Programmation java 8

Bangaly Cissé

21 février 2022

Introduction

Ce fichier contient les énoncés pour lesquelles on a fournit une correction.

Ces exercices ont été réaliser dans le module "programmation java 8".

On n'a pas recensé tout les exercices réalisés dans ce module. On a néanmoins pris garde à ce que l'ensemble des exercices choisi soit représentatif des notions abordées.

> **Remarque 0.1.** *La plupart des algorithmes sont commentés. Ils sont cependant fonctionnel. Lors du développement, je commentais en effet les lignes de codes pour lesquelles j'avais trouvé le résultat escompté afin d'éviter de recompiler tout le code à chaque fois.*

Chapitre 1

Java 8 - fonctions essentielles

1.1 Conditions

➽ Correction : "\ProgrammationFonctionnel\src\exercices\Conditions.java"

Exercice 1.1.

1. *Soit une variable, de type entier, affectée dans le programme. Indiquer si le nombre est positif ou négatif.*
2. *Soit la variable « age » qui représente l'âge d'un enfant qui a entre 6 et 10 ans. Cette variable est affectée dans le programme. Ce programme affichera la catégorie de l'enfant :*
   — *Si l'enfant a 6 ans, la catégorie est poussin*
   — *Si l'enfant a 7 ans, la catégorie est benjamin*
   — *Si l'enfant a 8 ans, la catégorie est cadet*
   — *Sinon, l'enfant est de catégorie « adolescent »*
3. *En utilisant le switch, écrire un programme où en fonction du chiffre entre 0 et 5, il affichera en lettre le chiffre. Ex : Si le nombre choisi est 2, afficher « deux ».*
4. *Faire un décompte.*

1.2 Boucles

➽ Correction : "\ProgrammationFonctionnel\src\exercices\Boucles.java"

Exercice 1.2.

1. *Ecrire un programme avec une boucle, qui écrira 10 fois : « Bonjour tout le monde ! »*
2. *Ecrire un programme avec une boucle, qui écrira 10 fois : « Bonjour tout le monde ! Salutation n » + nombre_de_l_iteration*
3. *Avec la boucle "for", calculer la somme des nombres de 0 à <nombre choisi>*
   Puis : Avec la boucle "for", calculer la somme des nombres de 4 à <nombre choisi>

1.3 Classe

➽ Correction : "\ProgrammationOrienteObject\src\animaux"

Exercice 1.3.

1. *Créer la classe Animal avec les attributs[a] et méthodes suivantes : .*
   — *String nom*
   — *boire() => Qui affichera "glou glou"*
   — *manger() => Qui affichera "miam miam"*
2. *Créer la classe Chat, qui étend la classe Animal et qui aura les attributs et les méthodes suivantes :*
   — *boolean rayure => indique si le chat a une fourrure rayée ou non*

— *demanderSortir() => Qui affichera "Je veux sortir"*
— *faireSesGriffres() => Qui affichera "Je fais mes griffes"*

3. *Créer la classe Chien, qui étend la classe Animal et qui aura les attributs et les méthodes suivantes :*
   — *String couleurCollier*
   — *demanderPromener() => Qui affichera "Je veux me promener"*
   — *rattraperLaBalle() => Qui affichera "Je vais chercher la balle et je la ramène"*

a. Mettre la visibilité des attributs de Animal à protected

1.4 Classe abstraite et héritage

➽ Correction : "\ProgrammationOrienteObject\src\vehicules"

Exercice 1.4.

1. *Créer une classe Véhicule avec les informations suivantes :*
   — *nbRoue : entier*
   — *nbPassager : entier*
   — *propriétaire : String*
   Qui possède les méthodes abstraites accelerer() et freiner().
   Ainsi que la méthode présentation() qui affiche : "J'appartiens à <proprietaire>, j'ai <nbRoue> roues et je peux contenir <nbPassager>
2. *Créer une classe Velo qui hérite de la classe Véhicule avec les informations suivantes :*
   — *Velo est un véhicule qui a 2 roues et un seul passager.*
   — *couleurGuidon : String*
   Lors de l'appel de accelerer() => afficher ("Je pédale vite")
   Lors de l'appel de freiner() => afficher ("Je presse le frein")
3. *Créer une classe Moto avec les informations suivantes :*
   — *Moto est un véhicule motorisé qui a 2 roues et 2 passagers.*
   Lors de l'appel de accelerer() => afficher ("J'appuie sur la pédale")
   Lors de l'appel de freiner() => afficher ("J'appuie sur la pédale de frein")
4. *Créer une classe Voiture avec les informations suivantes : - Voiture est un véhicule motorisé qui a 4 roues et 5 passagers.*
   Lors de l'appel de accelerer() => afficher ("J'utilise la pédale d'accelerateur")
   Lors de l'appel de freiner() => afficher ("J'utilise la pédale de frein")
5. *Créer une classe Execution Dans la fonction main :*
   — *Créer au moins un vélo, une moto et une voiture. Les mettre dans un tableau Vehicule*
   — *Dans un foreach :*
      — *presentation*
      — *accelerer*
      — *freiner*

