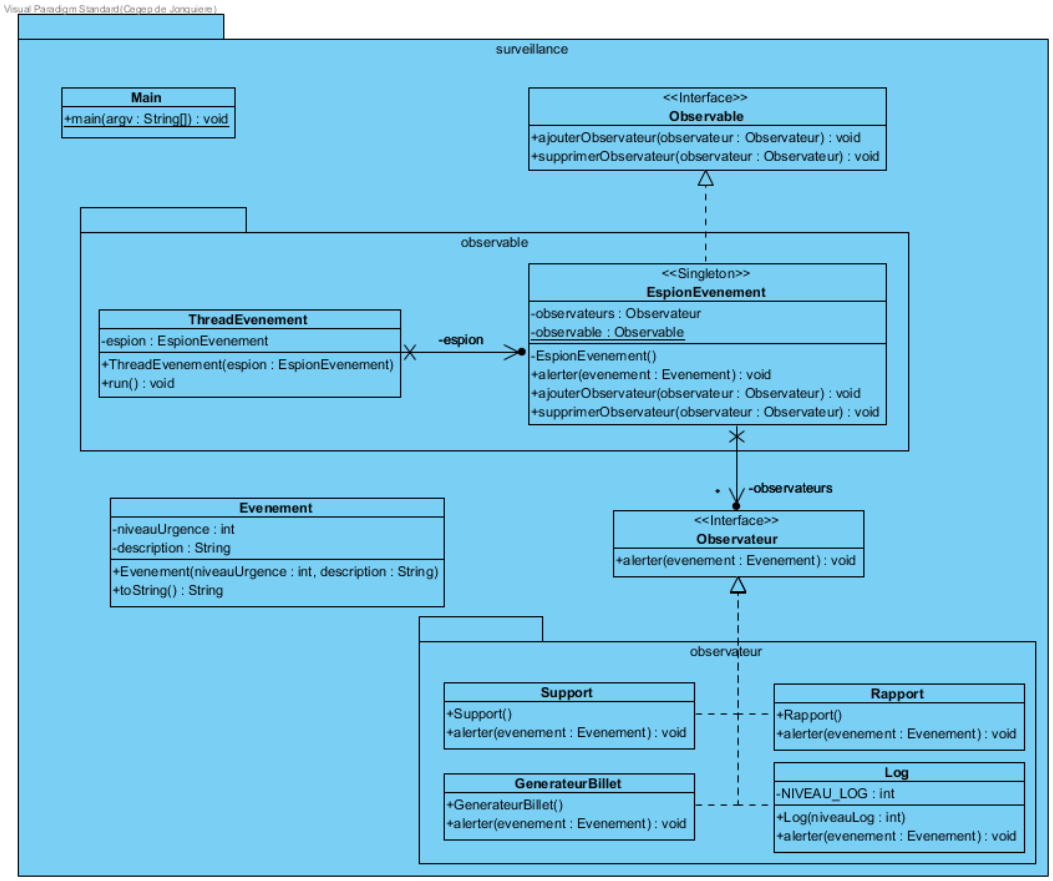
Patron #6 : Observateur



Mise en contexte

Dans cet exemple, nous avons deux groupes de classes à réaliser. Dans le premier groupe, nous avons un thread qui va générer des évènements aléatoirement et les fournir à un observable (on peut remarquer que EspionEvenement est un Singleton et que la méthode statique retourne l'objet sous sa forme « Observable »). Dans le second groupe, nous retrouvons quatre observateurs qui ont, comme seul point commun, l'interface Observateur. Ils ne font qu'une seule opération (dans le constructeur) qui consiste à s'ajouter eux-mêmes en tant qu'observateur (à l'observable)

Contraintes

- La classe « **ThreadEvenement** »
  - Hérite de la classe « Thread »
  - La méthode « run » doit être redéfini et respecter l'algorithmique suivante

Attendre 1000 milisecondes
Générer un évènement aléatoirement

```
descriptionsEvenement = new String[]{
        "Feu", "Fuite eau", "panne de courant",
        "Greve", "Bris majeur"
};
```

```
new Evenement((int)(Math.random() * 100),
 descriptionsEvenement[(int)(Math.random() * descriptionsEvenement.length)]);
```

- La classe « **EspionEvenement** »
  - *Errata UML* : l'attribut « observateurs » est de type **List<Observateur>**
- Indices pour les méthodes « **alerter** » des quatre « **Observateurs** »
  - **Log** : Si le niveau d'urgence de l'évènement est supérieur à NIVEAU_LOG, afficher
  - **Rapport** : Afficher la description de l'évènement seulement
  - **GenerateurBillet** : Si c'est un bris majeur, ouvrir un billet (afficher l'évènement)
  - **Support** : Si le niveau d'urgence de l'évènement est supérieur à 90, afficher

Exemple de main ainsi qu'un exemple d'exécution

```
Log log = new Log( niveauLog: 50);
Rapport rapport = new Rapport();
Support support = new Support();
GenerateurBillet generateurBillet = new GenerateurBillet();

Thread.sleep( millis: 5000);

EspionEvenement.getObservable().supprimerObservateur(log);
EspionEvenement.getObservable().supprimerObservateur(rapport);
EspionEvenement.getObservable().supprimerObservateur(support);
```

```
LOG : Evenement{niveauUrgence=79, description='Feu'}
RAPPORT : Feu
LOG : Evenement{niveauUrgence=68, description='Bris majeur'}
RAPPORT : Bris majeur
OUVERTURE d'UN BILLET : Evenement{niveauUrgence=68, description='Bris majeur'}
LOG : Evenement{niveauUrgence=61, description='Fuite eau'}
RAPPORT : Fuite eau
RAPPORT : Fuite eau
LOG : Evenement{niveauUrgence=94, description='panne de courant'}
RAPPORT : panne de courant
SUPPORT URGENCE : Evenement{niveauUrgence=94, description='panne de courant'}
RAPPORT : Fuite eau
LOG : Evenement{niveauUrgence=53, description='Greve'}
RAPPORT : Greve
LOG : Evenement{niveauUrgence=97, description='Fuite eau'}
RAPPORT : Fuite eau
SUPPORT URGENCE : Evenement{niveauUrgence=97, description='Fuite eau'}
LOG : Evenement{niveauUrgence=75, description='Greve'}
RAPPORT : Greve
OUVERTURE d'UN BILLET : Evenement{niveauUrgence=73, description='Bris majeur'}
OUVERTURE d'UN BILLET : Evenement{niveauUrgence=37, description='Bris majeur'}
OUVERTURE d'UN BILLET : Evenement{niveauUrgence=13, description='Bris majeur'}
OUVERTURE d'UN BILLET : Evenement{niveauUrgence=74, description='Bris majeur'}
```