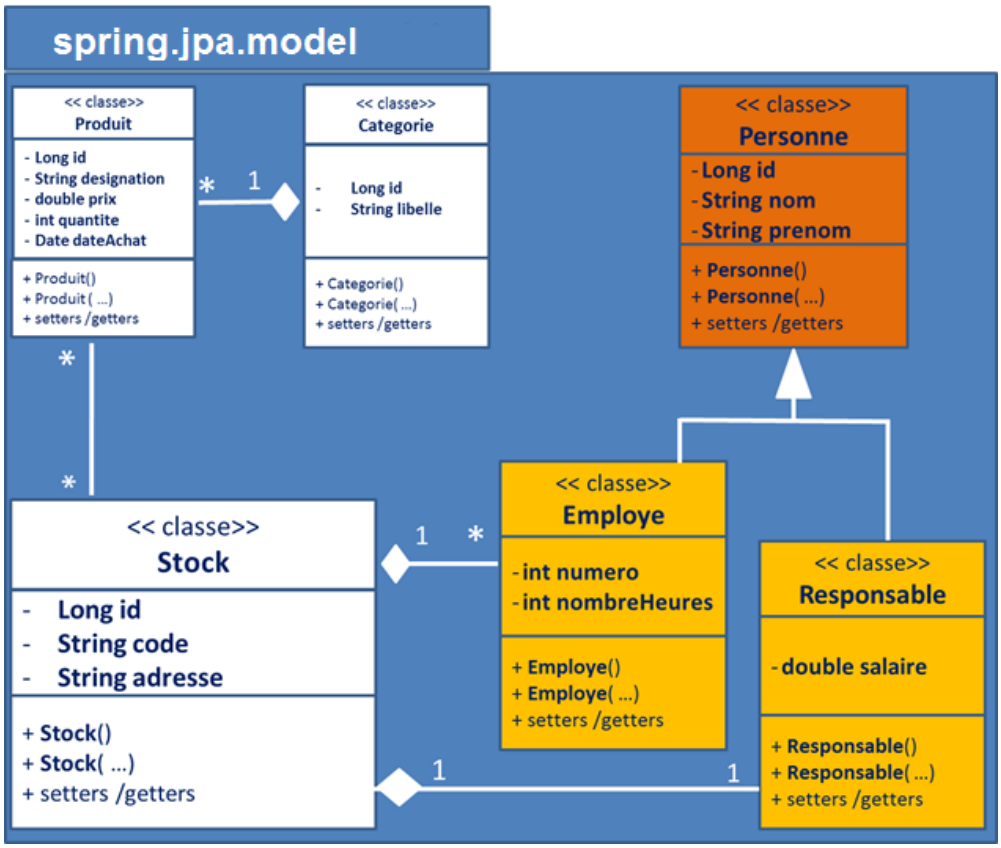
Atelier Spring -03-
JPA –Partie11

Objectifs

- **JPA et Spring Data**
  - Mettre en œuvre d'une relation d'héritage JPA
  - Stratégie : **@Inheritance : Joined Table**
  - Réaliser les requêtes polymorphes

Voici un modèle conceptuel qui modélise une relation d'héritage :

- Une sous-classe « **Responsable** » caractérisée par un **salaire**
- Une sous-classe « **Employe** » caractérisée par deux attributs (**numero** et **nombreHeures**)
- Une classe mère « **Personne** » qui présente les attributs communs (**id**, **nom** et **prenom**)



• La stratégie « @Inheritance : Joined Table»

1. Prendre une copie du projet «**jpa-spring-data-spring-boot-2-relations-3-heritage2**» et le nommer «**jpa-spring-data-spring-boot-2-relations-3-heritage3**»

2. Modifier la classe « **Personne** » (classe mère comme entité) en changeant la stratégie d'héritage au « JOINED ». Cette stratégie consiste à enregistrer les champs de chaque entité dans une table propre à la classe correspondante.

```
……
//pour déclarer une classe mère (entité)
@Entity
@Inheritance(strategy = InheritanceType.JOINED)
@DiscriminatorColumn(name = "TYPE_PERSONNE")
@DiscriminatorValue("Personne")
public class Personne {
        // code de la classe
```

3. Lancer l'exécution du projet en utilisant une autre base de données «**spring-jpa-3**» (créer cette base de données et la référencer dans le fichier « **application.properties** ») et remarquer :
   - ✓ La génération de trois tables de la hiérarchie :« **Personne** », « **Responsable** » et « **Employe** » :

✓ La clé primaire de chacune des tables « **Responsable** » et « **Employe** » est aussi une clé étrangère qui référence la clé primaire de la table « **Personne** ».

✓ La colonne « **type_personne** » permet de distinguer les données de chaque entité.