Cahier exercices Programmation Orientée Objet (P.O.O)

Fascicule 1 –

Réalisé par Mr Jacquot David

Fait le lundi 14 octobre 2019

Table des matières

Exercice 01 — Les structures de base

Objectif

- Manipuler les structures de base (les conditions et les boucles...).

Énoncé

1. Ecrire un programme java qui lit trois notes à partir du clavier, calcule et affiche la moyenne de ces notes. En fin, il affiche la mention correspondante :
   - "Bien" Si la moyenne est >12.
   - "Passable" Si la moyenne est comprise entre 10 et 12.
   - "Non admis" Si la moyenne est <10.
2. Ecrire un programme qui détermine si un nombre entier est pair ou impair.
3. Écrivez un programme qui demande à l'utilisateur son sexe (M et F) et son Nom et affiche un message de bienvenue approprié.

Vos notes

Exercice 02 — Les étudiants

Objectif

- Créer une classe et utiliser ses méthodes

Énoncé

Partie 1 :

Soit un ensemble d'étudiants. Ils sont tous caractérisés par les informations suivantes :

- Un nom
- 4 notes
- Moyenne
- Résultat (qui vaut "admis" ou "non admis")

Définir la classe Etudiant correspondante en la structurant à l'aide des attributs et en la dotant des méthodes nécessaires pour construire les objets "étudiant" et leur donner les moyens de calculer leurs moyennes et leurs résultats.

Partie 2 :

Les attributs moyenne et resultat sont-ils nécessaires ? justifiez votre réponse.
Faire la modification de la classe et du programme précédent si besoin.

Vos notes

Exercice 03 — Tableau de nombres

Objectif

- Créer une classe et utiliser ses méthodes

Énoncé

Définir une classe TableauNombres de façon à étudier des séries de 10 nombres positifs.
La classe doit être dotée d'un constructeur ayant la capacité d'allouer la place pour un tableau de 10 nombres et de permettre à l'utilisateur de donner les 10 valeurs numériques qui l'intéressent.
Elle doit aussi être dotée des méthodes nécessaires pour que chaque objet puisse calculer la moyenne des nombres et la valeur du plus grand d'entre eux.
Elle doit être aussi doter de capacités de traitements supplémentaire. Par exemple :

- Calculer min, la valeur du nombre le plus petit.
- Afficher tous les nombres supérieurs à la moyenne.
- Calculer la valeur de l'écart le plus grand entre le deux nombres placés côte à côte dans le tableau.

Vos notes

Exercice 04 — Les chaines de caractères

Objectif

- Utilisation des classes String et StringBuffer

Énoncé

Définir la classe Chaine1 en la dotant de la capacité d'extraire toutes les voyelles puis toutes les consonnes d'une chaîne de caractères fournie par l'utilisateur. Vous devrez utiliser les classes String et StringBuffer.

Définir la classe Chaine2 permettant de calculer le nombre de voyelles et le nombre de mots d'une chaine de caractères fournie par l'utilisateur.

Vos notes

Exercice 05 — Le jeu de dés

Objectif

- Utilisation du random

Énoncé

Ecrire un programme qui simule le lancer de 3 dés, qui affiche "Vous avez GAGNE" lorsqu'au moins 2 dés ont donné le même résultat et "Vous avez PERDU" dans le cas contraire.

Vos notes

Exercice 06 L'encapsulation

Objectif

- Utilisation des classes

Énoncé

Partie 1
Définir une classe Administration et une classe Personne avec les propriétés nom et age, toutes deux private.
Doter la classe Administration d'une méthode qui donne les moyens à un objet de cette classe de comparer l'âge de 2 personnes en respectant l'encapsulation des attritbus nom et age.
Doter aussi la classe Personne d'une méthode qui permet à une personne de comparer son âge à celui d'une autre personne.
Quelle est la différence entre les 2 méthodes de comparaison des âges ?

Partie 2
Modifier le programme précédent en déclarant protected les attributs de la classe Personne

Partie 3
Soit la classe Entreprise et la classe Personne structurée avec les propriétés nom Age et salaire.
Ces 2 variables doivent être déclarées private de façon à encapsuler leurs valeurs.
Doter la classe Entreprise d'une méthode qui permet d'augmenter une personne selon un certain coefficient lorsque son salaire est inférieur à 1500 euros.

