



Formation Java 17

Pattern Optional

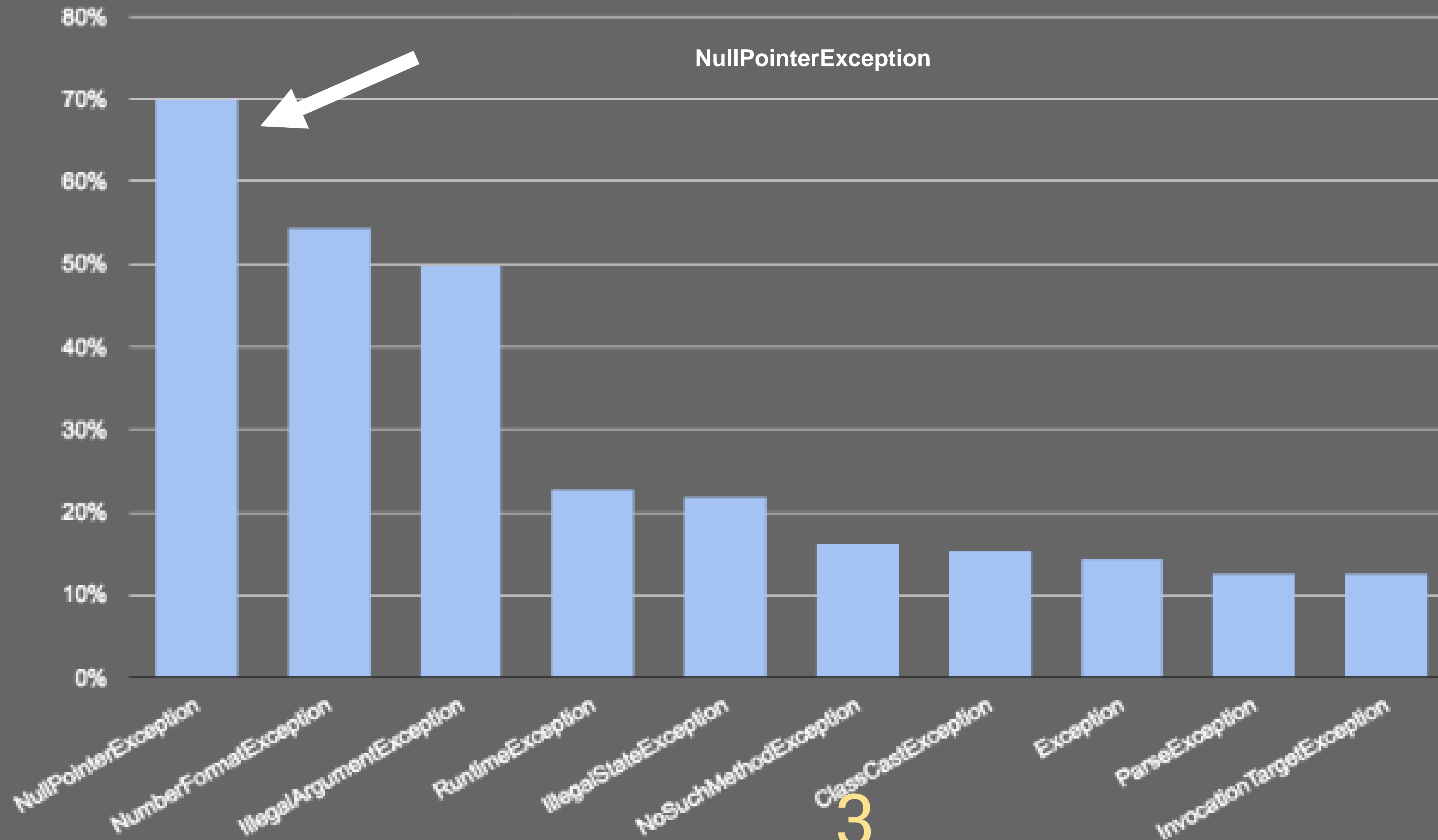
Sommaire

Problématique du NullPointerException

Le type Optional



Top 10 Exception Types by Frequency in 1,000+ Applications



NullPointerException

null ?



```
order.getCustomer().getFirstname().toString();
```

NullPointerException

null ?

```
order.getCustomer().getFirstname().toString();
```

NullPointerException

null ?

```
order.getCustomer().getFirstname().toString();
```

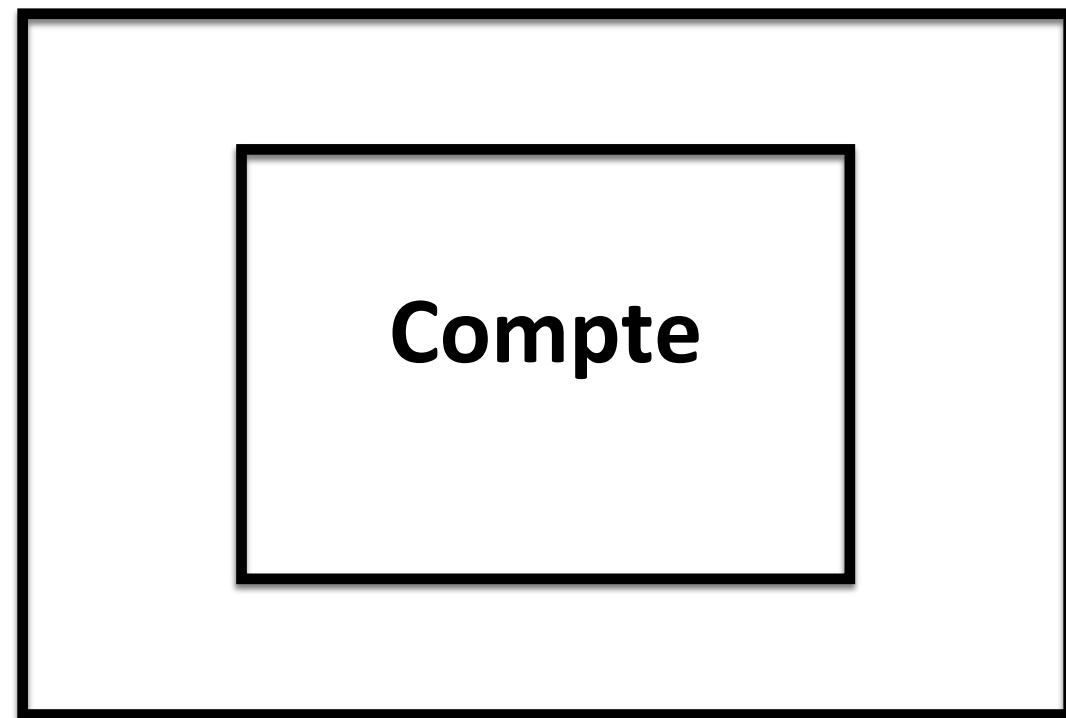
La peur d'être null...

```
if (order!=null) {  
    if (order.getCustomer() != null) {  
        if (order.getCustomer().getFirstname() != null) {  
            order.getCustomer().getFirstname().toString();  
        }  
    }  
}
```

Classe java.util.Optional<T>

L'optional est un conteneur

Optional<Compte>



Optional<Compte>



Même structure qu'il y ait un objet ou non.

