Factory, Service Object

Leleux Laurent

2016 - 2017

Instanciation

- Création d'instances
- « new »
- Pas d'encapsulation
- Pas de polymorphisme

Polymorphisme

« Le polymorphisme est l'idée de permettre à un même code d'être utilisé avec différents types, ce qui permet des implémentations plus abstraites et générales. »

Polymorphisme

« En proposant d'utiliser un même nom de méthode pour plusieurs types d'objets différents, le polymorphisme permet une programmation beaucoup plus générique. Le développeur n'a pas à savoir, lorsqu'il programme une méthode, le type précis de l'objet sur lequel la méthode va s'appliquer. Il lui suffit de savoir que cet objet implémentera la méthode. »

Encapsulation

« L’encapsulation est l'idée de protéger l'information contenue dans un objet et de ne proposer que des méthodes de manipulation de cet objet. Ainsi, les propriétés contenues dans l'objet seront assurés/validés par les méthodes de l'objet et ne seront plus de la responsabilité de l'utilisateur extérieur. »

Factory

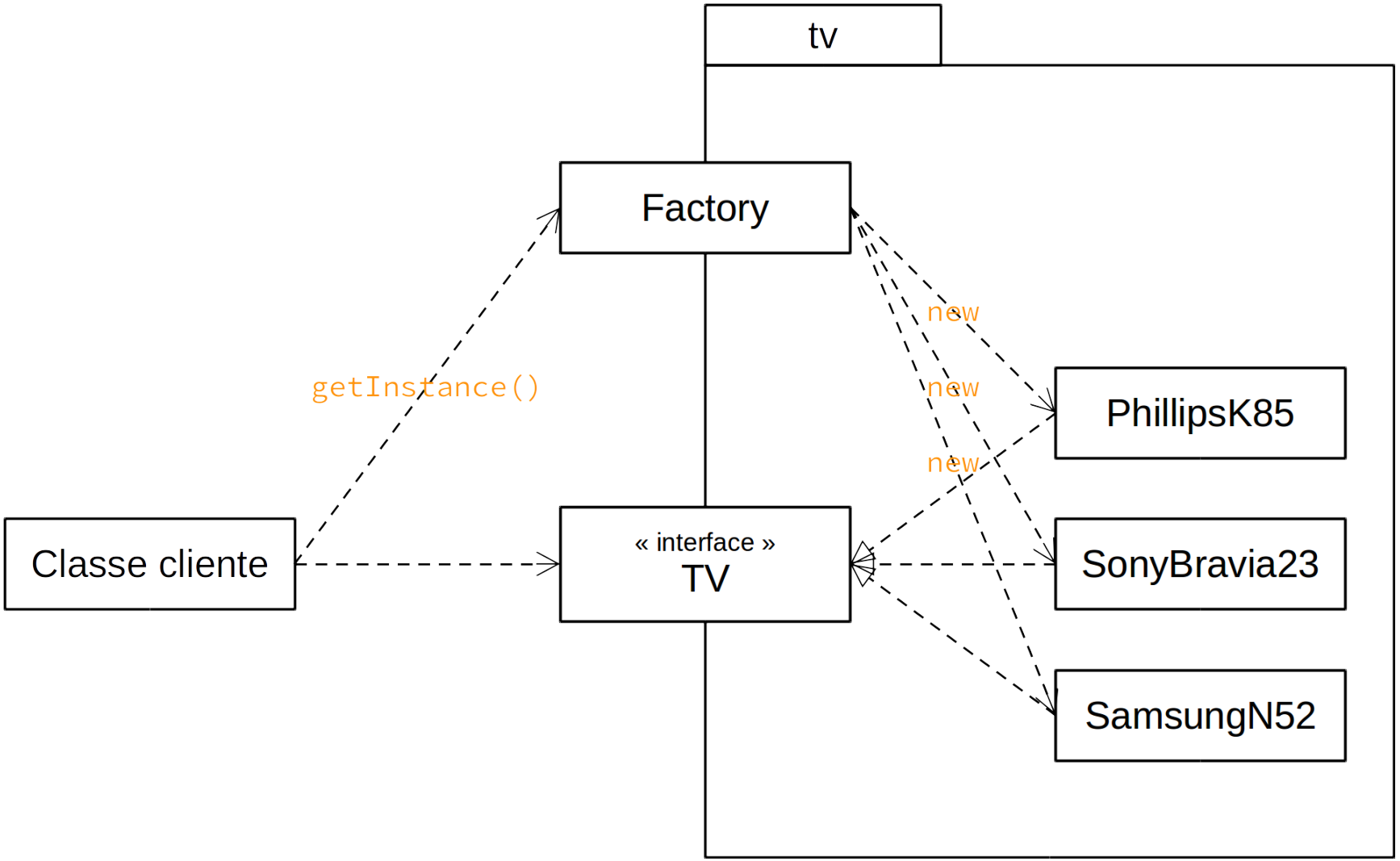
- Encapsuler l'instanciation
- Masquer le « new »
- Ajouter une stratégie d'instanciation
- Dépendances abstraites

Méthodes

- `TvFactory.getInstance()`
- `DvdFactory.getInstance(caracteristiques)`
- `ReveilFactory.getInstance(type)`
- `….getInstance(type, caracteristiques)`

- /!\ Type de retour

Exemple


tv
Factory
new
new
new
getInstance()
PhillipsK85
Classe cliente
« interface »
TV
SonyBravia23
SamsungN52

Bénéfices

- Le client
  - ne connaît aucune classe
  - connaît que des abstractions
  - est indépendant des implémentations
  - utilise le « dynamic binding »

- Stratégies

A retenir !