Vos notes

Exercice 07 — Mon entreprise

Objectif

- Utilisation des classes

Énoncé

Vos notes

Exercice 08 — Les tableaux

Objectif

- Utilisation des classes

Énoncé

Partie 1 (59)

Ecrire un programme qui crée un tableau comportant les valeurs des carrés des n premiers nombre impairs, la valeur "n" étant lue au clavier et qui en affiche les valeurs sous la forme suivante :

Combien de valeur ? 5
1 a pour carré 1
3 a pour carré 9
4 a pour carré 25
7 a pour carré 49
9 a pour carré 81

Partie 2 (65)

Ecrire une classe utilitaire UtilTab disposant des méthodes statiques suivantes :

- somme qui fournit la somme des valeurs d'un tableau de reels (double) de taille quelconque
- incre qui incrémente d'une valeur donnée toutes les valeurs d'un tableau de réels (double).

Ecrire un petit programme d'essai. Pour faciliter les choses, on pourra également doter la classe UtilTab d'une méthode d'affichage des valeurs d'un tableau de réels.

Vos notes

Exercice 9 L'encapsulation - Ma classe voiture

Objectif

- Créer une classe.
- Définir des attributs.
- Implémenter une méthode.

Énoncé

Il est possible pour une voiture de saisir son matricule, son modèle, l'année de son modèle, son prix et sa couleur.

- Lors de la création d'une voiture, son matricule est incrémenté selon le nombre de véhicules créés.
- Créer 3 méthodes :
  - afficherCouleur(): Cette méthode doit afficher la couleur du véhicule
  - démarrerVehicule() : cette méthode doit démarrer le véhicule
  - arreterVehicule(): Cette méthode doit arrêter le véhicule
- Créer une méthode afficher () qui retourne une chaîne de caractères contentant les valeurs du matricule, du modèle, de l'année du modèle, la couleur et du prix.

Travail à faire

Créer un programme permettant la saisie des informations de 5 voitures en utilisant la classe Voiture. Faire démarrer et arrêter les voitures.

Vos notes

Exercice 10 — Ma moto, mon camion

Objectif

- Créer une classe.
- Définir des attributs.
- Implémenter une méthode.

Énoncé

Créer les classes moto et camion comme celle de la classe voiture en personnalisant les messages d'affichage.

Travail à faire

Créer un programme permettant la saisie des informations de 2 voitures en utilisant la classe Voiture, 4 motos (classe Moto) et 1 camion (classe camion).
Faire démarrer tous les véhicules.

Vos notes

Exercice 11 — Ma classe Article

Objectif

- Définir les propriétés et méthodes d'une classe
- Définir des propriétés statiques
- Définir des constructeurs
- Créer une instance de classe
- Accéder par les accesseurs aux propriétés en lecture et écriture d'un objet
- Appliquer des méthodes

Énoncé

1. Créer la classe Article caractérisée par les attributs : Référence, Désignation, PrixHT, TauxTVA. Ces attributs doivent seulement être accessibles par le biais des accesseurs (get / set) en lecture/écriture mis en œuvre par les propriétés.
2. Ajouter les constructeurs suivants :
   - Un constructeur par défaut
   - Un constructeur initialisant tous les attributs.
   - Un Constructeur qui permet de renseigner la référence et la désignation lors de l'instanciation
   - Un constructeur de recopie
3. Implémentez la méthode CalculerPrixTTC(). Cette méthode doit calculer le prix TTC d'un article qui équivaut à : PrixHT + (PrixHT*TauxTVA/100) et retournera la valeur calculée.
4. Ajouter la méthode AfficherArticle() qui affiche les informations de l'article.
5. Le taux de TVA est en fait commun à tous les articles.

Travail à faire

Créer un programme de test où il faut créer des objets (en utilisant les différents constructeurs) et leur calculer le prix TTC.

Vos notes

Exercice 12 — Mon Livre

Objectif

- Définir les propriétés et méthodes d'une classe
- Définir des propriétés statiques
- Définir des constructeurs
- Créer une instance de classe
- Accéder par les accesseurs aux propriétés en lecture et écriture d'un objet
- Appliquer des méthodes

Énoncé

1. Définir une classe Livre avec les attributs suivants : Id, Titre, nom de l'auteur, Prix.
2. Définir les accesseurs aux différents attributs de la classe.
3. Définir un constructeur permettant d'initialiser les attributs d'un objet livre par des valeurs saisies par l'utilisateur. Sachant que Id doit être auto-incrément.
4. Définir la méthode toString ( ) permettant d'afficher les informations du livre en cours.
5. Écrire un programme testant la classe Livre.

