



INF4067

# DEVOIR N°1 PORTANT SUR LE PATRON DE CONSTRUCTION

Etudiant :

NOM	MATRICULE
NOUBISSIE KAMGA WILFRIED	20U2671

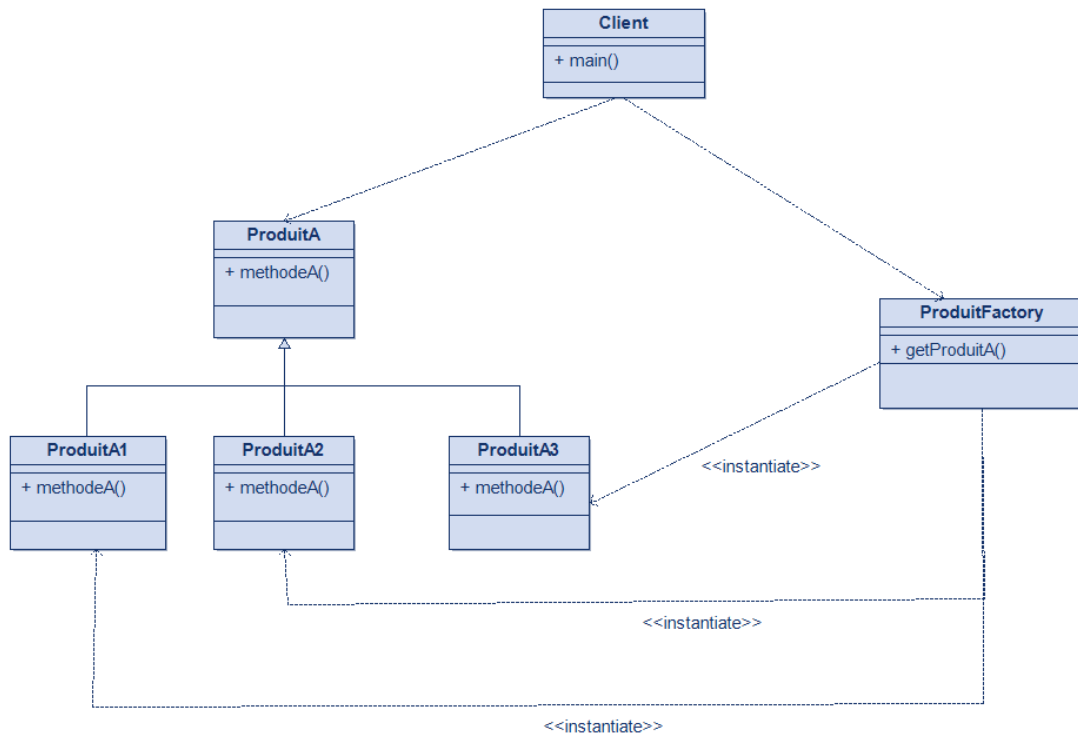
## Table des matières

<b>I.</b>	<b>LE PATRON FACTORY .....</b>	<b>2</b>
1.	Schéma .....	2
a.	Avec une seule fabrique pour tous les type de produit .....	2
b.	Avec une fabrique abstraite et des fabriques concrètes.....	2
2.	Bibliothèque java qui l'implémente .....	3
<b>II.</b>	<b>LE PATRON SINGLETON .....</b>	<b>5</b>
1.	Schéma .....	5
2.	Bibliothèque java qui l'implémente .....	5
<b>III.</b>	<b>LE PATRON BUILDER.....</b>	<b>7</b>
3.	Schéma .....	7
4.	Bibliothèque java qui l'implémente .....	7