1.5 Classe abstraite et interface

➽ Correction : "\ProgrammationOrienteObject\src\vehicules"

Exercice 1.5.

1. *Créer une classe Véhicule avec les informations suivantes :*
   (a) *nbRoue : entier*
   (b) *nbPassager : entier*
   (c) *propriétaire : String*
   Qui possède les méthodes abstraites accelerer() et freiner().
   Ainsi que la méthode présentation() qui affiche : "J'appartiens à <proprietaire>, j'ai <nbRoue> roues et je peux contenir <nbPassager>
2. *Créer une interface Motorise qui contient les méthodes suivantes :*

- *allumerMoteur()*
- *eteindreMoteur() qui affiche par défaut : "Je coupe le moteur"*

3. *Créer une classe Velo qui hérite de la classe Véhicule avec les informations suivantes :*
   - *Velo est un véhicule qui a 2 roues et un seul passager.*
   - *couleurGuidon : String*

   Lors de l'appel de accelerer() => afficher ("Je pédale vite")

   Lors de l'appel de freiner() => afficher ("Je presse le frein")
4. *Créer une classe VeloElectrique qui hérite de la classe Velo avec les informations suivantes :*
   - *Velo est un véhicule motorise*

   Lors de l'appel de allumerMoteur() => afficher ("Je mets sur ON")
5. *Créer une classe Moto avec les informations suivantes :*
   - *Moto est un véhicule motorisé qui a 2 roues et 2 passagers.*

   Lors de l'appel de accelerer() => afficher ("J'appuie sur la pédale")

   Lors de l'appel de freiner() => afficher ("J'appuie sur la pédale de frein")

   Lors de l'appel de allumerMoteur() => afficher ("Je mets le contact")
6. *Créer une classe Voiture avec les informations suivantes :*
   - *Voiture est un véhicule motorisé qui a 4 roues et 5 passagers.*

   Lors de l'appel de accelerer() => afficher ("J'utilise la pédale d'accelerateur")

   Lors de l'appel de freiner() => afficher ("J'utilise la pédale de frein")

   Lors de l'appel de allumerMoteur() => afficher ("Je mets le contact et j'utilise le starter")
7. *Créer une classe Execution Dans la fonction main :*
   - *Créer un tableau de Motorise, contenant au moins un vélo électrique, une moto et une voiture.*
   - *Dans un foreach :*
     - *allumer le moteur*
     - *accelerer*
     - *freiner*
     - *eteindre le moteur*

1.6 Collections

➽ Correction : "\ProgrammationOrienteObject\src\collections\ConstructionListe.java"

Exercice 1.6.

Soit une liste d'Integer.

1. *Ajouter tous les nombres de 1 jusqu'à 100 dans la liste*
2. *Afficher tous les éléments de la liste*
3. *Retirer tous les multiples de 5 de la liste.*
4. *Afficher tous les éléments de la liste, avec leur index.*

➽ Correction : "\ProgrammationOrienteObject\src\collections\Etudiant.java" + "\ProgrammationOrienteObject\src\collections\TestEtudiant.java"

Exercice 1.7.