Optional vide

```
Optional<Order> optA = Optional.empty();
```

```
Optional<Customer> optB = Optional.empty();
```

Optional non nullable

```
Optional<Customer> opt = Optional.of(findByCode(« ACT »));
```

```
Customer c = opt.get();
```

- La génération de l'optional échoue et génère une exception de type `NullPointerException` si la méthode **findByCode** retourne **null**

Optional nullable

```
Optional<Customer> opt = Optional.ofNullable(findByCode(« ACT »));
```

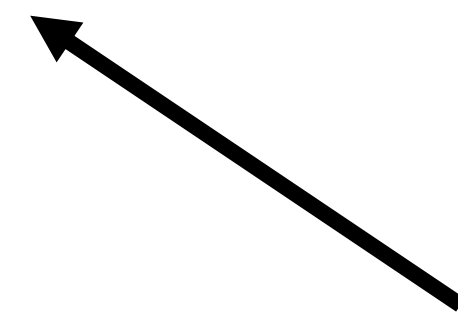
```
if (opt.isPresent()){  
    Customer c = opt.get();  
}
```

- Ne génère **pas** d'exception si la méthode **findByCode** retourne null
- En revanche si la méthode findByCode retourne null, l'utilisation de isPresent pour tester le contenu de l'optional est obligatoire

Optional nullable

```
Optional<Customer> opt = Optional.ofNullable(findByCode(« ACT »));
```

