Exempel Hemtenta

Mikael Svahnberg*

2020-05-12

Contents

1 Uppgifter

1.1 Uppgift 1

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

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

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

*Mikael.Svahnberg@bth.se

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

1.2 Uppgift 2

1. På vilket sätt är Observer Pattern relaterat till GRASP-mönstren **Information Expert** och **Controller**? Beskriv kortfattat.

2. Använd ditt system från uppgift 1 och beskriv hur Information Expert och Controller används för att fördela ansvar mellan klasserna i ditt system.

2 Systembeskrivning

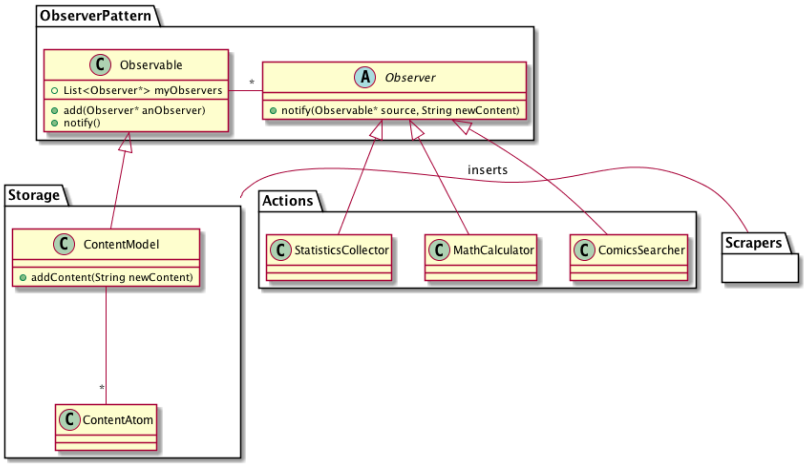
Systemet letar efter nyckelord på olika sociala medier, chatprogram och websidor, och när nyckelorden hittas så skall olika saker hända.

Observer Pattern används för att meddela när nytt innehåll har hittats, så att "olika saker" kan notifieras och eventuellt göra det de skall.

3 Klassdiagram

```
package Scrapers {
' görs inte i uppgiften, använder sig inte av Observer Pattern.
}

package ObserverPattern {
class Observable {
+add(Observer* anObserver)
+notify()
+List<Observer*> myObservers
}

abstract class Observer {
+notify(Observable* source, String newContent)
}

Observable - "*" Observer
}

package Storage {

class ContentModel {
+addContent(String newContent)
}

ContentModel -- "*" ContentAtom

Observable <|-- ContentModel
}
```

```
package Actions {


Observer <|-- StatisticsCollector
Observer <|-- MathCalculator
Observer <|-- ComicsSearcher
}

Scrapers - Storage : inserts
```



4 Beskrivning av Klassdiagram

Klassdiagrammet har ett antal olika paket:

Scrapers Samlar information från t.ex. sociala media, skickar till Storage för att sparas.

Storage Skapar ContentAtom av nytt innehåll och lagrar. Meddelar sedan Actions att det finns nytt innehåll.

Actions Reagerar (eventuellt) på nytt innehåll

ObserverPattern Innehåller de klasser som behövs för ett generiskt Observer pattern.

4.1 Storage Paketet

Klassen ContentModel tar emot nytt innehåll, skapar ContentAtoms och har rollen Observable. Detta innebär att när det finns nytt innehåll skall den meddela "sina" observers om detta.

4.2 Actions-Paketet

Klassen ComicsSearcher Söka på ett antal serie-websidor efter de angivna nyckelorden och returnera dessa. Klassen har rollen Observer, vilket innebär att den prenumererar på förändringar som klassen ContentModel publicerar.

5 Pseudokod

5.1 Observable::add(Observer*)

```
void Observable::add(Observer* newObserver) {
 myObservers.append(newObserver);
}
```

5.2 Observable::notify(String newContent)

```
void Observable::notify(String newContent)
 myObservers.forEach( function(o) {
  o.notify(this, newContent);
});
```

5.3 ContentModel::addContent(String newContent)

```
void ContentModel::addContent(String newContent) {
 ContentAtom atom = new ContentAtom(newContent);
 DBHandler::store(atom);
 this->nofity(newContent); // Här använder vi Observer-pattern
}
```

5.4 ComicsSearcher::notify(Observable* source, String newContent)

```
void ComicsSearcher::notify(Observable* source, String newContent) {
 String key = newContent.split()[0];
 if (myKeywords.find(key)) {
   // Do relevant stuff
 }
}
```

6 Diskussion om GRASP-mönster

1. På vilket sätt är Observer Pattern relaterat till GRASP-mönstren **Information Expert** och **Controller**? Beskriv kortfattat.

Observable (och de som ärver från Observable) är information expert om vilka Observers som finns. Den är också en controller som delegerar ansvaret för att *agera* på ny information till respektive Observer. Den ser till att alla observers får en chans det de vill.

Subklasserna till Observer är information expert på exakt vad som skall hända när en viss ny information kommer (när Observable-klassen skickar ut den nya informationen)

7 Användning av GRASP-mönster

1. Använd ditt system från uppgift 1 och beskriv hur Information Expert och Controller används för att fördela ansvar mellan klasserna i ditt system.

ContentModel är InformationExpert på hur innehållet skall lagras och när Observers skall meddelas. ContentModel är de facto Information Expert på vilka Observers som finns. ContentModel är controller som distribuerar kontroll till var och en av Observers när det finns ny information så de kan göra det de vill.

t.ex. ComicSearcher är information expert på vilka nyckelord som den skall agera på, och vad som skall hända när dessa nyckelord nämns. t.ex. StatisticsCollector är information expert på vilka nyckelord som den skall samla statistik om, och hur denna statistik skall samlas, beskrivas, och lagras.