METHODES DE CONCEPTION

JAVA

UNIVERSITÉ CAEN NORMANDIE

Le but de cette première séance est de se remettre dans le bain de la conception objet, en travaillant sur différentes façons de créer des instances

1. Rappels sur l'utilisation de Netbeans

Netbeans --> file --> new Project --> Java project --> choisir "tp1L3" comme nom de projet.

Un projet contenant un package (boite jaune) du même nom sont créés. Un clic droit puis "new" puis "java class" sur le package permettra de créer de nouvelles classes.

2. Création de la classe Personne et de ses constructeurs

1. Créer la classe Personne, avec les attributs suivants :
   - nom : String
   - prenom : String
   - anneeNaissance : int
   - adresse : String
   - telephoneMobile : String

   Ajouter les getters/setters (menu Source --> insérer du code --> getters et setters)
2. Créer les trois constructeurs suivants :
   - public Personne(String nom, String prenom, int anneeNaissance, String adresse, String telephoneMobile)
   - public Personne(String nom, String prenom, String adresse)
   - public Personne(String nom, String prenom)

   Utiliser this(...) pour éviter la redondance de code
3. Faire en sorte que lorsqu'un attribut String n'est pas passé au constructeur, il soit mis à la valeur null, et que lorsque l'année de naissance n'est pas passée au constructeur, elle soit mise à la valeur -1.

3.Redéfinition de toString()

1. Redéfinir toString() de sorte que l'on obtienne un résultat du type suivant :
   Nom = Dupont
   Prénom = N/A
   Année de naissance = N/A
   Adresse = 3 rue Copernic, 14000 Caen
   Telephone Mobile = 0688773311
2. On pourra utiliser l'opérateur ternaire (booléen ? siVrai : siFaux) afin d'insérer N/A au lieu de null ou -1 pour les différents attributs non renseignés (cf. Prénom et année de naissance dans l'exemple ci-dessus).

4.Des constructeurs plus souples...

La surcharge de constructeurs vue en question 2 devient difficile à gérer dès lors que l'on souhaite que tout paramètre peut devenir optionnel (explosion combinatoire des constructeurs) : cela s'appelle l'anti-pattern constructeurs téléscopiques (on a des constructeurs de toutes les tailles, comme un téléscope que l'on déploie).
Autre problème : le constructeur avec nom+prénom+adresse et celui avec nom+prénom+téléphone ont la même signature, ce qui n'est pas possible.
Nous allons étudier 3 autres façon de procéder :

1. Beans java : créer un constructeur vide qui met toutes les valeurs par défaut comme null ou -1. Après avoir construit un objet, utiliser les setters pour compléter les valeurs d'attributs souhaitées. Problème : impossible de mettre des attributs immuables (final), comme le nom, le prénom et l'année de naissance
2. Factory methods : créer des méthodes statiques de construction, qui renvoient donc une instance de Personne correspondant aux paramètres passés. On a de nouveau le problème « téléscopique », mais plus de problème de signatures identiques car on peut utiliser différents noms pour différents factory methods. Exemple :
   - public Personne createPersonneFromNomPrenom(String nom, String prenom)
   - etc.
3. Mise en place d'un fluent builder :
   - Créer la classe PersonneBuilder, avec les mêmes attributs que la classe Personne
   - Définir son constructeur par défaut pour qu'il mette les valeurs par défaut (les null et -1)
   - Pour chacun des attributs, créer une méthode permettant de spécifier sa valeur, et de renvoyer, plutôt que void, une référence sur this. Exemple :

```
public PersonneBuilder avecNom(String nom) {
  this.nom = nom;
  return this;
}
```

   - Cette astuce permettra de chaîner élégamment les appels, par exemple :

```
new PersonneBuilder()
        .avecNom("Dupont")
        .avecAdresse("3 rue Copernic, 14000 Caen")
        .avecTelephoneMobile("0688773311")
```

- Créer enfin la méthode build() qui créée et renvoie une instance de personne correspondant au paramétrage spécifié.
- Dans le main(), tester la création d'une instance de la façon suivante :

```
Personne p1 = new PersonneBuilder().avecNom("Lemière").avecPrenom("René").build();
```

- Ce pattern, contrairement aux beans, permet de construire des objets avec des attributs immuables (final). Rendre les trois attributs nom, prénom et année de naissance immuables pour s'en assurer (il faut pour cela retirer les setters correspondants).