 Uppgifter

Samma som innan, fast med **Strategy Pattern**

1.1 Uppgift 1

1. Hitta på ett mindre system där du kan använda designmönstret **Strategy pattern**. Beskriv systemet och hur du tänker använda Strategy pattern för att lösa en del av systemet.

2. Gör ett klassdiagram med klasser, attribut och metoder för den del av systemet där ditt Strategy pattern används.

3. Beskriv hur du använt Strategy pattern och vilka klasser och metoder som är inblandade. Vilka roller har de? Vilka ansvarsområden har de?

4. Skriv pseudokod (eller kod i Java eller C++) för de metoder där du skapar objekt enligt ditt Strategy pattern och de metoder där du använder dig av dessa objekt.

*Mikael.Svahnberg@bth.se

1.2 Uppgift 2

1. På vilket sätt är Strategy Pattern relaterat till GRASP-mönstren **High Cohesion** och **Low Coupling**? Beskriv kortfattat.

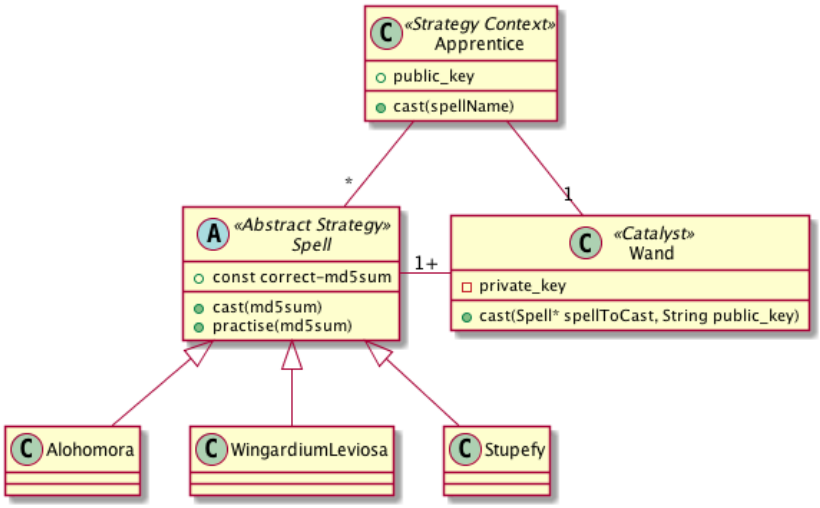
2. Använd ditt system från uppgift 1 och beskriv hur High Cohesion och Low Coupling används för att fördela ansvar mellan klasserna i ditt system.

2 Systembeskrivning

Ett spel där man är en trollkarlslärling och skall lära sig kasta olika trollformler. Strategy pattern används för att representera olika trollformler.

3 Klassdiagram

```
class Apprentice <<Strategy Context>>{
 +cast(spellName)
 +public_key
}

class Wand <<Catalyst>> {
 -private_key
 +cast(Spell* spellToCast, String public_key)
}

abstract Class Spell <<Abstract Strategy>> {
 +cast(md5sum)
 +practise(md5sum)
 +const correct-md5sum
}

Spell <|-- Alohomora
Spell <|-- WingardiumLeviosa
Spell <|-- Stupefy

Apprentice -- "1" Wand
Apprentice -- "*" Spell

Spell - "1+" Wand
```



4 Klassbeskrivning

Apprentice är «context» i Strategy pattern, den har ett antal `Spell`, där varje Spell är en Strategy.

Wand är inte del av strategy pattern men fungerar som katalysator i trollformler.

Spell har rollen «Abstract Strategy», tillhandahåller det gränssnitt (de metoder) som varje strategy skall implementera.

{Alohomora, WingardiuMLeviosa, Stupefy} är «concrete strategy», dvs. de implementerar (på olika vis) gränssnittet som «abstract strategy» har utlovat.

5 Pseudokod

5.1 create()

```
Apprentice::create() {
   Spell* basicSpell = new Alohomora(); mySpells.append(basicSpell);
}
```

5.2 cast()

```
Apprentice::cast(spellName) {
  Spell* spellToCast = mySpells.find(spellName);
  myWand->cast(spellToCast, public_key); // indirection through Wand
  was just for fun, not part of Strategy pattern.
}

Wand::cast(Spell* spellToCast, String public_key) {
  String md5 = this->generateSpellMd5(public_key, private_key); // not
```

```
  part of Strategy pattern, can be ignored SpellToCast->cast(md5);
}

Alohomora::cast(md5sum) {
  // Match md5sum with what should be allowed // if correct, perform
 cast // if incorrect, launch kaboom!
}
```

6 GRASP patterns

- low coupling && high cohesion

Enligt **high cohesion** representeras varje trollformel av en separat klass som är världsbäst på allt om just den formeln, utan att veta någonting om resten av världen.

Enligt **high cohesion** vet Apprentice bara vilka trollformler hen kan, och hur man kastar dem med hjälp av sin trollstav. Detaljerna sköts av andra klasser.

low coupling offras lite för att få separata klasser med varje trollformel.

Low coupling främjas av den abstrakta basklassen `Spell`, eftersom Apprentice inte behöver ha en relation till varje enskild typ av trollformel. Att man kan använda pekare till basklassen `Spell` i stället för att alltid veta vilken subtyp man "håller" i gör också att man blir mer löst kopplad (low coupling) och kan lätt lägga till nya subklasser (trollformler).