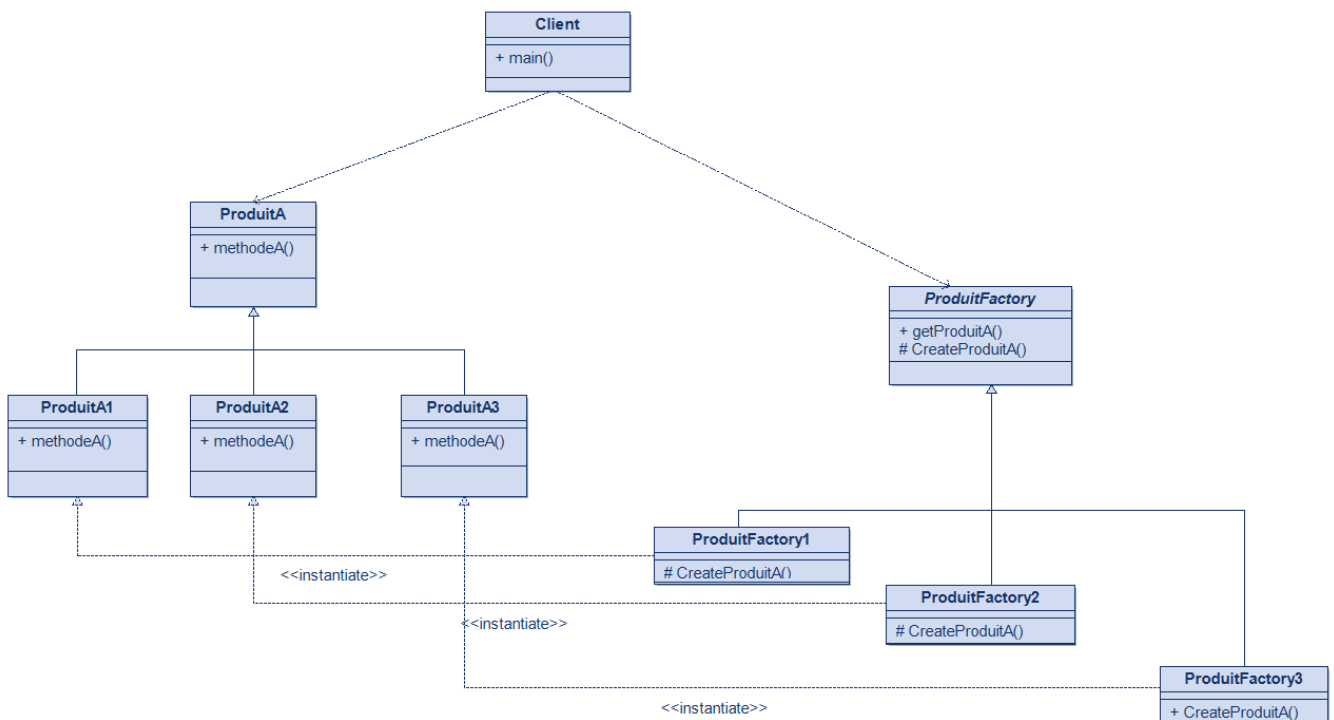
# I. LE PATRON FACTORY

## 1) Schéma

a) Avec une seule fabrique pour tous les types de produit



b) Avec une fabrique abstraite et des fabriques concrètes



## 2) Bibliothèque java qui l'implémente

Une bibliothèque java courante pour l'implémentation du patron Factory est **Spring Framework**, avec l'utilisation des **@Beans** et l'**inversion de contrôle (IoC)**. Ci-dessous, un exemple d'utilisation :

```
import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
```

```
interface Personne {
    void salutation();
}
```

```
class Femme implements Personne {
    public void salutation() {
        System.out.println("Bonjour madame !");
    }
}
```

```
class Homme implements Personne {
    public void salutation() {
        System.out.println("Bonjour monsieur !");
    }
}
```

```

@Configuration
class PersonneFactory {

    @Bean
    public Personne getPersonne(String genre) {
        switch (genre.toLowerCase()) {
            case "femme":
                return new Femme();
            case "homme":
                return new Homme();
            default:
                throw new IllegalArgumentException("Genre inconnu");
        }
    }
}

public class Main {
    public static void main(String[] args) {
        ApplicationContext context = new AnnotationConfigApplicationContext(PersonneFactory.class);

        Personne femme = context.getBean("getPersonne", "femme");
        femme.salutation(); // Resultat: Bonjour madame !

        Personne homme = context.getBean("getPersonne", "homme");
        homme.salutation(); // Resultat: Bonjour monsieur !
    }
}

```

## II. LE PATRON SINGLETON

### 1) Schéma

Singleton
- instance : Singleton
- Singleton() + getInstance()

### 2) Bibliothèque java qui l'implémente

Pour l'implémentation du patron singleton en java, nous pouvons utiliser la bibliothèque **Guava**. Ci-dessous, un exemple d'utilisation :

```
import com.google.common.base.Supplier;
```

```
import com.google.common.base.Suppliers;
```

```
public class Singleton {
```

```
    // Instance unique avec Guava
```

```
    private static final Supplier<Singleton> INSTANCE = Suppliers.memoize(Singleton::new);
```

```
    // Constructeur privé pour empêcher l'instanciation
```

```
    private Singleton() {}
```

```
    // Méthode pour obtenir l'instance unique
```

```
    public static Singleton getInstance() {
```

```
        return INSTANCE.get();
```

```
    }
```

