PA1458 Example III

Mikael Svahnberg

March 8, 2021

Contents

pattern Strategy Pattern

GRASP1 Low Coupling

GRASP2 High Cohesion

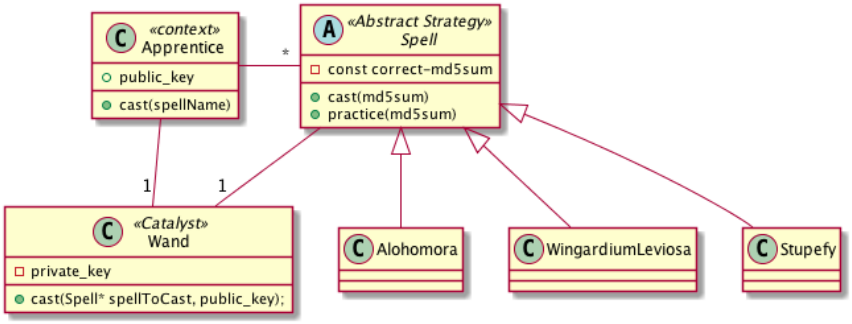
1 System Description

A Game where you play a wizard apprentice. The different spells you can cast will make use of the Strategy pattern.

2 Class Diagram

```
class Apprentice <<context>> {
 +cast(spellName)
```

```
 +public_key
}

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

Apprentice - "*" Spell

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

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

Apprentice -- "1" Wand
Spell -- "1" Wand
```



3 Description of Classes

Apprentice is the «context» in Strategy pattern. Has a number of spells, each spell is a strategy.

Wand is not a part of the strategy pattern, acts as a catalyst in spells.

Spell has the role «abstract strategy», provides the interface that all concrete spells must implement.

{Alohomora, WingardiumLeviosa, Stupefy} are concrete spells.

4 Pseudocode

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


Apprentice::cast(String spellName) {
  Spell* spellToCast = mySpells.find(spellName);
  myWand->cast(spellToCast, public_key);
}

// Not really part of the strategy pattern, let's have some fun instead
Wand::cast(Spell* spellToCast, public_key) {
  String md5 = this->generateSpellMD5(spellToCast, public_key, private_key);

  spellToCast->cast(md5);
}

Alohomora::cast(md5sum) {
  // Match md5sum with the correct_md5sum
  // if correct, perform spell-cast
  // if incorrect, launch Kaboom!
}
```

5 Discussion of Design Pattern wrt. GRASP patterns

GRASP1 Low Coupling

GRASP2 High Cohesion

Since the «context» only has a relation to the «abstract strategy» and does not need a specific association to any of the concrete strategies, this is

an example of **low coupling**. The concrete strategies do not need to know anything about how and where they are used, which is also **low coupling**.

High Cohesion : Each class has well defined responsibilities. The «context» only has a pointer to the current strategy, and knows about the available strategies. The «abstract strategy» class is only responsible for the interface, and the concrete strategy classes only know about the actual strategy that each of them implement.

6 Usage of GRASP Patterns

We follow the basic principles for the Strategy pattern, as described above. In addition, everything involved in calculating md5sums from public and private keys is delegated to a separate class, the **Wand** class, which keeps the set of responsibilities for the Apprentice class simple. Thus, we continue ensure high cohesion.