ENCART CAMÉRA

Zoom sur le patron Singleton

Sébastien Mosser - INF5153
Chapitre 6 - Capsule 4
Automne 2020

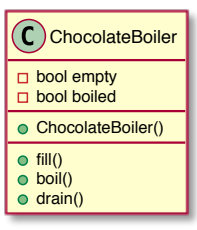
ENCART CAMÉRA

Classification des patrons de conception

Portée		Objectif: Création	Objectif: Structure	Objectif: Comportement
	Classe			
	Objet	Singleton		

ENCART CAMÉRA

Exemple : Fabrique de chocolat




```
public class ChocolateBoiler {

  private boolean empty;
  private boolean boiled;

  public ChocolateBoiler() {
    empty = true;
    boiled = false;
  }

  public void fill() {
    if (isEmpty()) {
      empty = false;
      boiled = false;
    }
  }
```

```
  public void drain() {
    if (!isEmpty() && isBoiled()) {
      empty = true;
    }
  }

  public void boil() {
    if (!isEmpty() && !isBoiled()) {
      boiled = true;
    }
  }
```

Exemple : Fabrique de chocolat

ENCART CAMÉRA

```
class ChocolateFactory {

  private ChocolateBoiler theBoiler;

  public ChocolateFactory() {
    this.theBoiler = new ChocolateBoiler();
  }

  public void makeChocolateBars() {
    // ... preprocessing
    if (theBoiler.isEmpty()) {
      theBoiler.fill();
      theBoiler.boil();
      theBoiler.drain();
    } else {
      throw new RuntimeException("Busy Boiler!");
    }
    // ... postprocessing
  }

}
```


Exemple : Fabrique de chocolat

ENCART CAMÉRA

```
public void makeChocolateTruffles() {
  // ... preprocessing
  theBoiler = new ChocolateBoiler();
  if (theBoiler.isEmpty()) {
    theBoiler.fill();
    theBoiler.boil();
    theBoiler.drain();
  } else {
    throw new RuntimeException("Busy Boiler!");
  }
  // ... postprocessing
}
```

Erreur de débutant
Manque de documentation
... whatever ...

Problème

ENCART CAMÉRA

N'importe qui peut créer une instance de ChocolateBoiler

Pourtant il n'en existe qu'une seule

C'est au développeur de faire attention.

ENCART CAMÉRA

C'est au dévelopeur de faire attention.

ENCART CAMÉRA

SINGLETON — Object Creational

Intent

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

Motivation

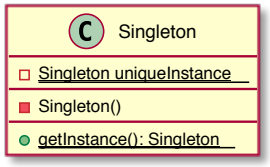
It's important for some classes to have exactly one instance. Although there can be many printers in a system, there should be only one printer spooler. There should be only one file system and one window manager. A digital filter will have one A/D converter. An accounting system will be dedicated to serving one company.

How do we ensure that a class has only one instance and that the instance is easily accessible? A global variable makes an object accessible, but it doesn't keep you from instantiating multiple objects.

A better solution is to make the class itself responsible for keeping track of its sole instance. The class can ensure that no other instance can be created (by intercepting requests to create new objects), and it can provide a way to access the instance. This is the Singleton pattern.

ENCART CAMÉRA

Garantir l'unicité, par construction



Instance statique, Constructeur } privé

Accesseur } public

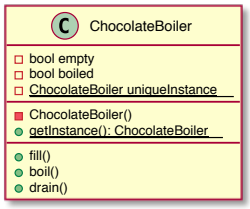
```
class Singleton {

  private static Singleton uniqueInstance = null;

  private Singleton() { ... }

  public static Singleton getInstance() {
    if (uniqueInstance == null)
      uniqueInstance = new Singleton();
    return uniqueInstance;
  }
  Singleton s = new Singleton();
}
```



```
Singleton s = Singleton.getInstance();
```



```
public class ChocolateBoiler {

  private boolean empty;
  private boolean boiled;

  private static ChocolateBoiler uniqueInstance = null;

  private ChocolateBoiler() {
    empty = true;
    boiled = false;
  }

  public static ChocolateBoiler getInstance() {
    if (uniqueInstance == null)
      uniqueInstance = new ChocolateBoiler();
    return uniqueInstance;
  }

}
```

Même en cas d'erreur de programmation, on parle toujours à la même instance de ChocolateBoiler !

```
class ChocolateFactory {

  private ChocolateBoiler theBoiler;
  public ChocolateFactory() {
    // this.theBoiler = new ChocolateBoiler();
    this.theBoiler = ChocolateBoiler.getInstance();
  }

  public void makeChocolateBars() {
    // ... preprocessing
    if (theBoiler.isEmpty()) {
      theBoiler.fill(); theBoiler.boil();
      theBoiler.drain();
    } else {
      throw new RuntimeException("Busy Boiler!");
    }
    // ... postprocessing
  }

  public void makeChocolateTruffles() {
    // ... preprocessing
    // theBoiler = new ChocolateBoiler();
    theBoiler = ChocolateBoiler.getInstance();
    if (theBoiler.isEmpty()) {
      theBoiler.fill(); theBoiler.boil();
      theBoiler.drain();
    } else {
      throw new RuntimeException("Busy Boiler!");
    }
    // ... postprocessing
  }
}
```


Vu que ça repose sur des éléments statiques, ça rend compliqué le code concurrent (et donc les tests)

Danger !

Le Singleton est classiquement mal utilisé, et apporte plus de problèmes que de solutions dans de nombreux cas.

Le singleton est un symptôme de stupidité

STUPID code, seriously?

This may hurt your feelings, but you have probably written STUPID code already. I have too. But, what does that mean?

- **S**ingleton
- **T**ight Coupling
- **U**ntestability
- **P**remature Optimization
- **I**ndescriptive Naming
- **D**uplication

https://williamdurand.fr/2013/07/30/from-stupid-to-solid-code/

Différencier le bon singleton du mauvais singleton

- Le singleton est une **instance unique, statique,** d'une classe

- **Le bon singleton**
  - Gère une instance à **responsabilité unique, sans état**
  - *Par-exemple : Logger, cache, formatage, accès au matériel*

- **Le mauvais singleton**
  - Est un **raccourci** 🐷 pour éviter de passer un objet en paramètre
  - *Générateur de bogues extrêmement difficile à investiguer !*

Autres problèmes associés au Singleton

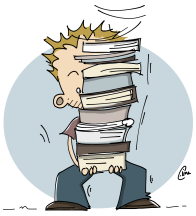
ENCART CAMÉRA

- Défini un **graphe de dépendances implicites** entre objets
- Difficile à **tester**
  - Un Singleton est **fortement couplé** à ses utilisateurs
  - Quand on les teste, on doit **tester l'état global jusqu'au singleton**
- **Concurrence** et accès parallèle
  - Problème de ***thread-safety***
  - Ou, à l'inverse, **goulot d'étranglement**

UQÀM | Département d'informatique

FACULTÉ DES SCIENCES
Université du Québec à Montréal

ENCART CAMÉRA



https://mosser.github.io/

ace

https://ace-design.github.io/

***Abonne toi** à la chaine,*
et met un pouce bleu !