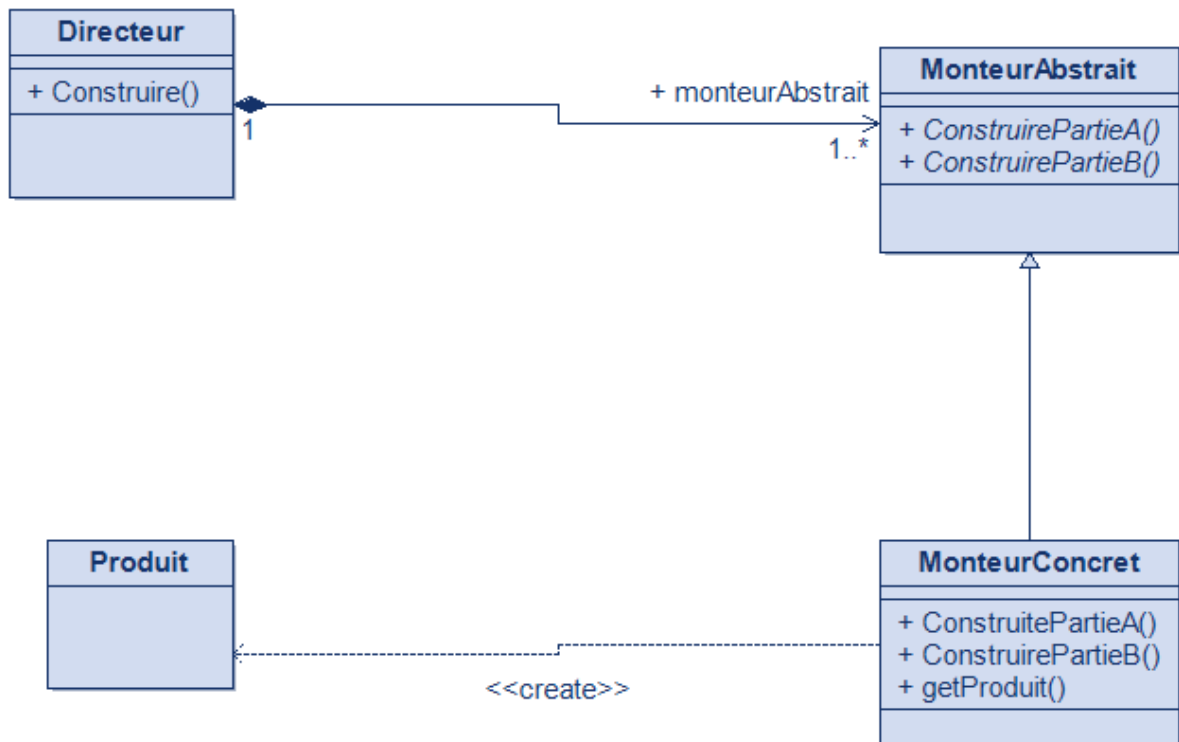
```
// Méthode d'exemple
public void displayMessage() {
    System.out.println("Hello from Guava Singleton!");
}

}

public class Main {
    public static void main(String[] args) {
        Singleton singleton = Singleton.getInstance();
        singleton.displayMessage(); // Affiche : Hello from Guava Singleton!
    }
}
```

### III. LE PATRON BUILDER

#### 1) Schéma



#### 2) Bibliothèque java qui l'implémente

Une bibliothèque très populaire ici pour l'implémentation du patron Builder est la bibliothèque **Lombok**. Ci-dessous, un exemple d'implémentation :

```
import lombok.Builder;
```

```
import lombok.Getter;
```

```
@Getter
```

```
@Builder
```

```
public class Person {
```

```
    private final String nom;
```

```
    private final String matricule;
```



```

        private final int telephone ;
    }

    public class Main {
        public static void main(String[] args) {
            Person person = Person.builder()
                .nom("Noubissie Kamga Wilfried")
                .matricule("20U2671")
                .telephone(690232120)
                .build();

            System.out.println("Nom : " + person.getNom() + " --- " + person.getMatricule());
            System.out.println("Numéro : " + person.getTelephone());
        }
    }

```