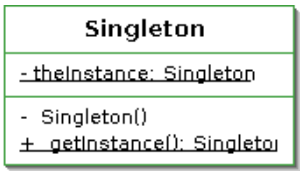
Design Pattern : singleton

Intent

Ensure a class only has one instance, and provide a global point of access to it.

Structure



Eléments caractéristiques

- constructeur privé
- une instance `static final` soit `public` soit `private` mais alors accessible par une méthode `public static`

```
public class SingletonClass {
  private SingletonClass() { }
  private static final SingletonClass UNIQUE_INSTANCE = new SingletonClass();
  public static SingletonClass getTheInstance() {
    return UNIQUE_INSTANCE;
  }
}
```

ou

```
public class SingletonClass {
  private SingletonClass() { }
  public static final SingletonClass SINGLETON = new SingletonClass();
}
```

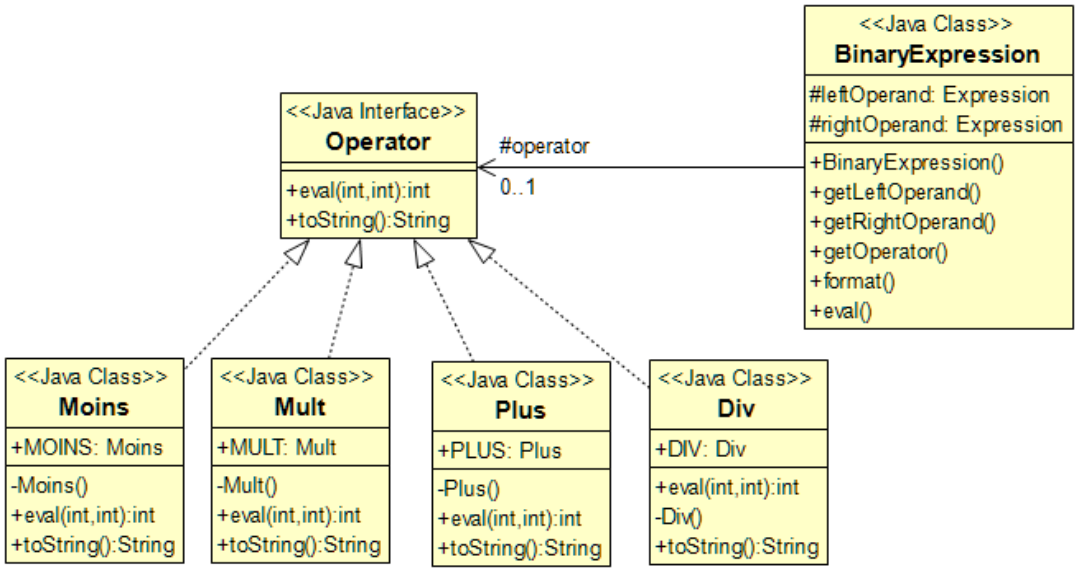
Remarque : ce design pattern peut toujours s'appliquer pour les classes « sans état » (pas d'attribut).

Exemples rencontrés

TD – Casse-briques : le joueur *« Dans cette version du jeu il n'y a nécessairement qu'un seul joueur. Une seule instance de la classe `Player` est donc nécessaire. Comme prendre en compte cela dans la conception de cette classe ? »*

Player
- nbBalls : int - paddle : Paddle - nbPoints : int + INSTANCE : Player
- Player() + addPoints(points : int) + changeNbBalls(diff : int) + changePaddleSize (scale : float)

TD – Expressions : gestion des opérateurs Chaque opérateur peut être représenté par une classe singleton. Par rapport à un type énuméré, cela offre la possibilité d'ajouter de nouveaux opérateurs (*modulo* par exemple) : respect du *principe ouvert-fermé*.



TD – Parc d'attractions Parmi les stratégies de gestion de la politique de prix et des contraintes d'accès à une attraction. La mise en place d'un « prix gratuit » ou de l'absence de restriction d'accès correspond à des classes « sans état » qui peuvent donc être mises en œuvre par des singletons :

```
public class NoFeePolicy extends FixedPrice {
  public static final NoFeePolicy SINGLETON = new NoFeePolicy();
  private NoFeePolicy() {
    super(0);
  }
}
```

```
public class OpenAccess implements AccessStrategy {
  private static final String FREE_ACCESS_FOR_EVERYONE = "free access for everyone";
  public static final OpenAccess SINGLETON = new OpenAccess();
  private OpenAccess() {}

  public boolean accessGranted(Person person) {
    return true;
  }
  public String description() {
    return FREE_ACCESS_FOR_EVERYONE;
  }
}
```