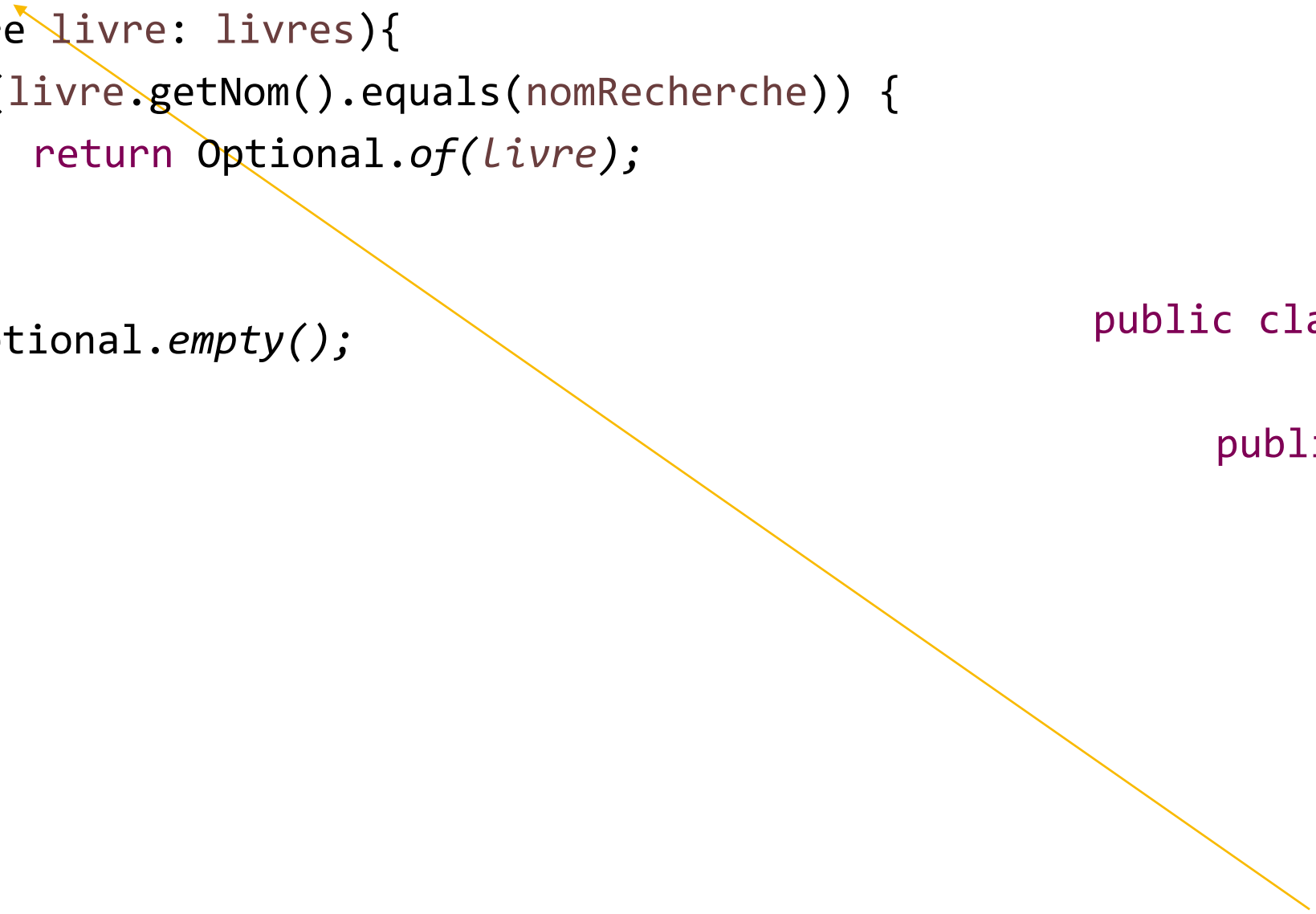
```
Customer c1 = opt.get();
```



NoSuchElementException si aucune valeur

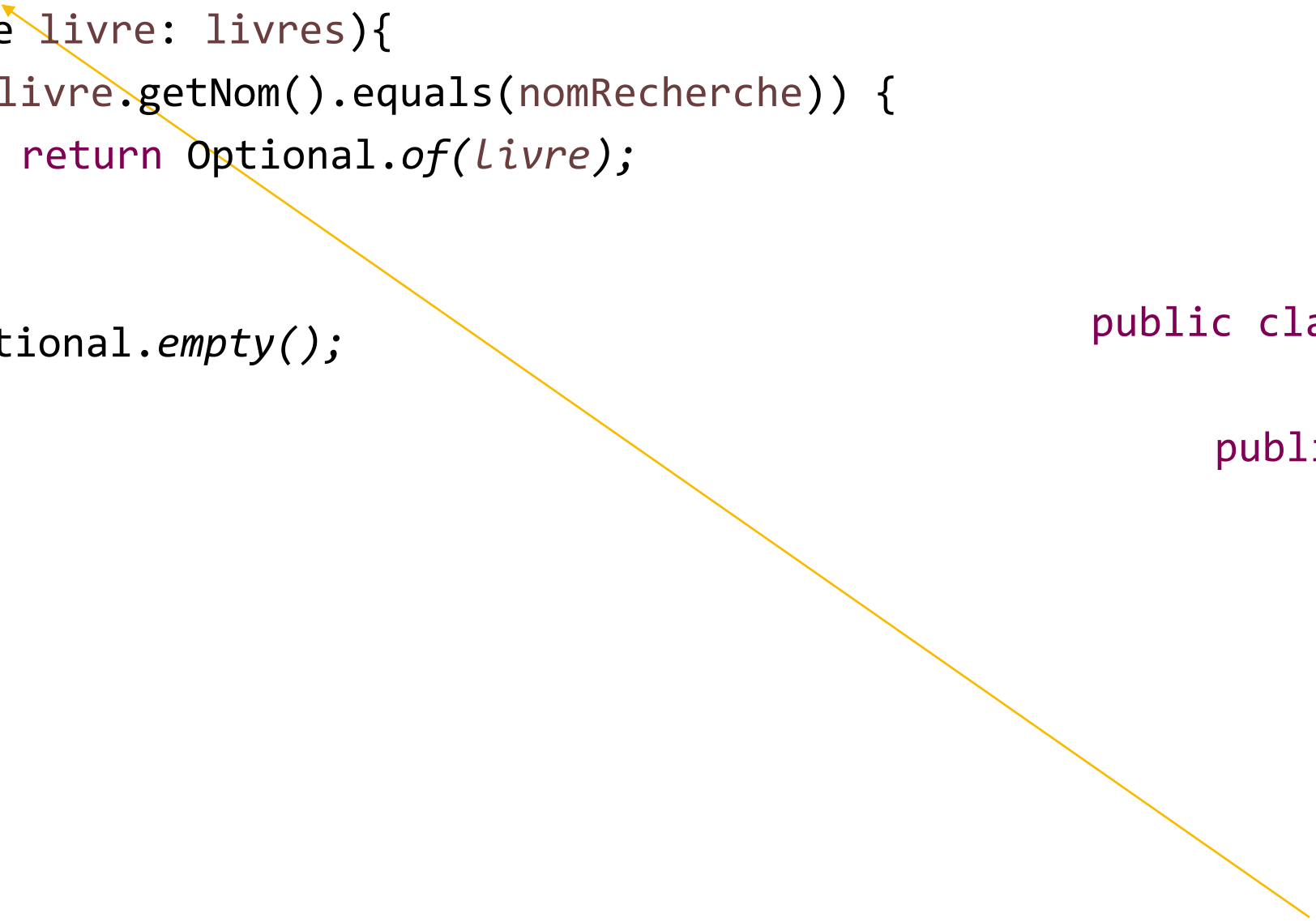
Exemple: Implémentation impérative

```
public class LivreServices {  
  
    public Optional<Livre> getLivre(List<Livre> livres, String nomRecherche){  
        for (Livre livre: livres){  
            if (livre.getNom().equals(nomRecherche)) {  
                return Optional.of(livre);  
            }  
        }  
        return Optional.empty();  
    }  
}  
  
public class TestLivre {  
  
    public static void main(String[] args) {  
        List<Livre> livres = new ArrayList<>();  
        livres.add(new Livre("La programmation en 1h!", 15));  
        livres.add(new Livre("Les 1001 façons d'être lourd", 25));  
        livres.add(new Livre("Le secret de la pêche à la mouche", 14));  
        livres.add(new Livre("Les chiens peuvent ils parler ?", 24));  
  
        LivreServices service = new LivreServices();  
        Optional<Livre> optional = service.getLivre(livres, "La programmation en 1h!");  
        if (optional.isPresent()){  
            Livre livre = optional.get();  
            System.out.println(livre);  
        }  
    }  
}
```



