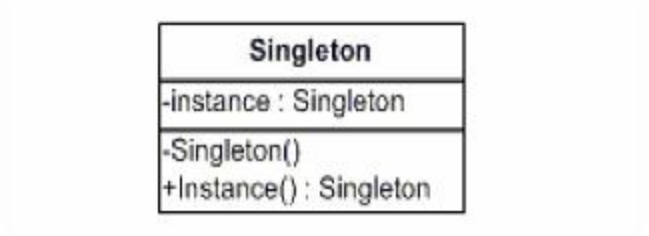
DESIGN PATTERNS

[GoF] Créateurs

- Abstraction du processus de création.
- Encapsulation du choix de création.
- On ne sait pas à l'avance ce qui sera créée ou comment cela sera crée.

[GoF] Singleton

- Avoir une seule et unique instance d'un objet



[GoF] Singleton, exemple

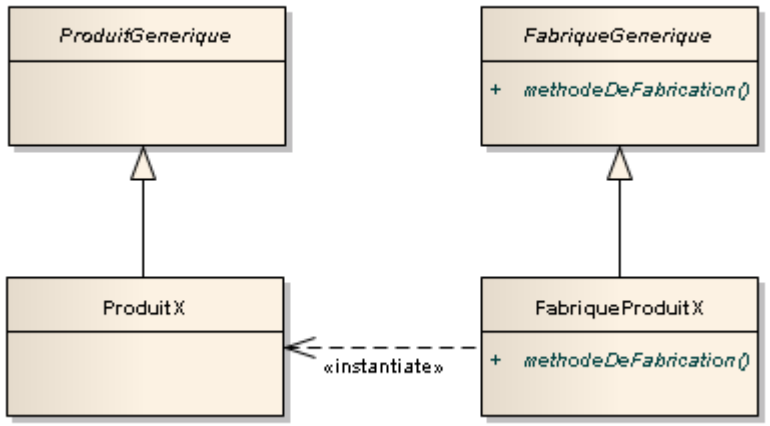
```
class ApplicationConfiguration {
  private static $_instance = false;
  public static function getInstance (){
    if (self::$_instance === false){
      self::$_instance = new ApplicationConfiguration ();
    }
    return self::$_instance;
  }
  private function __construct (){
    //chargement du fichier de configuration
  }
}
```

[GoF] Singleton / Classe statique ?

- Une instance à manipuler.
- Conserve un contexte.
- Peut être passé en paramètre à n'importe quelle méthode.

[GoF] Factory method / Factory / Fabrique

- On ne connait pas encore l'objet qu'il nous faut à l'écriture du code.