« Soyez le plus possible insensible aux changements qui se produisent dans les autres packages ! »

« Déléguez la logique d'instanciation des objets éloignés aux factories des packages éloignés »

« Utilisez les interfaces proposés par les packages éloignés, qui exposent les services rendus par les objets éloignés »

Problème

`getInstance()` n'est pas polymorphe

- Toujours le même code
- Pas de changement possible
- Toujours la même factory

Service Object

- Classe / Objet utilitaire
- Plus de static !
- Objet caché par une interface
- Liaison à l'exécution

- Stateless

Service Object

- Caché dans l'interface :

```
MonInterface INSTANCE = (MonInterface) new ClasseUtilitaire();
```

- Utilisation :

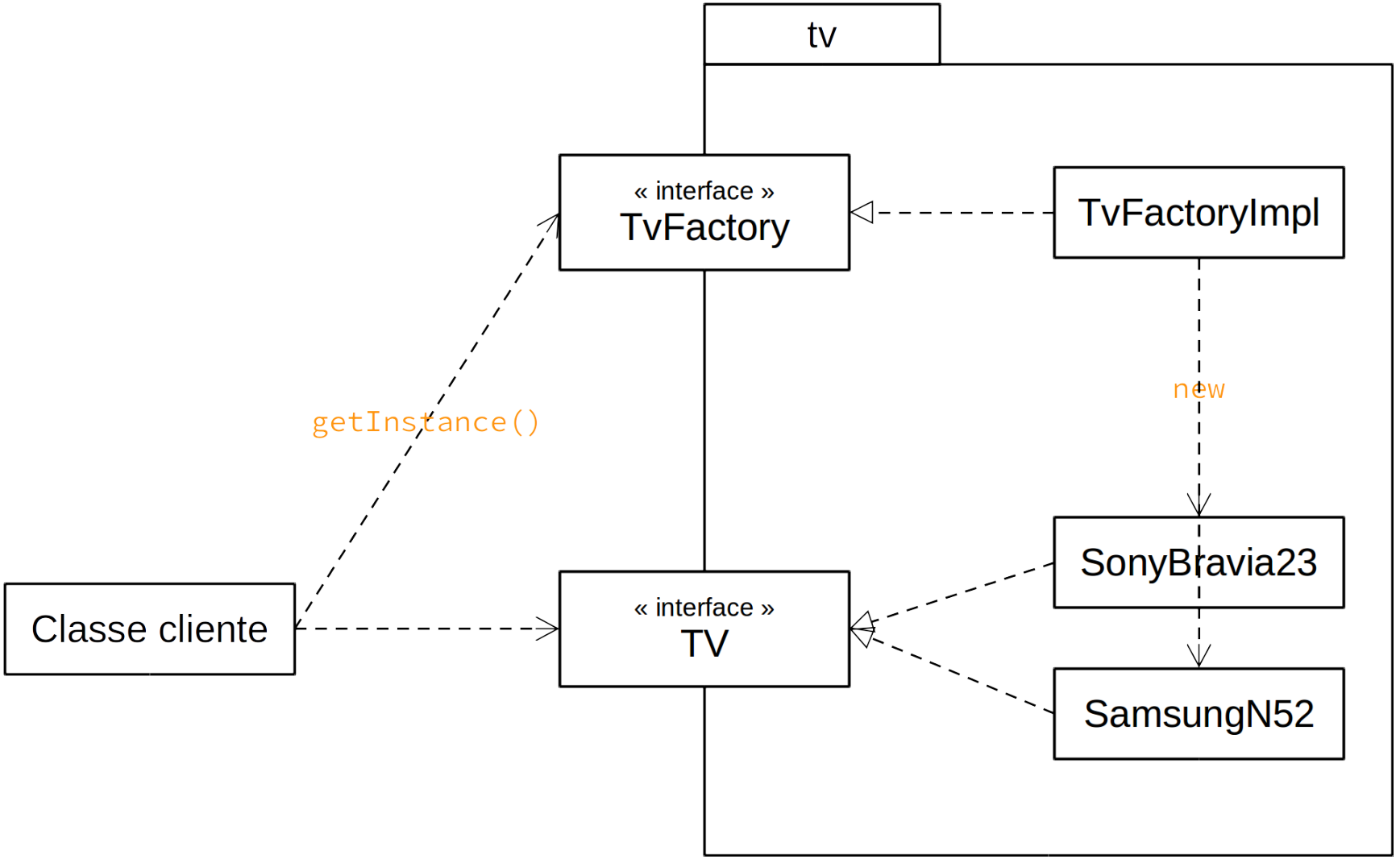
```
MonInterface.INSTANCE.methodeUtile();
```

- Initialisation ?
- Singleton ?

Remarques

- Interface = Factory du Service Object
- Stateless = Immuable = Thread-safe
- Quasi-Singleton

Exemple


tv
« interface »
TvFactory
TvFactoryImpl
new
getInstance()
SonyBravia23
Classe cliente
« interface »
TV
SamsungN52