Exemple: Implémentation fonctionnelle avec ifPresent

```
public class LivreServices {  
  
    public Optional<Livre> getLivre(List<Livre> livres, String nomRecherche){  
        for (Livre livre: livres){  
            if (livre.getNom().equals(nomRecherche)) {  
                return Optional.of(livre);  
            }  
        }  
        return Optional.empty();  
    }  
}  
  
public class TestLivre {  
  
    public static void main(String[] args) {  
        List<Livre> livres = new ArrayList<>();  
        livres.add(new Livre("La programmation est un art !", 15));  
        livres.add(new Livre("Les 1001 façons d'être lourd", 25));  
        livres.add(new Livre("Le secret de la pêche à la mouche", 14));  
        livres.add(new Livre("Les chiens peuvent ils parler ?", 24));  
  
        LivreServices service = new LivreServices();  
        service.getLivre(livres, "La programmation est un art !")  
            .ifPresent(livre -> System.out.println(livre));  
    }  
}
```



Optional

Méthode	Description
<code>empty</code>	Retourne une instance vide (Optional)
<code>filter</code>	Retourne un optional valorisé si le prédicat est respecté sinon un Optional vide.
<code>flatMap</code>	Si la valeur est présente, la transformation est appliquée.
<code>get</code>	Retourne la valeur si elle est présente, sinon lance une exception.

Optional

Méthode	Description
ifPresent	Si la valeur est présente, invoque le Consumer en paramètre.
isPresent	Retourne true si la valeur est présente.
map	Si la valeur est présente, la transformation est appliquée.
of	Retourne un optional si la valeur n'est pas null.

Optional

Méthode	Description
orElse	Retourne la valeur si elle est présente, sinon retourne la valeur par défaut fournie
orElseGet	Retourne la valeur si elle est présente, sinon invoque la fonction Supplier fournie.
orElseThrow	Retourne la valeur si elle est présente, sinon lance l'exception fournie.
ofNullable	Retourne un optional.

Travaux Pratiques