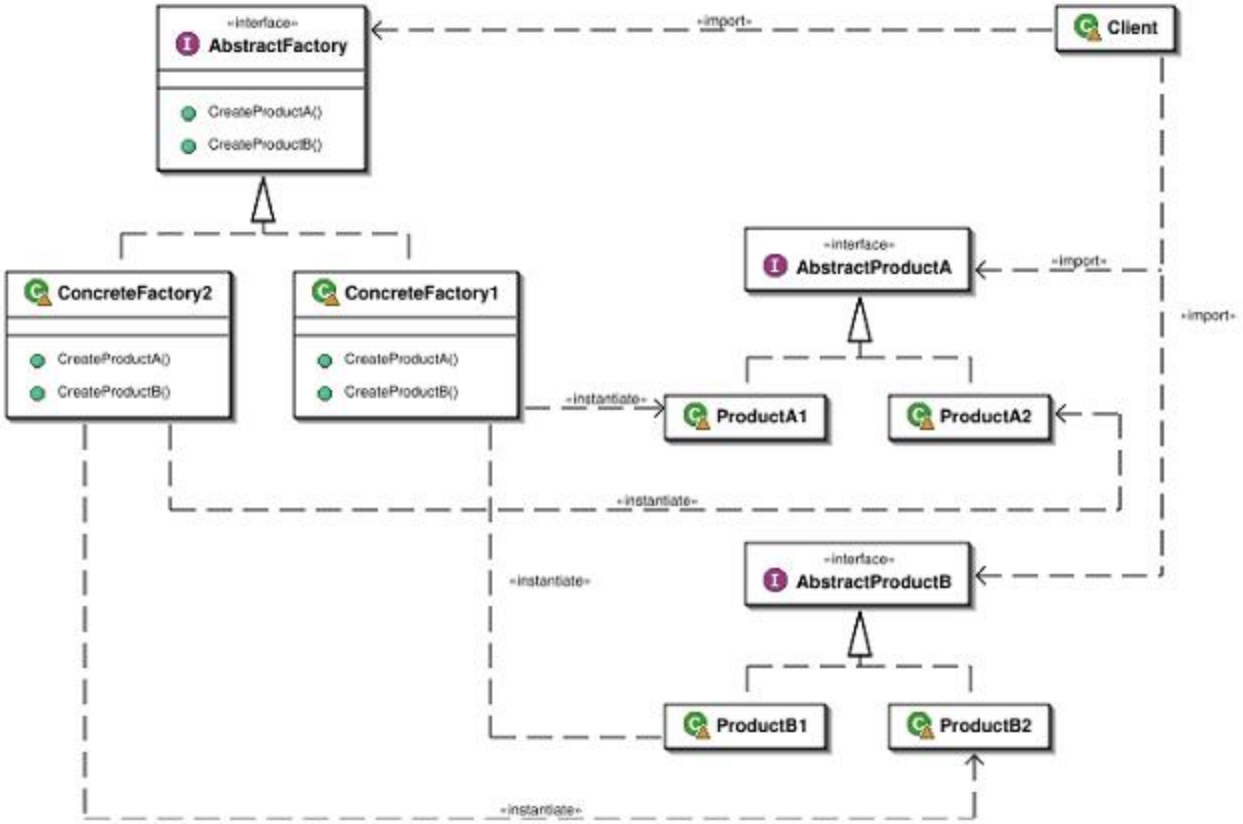


[GoF] Factory, exemple simple

```
require_once ('DBAbstract.class.php');
class DBFactory {
  static function create ($dataSourceId){
    switch (self::_getDriver ($dataSourceId)){
      case self::MySQL :
        require_once ('DBMySQL.class.php');
        return new DBMySQL ($dataSourceId);
      case self::MySQLI :
        require_once ('DBMySQLI.class.php');
        return new DBMySQLI ($dataSourceId);
    }
  }
}
```

[GoF] Abstract Factory / Fabrique abstraite

Création d'une famille d'objet



[GoF] Abstract Factory, exemple

```
abstract class AbstractDocumentFactory {
   abstract public function getDevis ();
   abstract public function getAdhesion ();
}
class PdfDocumentFactory {}
class HTMLDocumentFactory {}

abstract class Devis ();
abstract class Adhesion ();
```

[GoF] Builder

- **Création d'objets complexes**
  à partir d'un objet source

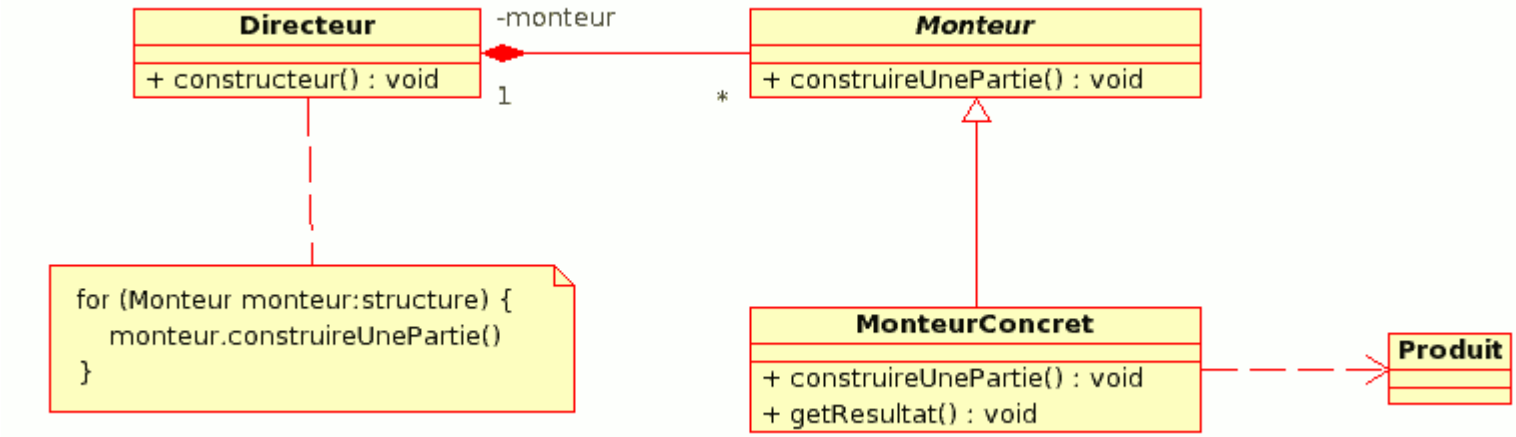
- **But**
  dissocier la construction de l'objet complexe
  de la représentation choisie