Travail à faire

Créer un programme de test où il faut créer des livres. Donner le nombre de livres créés et afficher la liste.

Vos notes

Exercice 13 — Ma bibliothèque

Objectif

Définir les propriétés et méthodes d'une classe
Définir des propriétés statiques
Définir des constructeurs
Créer une instance de classe
Accéder par les accesseurs aux propriétés en lecture et écriture d'un objet
Appliquer des méthodes

Énoncé

1. Définir une classe Bibliothèque avec les attributs suivants : id, nom, nombreDeLivreMax, liste de livre, adresse, ville
2. Le nombreDeLivreMax est commun à toutes les bibliothèques. Et il n'est pas possible de modifier cet attribut.
3. Définir les accesseurs aux différents attributs de la classe.
4. Définir la méthode toString ( ) permettant d'afficher les informations de la bibliothèque.
5. Écrire un programme testant la classe Bibliothèque.

Travail à faire

Créer un programme de test où il faut créer une bibliothèque avec ses livres.

Vos notes

Exercice 14 L'héritage – Personne

Objectif

- Implémentation correcte de l'évolution des objets dans le temps (héritage).
- Redéfinition des méthodes.
- Appel aux méthodes de la classe mère avec super.
- Appel aux constructeurs de la classe mère.

Description

- ✓ La classe Etudiant hérite de la classe Personne.
- ✓ La classe Professeur hérite de la classe Employe et la classe Employe hérite de la classe Personne.
- ✓ Un Etudiant est une Personne.
- ✓ Un Professeur est un Employe et un Employe est une Personne.

Liste des attributs :

Personne : nom, prénom, adresse
Professeur : matiere
Employe : lieuDeTravail
Etudiant: ecole

Liste des méthodes

Personne : toString()
Professeur : attribuerNote(), toString()
Employe : toString()
Etudiant : suivreCours() et passeExamen(), toString()

Travail à faire

1. Créer les classes
   - Chaque classe doit contenir un constructeur d'initialisation.
   - Chaque classe doit redéfinir la méthode toString()

     Exemple : " l'etudiant " + nom + autres attributs
     ou " l'employe "+ nom + autres attributs

.

2. Développer un programme dans lequel on demande de créer :
   - deux étudiants
   - deux employés
   - deux professeurs
3. Afficher les informations de chaque personne.

Exercice 15 L'héritage – figure

Objectifs :

- Définir les propriétés et méthodes d'une classe.
- Définir des constructeurs.
- Créer une instance de classe.
- Accéder par les accesseurs aux propriétés en lecture et écriture d'un objet.
- Appliquer des méthodes.
- Utiliser l'héritage
- Définir l'abstraction

Travail à faire :

1. Définir une classe Rectangle ayant les attributs suivants : Longueur et Largeur.
2. Ajouter un constructeur d'initialisation.
3. Définir les accesseurs aux attributs de la classe.
4. Ajouter les méthodes suivantes :
    - périmètre ( ) : retourne le périmètre du rectangle.
    - aire ( ) : retourne l'aire du rectangle.
    - isCarre ( ) : vérifie si le rectangle est un carré.
    - toString ( ) : expose les caractéristiques d'un rectangle comme suit :
      Longueur : […] - Largeur : […] - Périmètre : […] - Aire : […] - C'est un carré / Ce n'est pas un carré
5. Définir une classe Figure. Cette classe doit être abstraite.

Vos notes

Exercice 16 — Mon parc de véhicules

Objectifs :

- Définir les objets et les classes en utilisant l'héritage, l'encapsulation, l'abstraction,

Travail à faire :

Créer un parc de véhicules en utilisant les notions du langage objet. Ce parc doit contenir des voitures, des camions, des motos, des tracteurs. Ce parc de véhicules doit être aussi capable de nous fournir le nombre véhicule pour chaque type de véhicule ainsi que le total de véhicule.

Vos notes

Exercice 17 Bâtiment

Objectifs :

- Définir les objets et les classes en utilisant l'héritage, l'encapsulation

Enoncé :

1. Ecrire une classe Batiment avec comme attribut l'adresse du bâtiment avec les constructeurs suivants : Batiment() et Batiment(adresse).
2. Ecrire une classe Maison héritant de Batiment avec comme attribut le nombre de pièces de la maison avec les constructeurs suivants : Maison() et Maison(adresse, nbPieces).
3. Ecrire une classe Immeuble héritant de Bâtiment avec comme attribut le nombre d'appartements de l'immeuble avec les constructeurs suivants : Immeuble() et Immeuble(adresse, nbAppart).
4. Chaque classe contenir les accesseurs pour les différents attributs ainsi qu'une méthode **toString**() donnant une représentation de l'objet.
5. Ecrire aussi une classe TestBatiment afin de tester les classes.