1. *Créer une classe Etudiant avec les attributs suivants :*
   - *String nom*
   - *String prenom*
   - *boolean joueurQuidditch*
2. *écrire le constructeur avec tous ces attributs en paramètres.*
3. *écrire les accesseurs et les mutateurs*
4. *écrire la fonction getPrenomNom qui retournera la concaténation du prénom et du nom.*
5. *Créer les étudiants :*
   - *Harry Potter, joueur de quidditch*
   - *Ron Weasley, non joueur de quidditch*
   - *Hermione Granger, non joueuse de quidditch*

— *Drago Malefoy, joueur de quidditch*

6. *Afficher le nom et le prénom de tous les étudiants.*
7. *Afficher le nom et le prénom de tous les joueurs de quidditch.*
8. *Retirer de la liste tous les non joueurs de quidditch.*
9. *Vérifier si la liste des étudiants est non vide.*
10. *Vérifier si la liste des étudiants ne contient plus l'étudiant Ron Weasley, créé précédemment.*

➽ Correction : "\ProgrammationOrienteObject\src\collections\EnsembleString.java"

Exercice 1.8.

1. *Créer un ensemble de String.*
2. *Rajouter les couleurs suivantes - Rouge - Bleu - Vert - Rouge - Rose - Orange*
3. *Afficher tous les éléments du Set*
4. *Constater que l'ensemble contient bien "Bleu".*
5. *Ecrire un foreach où à l'intérieur, on veut supprimer l'élément en cours. Et constater que nous avons une exception. Donc mettre ce code dans un bloc de gestion des exceptions.*
6. *Retirer (toutes) les couleurs : Orange, Rouge, Bleu, Rose, Vert*
7. *Vérifier si l'ensemble est vide.*
8. *Bonus : trouver un moyen de supprimer automatiquement tous les éléments contenant la lettre "e"*

Chapitre 2

Java et frameworks

2.1 J2EE, JSTL

➽ Correction : "\JSTL\src\main"

Exercice 2.1.

Etape 1 *Créez la classe Abonne de type model (avec des propriétés privates, les getters/setters, potentiellement un constructeur) avec les propriétés suivantes :*

- *String prenom*
- *String nom*
- *Integer typeAbonnement (valeurs possibles :*
  - *1 : abonnement classique*
  - *2 : abonnement duo*
  - *3 : abonnement premium*
  - *0, null ou autre : non abonné)*
- *String adresse*

Dans une servlet, instancier un objet Abonne nommé abonne. Et l'enregistrer en tant qu'attribut. Dans la JSP : afficher les valeurs brutes de l'objet Abonne.

Exemple

- *Prénom : Jacques*
- *Nom : Martin*
- *Type Abonnement : 2*
- *Adresse : 56 rue de la Boustifaille 75012 PARIS*

Etape 2 *Dans la JSP : au lieu d'afficher la valeur brute de typeAbonnement, afficher :*

Abonnement Classique si typeAbonnement vaut 1

Abonnement Duo si typeAbonnement vaut 2

Abonnement Premium si typeAbonnement vaut 3

Aucun Abonnement sinon

Etape 3 *Créez la classe Film de type model (avec des propriétés privates, les getters/setters, potentiellement un constructeur) avec les propriétés suivantes :*

- *String titre*
- *String realisateur*
- *Integer duree (durée en minutes)*
- *Boolean estVu (indique s'il a été vu ou pas)*

Dans la même servlet, instancier un objet Film nommé film. Et l'enregistrer en tant qu'attribut.

Dans la même JSP : afficher les valeurs brutes de cette instance

Etape 4 *Rajouter dans la classe Abonne une propriété (au choix) :*

List<Film> films ou Film[] films

Dans la servlet, rajouter une liste ou un tableau à votre variable de type Abonne.

Dans la JSP, rajouter un tableau HTML avec tous les films.

Etape 5 *Dans la JSP, rajouter une liste affichant les titres des films que l'abonné n'a pas vu.*

Etape 6 *Dans la JSP, rajouter un tableau n'affichant que les films dont la durée est supérieure à 120 (pensez à adapter vos données pour voir une différence)*

Etape 7 *Dans la classe Abonne, rajouter la propriété Abonne abonneDuo (correspond à la personne avec qui il partage l'abonnement) (Note : Oui, une classe peut posséder une propriété qui est de type elle-même)*

Dans la servlet :

Créer une nouvelle instance de Abonne, nommé abonneDuo, ne renseignez que le nom, le prenom et l'adresse. rajouter abonneDuo dans votre variable abonne. Et changer le type d'abonnement à 2 (abonnement duo) Dans la JSP,

Si le type d'abonnement est 2, afficher le nom, le prénom et l'adresse de abonneDuo