PA1458 Hemtenta Exempel I

Mikael Svahnberg

May 17, 2021

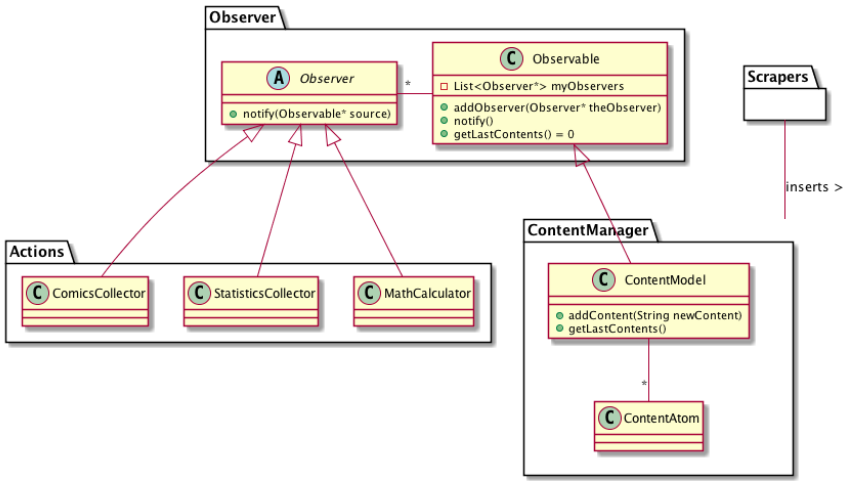
Contents

1 Systembeskrivning

Ett system som hämtar information från sociala media och agerar på den. Givet innehållet skall t.ex. statistik uppdateras, en seriestrip hämtas, eller ett mattetal räknas ut. Observer används för att berätta för "actions" att det finns nytt innehåll att titta på.

2 Klassdiagram

```
package Observer {
 abstract class Observer {
  +notify(Observable* source)
 }

 class Observable {
 -List<Observer*> myObservers
 +addObserver(Observer* theObserver)
 +notify()
 +getLastContents() = 0
 }

 Observable - "*" Observer
}

package Scrapers {
}

package ContentManager {
 class ContentModel {
  +addContent(String newContent)
  +getLastContents()
 }

 class ContentAtom

 ContentModel -- "*" ContentAtom
 Observable <|-- ContentModel
}

package Actions {
 Observer <|-- StatisticsCollector
 Observer <|-- MathCalculator
 Observer <|-- ComicsCollector
}

Scrapers -- ContentManager : inserts >
```



3 Beskrivning av hur designmönstret används

Klassdiagrammet består av ett antal paket:

Scrapers är inte en del av Observer pattern, men samlar data och ger till ContentManager

ContentManager lagrar innehåll som `ContentAtom` och meddelar alla `Observers` att det finns nytt innehåll.

Actions Reagerar (eventuellt) på nytt innehåll

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

3.1 Paketet ContentManager

klassen ContentModel tar emot nytt innehåll, skapar en `ContentAtom` och meddelar alla `Observers` att det finns nytt innehåll.

3.2 Paketet Actions

Innehåller alla klasser som agerar på innehåll. Till exempel ComicsCollector letar efter vissa nyckelord och söker efter en serisstrip som matchar dessa ord.

4 Pseudokod

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

Observable::notify() {
 myObservers.forEach( function(o) {
   o.notify(this);
 });
}

ContentModel::addContent(String newContent) {
 ContentAtom* atom = new ContentAtom(newContent);
 DBHandler::store(atom);
 this->notify();
}

ComicsCollector::notify(Observable* source) {
 String contents = source->getLastContents();
 String keyword = contents.split()[0];
 if(myKeywords.find(keyword) {
   // Search for comic with all the keywords
 }
}
```

5 Designmönstrets användande av GRASP

- Information Expert
- Controller

Observable (och de som ärver från Observable) är `information Expert` på vilka observers som finns (och vilken metod som finns i alla implementationer av `Observer` (dvs `notify()` - metoden. Observable är också en `controller` som delegerar ansvaret för att agera på ny information till de olika implementationerna av `Observer`.

Subklasserna till `Observer` är `information expert` på vad som skall hända när en viss typ av information blir tillgänglig.

6 Systemets användande av GRASP

`ContentModel` är Information Expert på hur nytt innehåll skall lagras. Den är också information expert på när det är dags att anropa sin `notify()` - metod.

`ContentModel` är controller som delegerar till sina Observers (Actions) när det finns något som skall göras.

De olika klasserna i paketet Actions (t.ex. `ComicsCollector`) är information experts dels på vilka nyckelord som skall trigfga dem, dels på vad som skall göras när dessa nyckelord dyker upp.