ITC315 - Informatique 2

TP1 - JAVA : Méthodes de classe et méthodes d'instance

Wahabou Abdou

Wahabou.abdou@u-bourgogne.fr

2019 - 2020

Exercice 1 : Classe `Ville`

La classe `Ville` comporte trois attributs privés : `nom`, `superficie` (de type `double`) et `population`.

1. Cette classe disposera de deux constructeurs. Le premier, constructeur par défaut, affectera la valeur "inconnu" à l'attribut `nom` et 0 aux attributs `superficie` et `population`. Le second aura trois paramètres correspondant aux attributs de la classe `Ville`.

   Indications :
   Un constructeur est une méthode qui porte le même nom que la classe et qui ne retourne aucune valeur (`void`). Il est utilisé pour l'instanciation d'un objet (création un objet à partir d'une classe).
   Vous aurez donc les deux constructeurs suivants (code à compléter) :

   ```
   public Ville() {
           ...
   }

   public Ville(String nom, double superficie, int population) {
           ...
   }
   ```

2. Prévoir des accesseurs et mutateurs pour chacun des attributs.

   Indications :
   Un accesseur (getter) est une méthode qui permet de lire un attribut d'une classe.
   Un mutateur (setter) est une méthode qui permet de modifier un attribut d'une classe.

   ```
   public String getNom() {
           return nom;
   }

   public void setNom(String nom) {
           this.nom = nom;
   }
   ```

   Écrivez les accesseurs et mutateurs des autres attributs.

3. Faire en sorte que l'instruction : `System.out.println(dijon);` où `dijon` est un objet de type `Ville` affiche un résultat semblable à celui-ci : `Dijon, 40.41 kilomètres carrés, pour 375831 habitants`

   Indications :
   Relisez la fin du TD2.

Exercice 2 : Classe `Departement`

Pour ce qui nous concerne, nous définirons un département comme un regroupement de villes. La classe `Departement` comportera donc un tableau (`tabVilles`) d'objets de type `Ville`. La taille de ce tableau est fixée par l'attribut `nbVilles` dont la valeur sera fournie au constructeur. En outre, cette classe disposera des attributs `numero`, `nom` et `nbVillesSaisies` (qui compte le nombre de villes déjà saisies pour le département).

Attention, les villes sont enregistrées dans un **tableau** et non une liste. Les listes seront vues dans un autre TP.

1. Définir un constructeur pour cette classe

> ***Indications :***
> *Inspirez-vous de la question 1 de l'exercice 1*

2. Définir une méthode `ajouterVille(Ville ville)` qui permet d'ajouter une villes au département si le nombre de ville déjà saisies est inférieur au nombre de villes prévu pour ce département. Sinon, un message d'erreur est affiché.

> ***Indications :***
> *Utilisez une variable qui compte le nombre de villes déjà enregistrées. Assurez-vous que ce nombre est inférieur à la taille de votre tableau avant tout ajout de ville.*

3. Faire en sorte que l'instruction : `System.out.println(coteDor);` où `coteDor` est un objet de type `Departement` affiche un résultat semblable à celui-ci :

```
Villes du département Côte d'or(21) :
1. Dijon, 40.41 kilomètres carrés, pour 375831 habitants
2. Quetigny, 8.19 kilomètres carrés, pour 9690 habitants
3. Beaune, 31.3 kilomètres carrés, pour 52741 habitants
```

> ***Indications :***
> *Inspirez-vous de la question 3 de l'exercice 1*

Exercice 3 : Classe `Main`

La classe `Main` contiendra une méthode `main`.

1. Créer cinq objets de type `Ville` :
   - `dijon` : Dijon, 40.41 kilomètres carrés, 375831 habitants
   - `quetigny` : Quetigny, 8.19 kilomètres carrés, 9690 habitants
   - `beaune` : Beaune, 31.3 kilomètres carrés, 52741 habitants
   - `macon` : Mâcon, 27.0 kilomètres carrés, 100172 habitants
   - `chalon` : Chalon-sur-Saône, 15.22 kilomètres carrés, 133557 habitants
2. Créer deux départements :
   - `coteDor` : Côte d'or, numéro 21, trois villes
   - `saoneEtLoire` : Saône-et-Loire, numéro 71, deux villes
3. Ajouter les villes `dijon`, `quetigny` et `beaune` au département `coteDor`
4. Ajouter les villes `macon` et `chalon` au département `saoneEtLoire`
5. Quel est le résultat affiché par les instructions suivantes ?
   - `System.out.println(coteDor);`
   - `System.out.println(saoneEtLoire);`

Exercice 4 : Attributs et méthodes de classe *vs* attributs et méthodes d'instance

1. Dans la classe `Ville`, ajouter le mot `static` lors de la déclaration de l'attribut `nom`. Exécutez à nouveau la classe `Main`. Quelles conclusions peut-on en tirer ?
2. Définir deux méthodes "booléennes" permettant de vérifier si deux villes sont identiques (sur la base de leurs attributs).
   - `estIdentiqueA(Ville ville)`
   - `sontIdentiques(Ville ville1, Ville ville2)`

> ***Indications :***
> *Observez attentivement les noms des villes créées. Que change l'utilisation de* `static` *? Faites valider vos conclusions par l'enseignant.*

3. Laquelle de ces deux méthodes devrait être une méthode de classe (*méthode statique*) ? Pourquoi ?