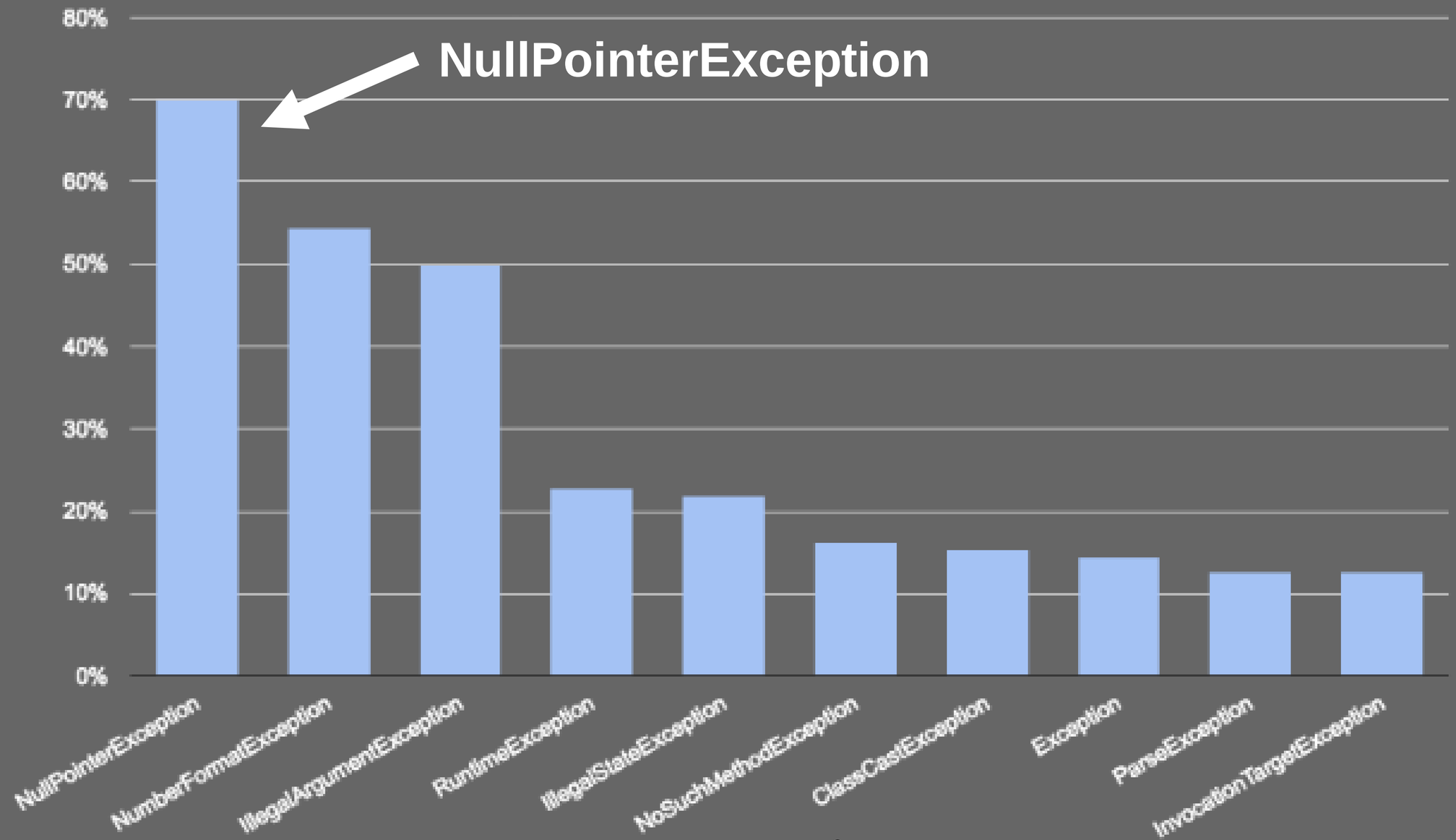
# Formation Java 8

## Pattern Optional

# Sommaire

- Problématique du NullPointerException
- Le type Optional

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






TAKIPI

# NullPointerException

**null ?**

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

# NullPointerException

**null ?**

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

# NullPointerException

**null ?**

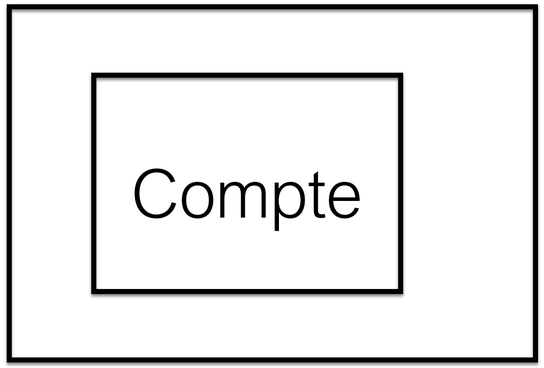
```
order.getCustomer().getFirstname().toString();
```

# 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>

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

- Génère une exception si la méthode retourne null

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

# Optional nullable

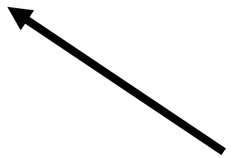
- Ne génère pas d'exception si la méthode retourne null

```
Optional<Customer> opt = Optional.ofNullable(findByCode(« ACT »));
if (opt.isPresent()){
   Customer c = opt.get();
}
```

# Optional nullable

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

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



**NoSuchElementException si aucune valeur**

# Exemple: Méthode qui renvoie un Optional

```java
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();
    }

}
```

```java
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();
        Optional<Livre> optional = service.getLivre(livres, "La programmation est un art !");
        if (optional.isPresent()){
            Livre livre = optional.get();
            System.out.println(livre);
        }

    }

}
```

# Optional

| Méthode | Description |
| --- | --- |
| empty | Retourne une instance vide (Optional) |
| filter | Retourne un optional valorisé si le prédicat est respecté sinon un Optional vide. |
| flatMap | Si la valeur est présente, la transformation est appliquée. |
| get | 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