  ➔ *garder le même processus si la représentation change*

[GoF] Builder

- **AbstractBuilder (MonteurAbstrait)**
  encapsule le produit construit
- **ConcreteBuilder (MonteurConcret)**
  implémente le monteur abstrait, assemble le produit
- **Directeur**
  utilise le monteur concret
- **Produit**
  l'objet complexe construit

[GoF] Builder


Directeur
+ constructeur() : void
-monteur
1
*
Monteur
+ construireUnePartie() : void
for (Monteur monteur:structure) {
monteur.construireUnePartie()
}
MonteurConcret
+ construireUnePartie() : void
+ getResultat() : void
Produit

[GoF] Builder, exemple

- **Produit : Pizza**
  pate, sauce, garniture (string)
  accesseurs (set)

- **AbstractBuilder : PizzaBuilder**
  creerPizza (monterPate, monterSauce, monterGarniture)

- **PizzaHawaiiBuilder *(sous-classe)***
  pate croisée, sauce douce, jambon ananas
  PizzaPiquanteBuilder *(sous-classe)*
  pate feuilletée, sauce piquante, pepperoni salami

[GoF] Builder, exemple

- **Directeur : Serveur**
  unPizzaBuilder
  setPizzaBuilder, getPizza
  assemblerPizza
  (creerPizza, monterPate, monterSauce, monterGarniture)

- **Main: Client**
  1- creation unServeur, unPizzaPiquanteBuilder
  2- associer unPizzaPiquanteBuilder à unServeur
  3- assembler la pizza

[GoF] Prototype

- **Éviter la création d'une instance longue et complexe**
  cout en temps

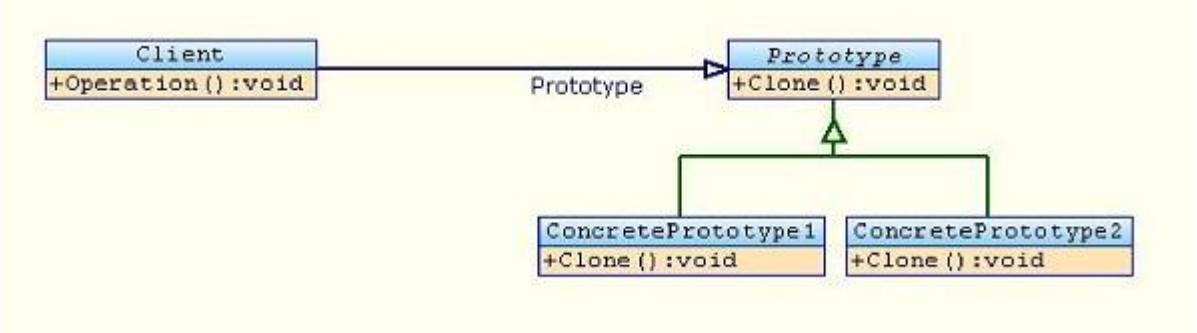
- **Copie (clone) de la première instance plutôt que création (new)**
  classe abstraite avec méthode clone (virtuelle)
  modification de la copie

[GoF] Prototype

- **Prototype (abstraite)**
  modèle pour la création de copie (clone)

- **ConcretePrototype1, ConcretePrototype2**
  attributs supplémentaires
  spécialisent la méthode de clonage

- **Client**
  appelle la méthode de clonage (operation)



[GoF] Prototype

- **Méthode clone() en PHP, java, …**

- **Méthode Clone C#**
  interface Icloneable

[GoF] Prototype, exemple

- **Prototype : Biscuit**
  implémente l'interface ICloneable (MemberwiseClone)

- **Prototype1 : BiscuitLU**
  sous classe

- **Client : MachineABiscuit**
  constructeur : un biscuitLU en parametre
  méthode FaireBiscuit

  Main :
  1- creation unBiscuitLU
  2- creation MachineABiscuit
  3- creation de 24 biscuits LU (boucle)

[GoF] Résumé

- Factory

  Interface de création d'objets

- Abstract Factory

  Fabrique de création pour des familles d'objets

- Singleton

  Instances uniques

- Builder

  Création d'objets complexes

- Prototype

  Création par copie

[GoF] Résumé

- **Singleton**
  instances uniques
- **Factory**
  interface de création d'objets
- **Abstract Factory**
  fabrique de création pour des familles d'objets
- **Builder**
  création d'objets complexes
- **Prototype**
  création par copie