Vos notes

Exercice 18 Surfaces

Objectifs :

- Définir les objets et les classes en utilisant l'héritage, l'encapsulation, le polymorphisme

Enoncé :

Calculez la surface d'un terrain composé de 3 rectangles.
Pour cela il faut créer une classe `Terrain` qui doit encapsuler 3 instances de la classe `Rectangle`.

Il doit être aussi possible d'ajouter d'autre forme de terrains, comme des triangles et des cercles.
Créer une classe `Forme` abstraite qui sera commune aux classes `Rectangle`, `Triangle` et `Cercle`.

Remplacez donc les variables d'instance par un tableau de `Forme` (le constructeur de la classe `Terrain` donnera à ce tableau une taille maximale de votre choix). Ajoutez une nouvelle méthode `ajouterForme` à `Terrain` afin de permettre à l'utilisateur d'étendre son terrain.
N'oubliez pas de modifier `calculerSurfaceTotale` pour prendre en compte vos modifications.

Vos notes

Exercice 19 Mon calcul de salaires

Objectifs

Concevoir une hiérarchie de classes utilisant la notion d'interface. Il vous servira également de révision pour les notions d'héritage, de classes abstraites et de polymorphisme

Enoncé

Le directeur d'une entreprise de produits chimiques souhaite gérer les salaires et primes de ses employés au moyen d'un programme.

Un employé est caractérisé par son nom, son prénom, son âge et sa date d'entrée en service dans l'entreprise.

Dans un fichier Salaires.java, codez une classe abstraite Employe dotée des attributs nécessaires, d'une méthode abstraite calculerSalaire (ce calcul dépendra en effet du type de l'employé) et d'une méthode toString retournant une chaine de caractère obtenue en concaténant la chaine de caractères "L'employé " avec le prénom et le nom.

Dotez également votre classe d'un constructeur prenant en paramètre l'ensemble des attributs nécessaires.

Calcul du salaire

Le calcul du salaire mensuel dépend du type de l'employé. On distingue les types d'employés suivants :

- Ceux affectés à la Vente. Leur salaire mensuel est le 20 % du chiffre d'affaire qu'ils réalisent mensuellement, plus 400 Euros.
- Ceux affectés à la Représentation. Leur salaire mensuel est également le 20 % du chiffre d'affaire qu'ils réalisent mensuellement, plus 800 Euros.
- Ceux affectés à la Production. Leur salaire vaut le nombre d'unités produites mensuellement multipliées par 5.
- Ceux affectés à la Manutention. Leur salaire vaut leur nombre d'heures de travail mensuel multipliées par 65 Euros.

Codez dans votre fichier Salaires.java une hiérarchie de classes pour les employés en respectant les conditions suivantes :

- La super-classe de la hiérarchie doit être la classe Employe.
- Les nouvelles classes doivent contenir les attributs qui leur sont spécifiques ainsi que le codage approprié des méthodes calculerSalaire et getNom, en changeant le mot "employé" par la catégorie correspondante.
- Chaque sous classe est dotée de constructeur prenant en argument l'ensemble des attributs nécessaires.

N'hésitez pas à introduire des classes intermédiaires pour éviter au maximum les redondances d'attributs et de méthodes dans les sous-classes

Employés à risques

Certains employés des secteurs production et manutention sont appelés à fabriquer et manipuler des produits dangereux.

Après plusieurs négociations syndicales, ces derniers parviennent à obtenir une prime de risque mensuelle.

Complétez votre programme Salaires.java en introduisant deux nouvelles sous-classes d'employés. Ces sous-classes désigneront les employés des secteurs production et manutention travaillant avec des produits dangereux.

Ajouter également à votre programme une interface pour les employés à risque permettant de leur associer une prime mensuelle fixe de 200

Collection d'employés

Satisfait de la hiérarchie proposée, notre directeur souhaite maintenant l'exploiter pour afficher le salaire de tous ses employés ainsi que le salaire moyen.

Ajoutez une classe Personnel contenant une "collection" d'employés. Il s'agira d'une collection polymorphique d'Employe - regardez le cours si vous ne voyez pas de quoi il s'agit.

Définissez ensuite les méthodes suivantes à la classe Personnel :

- void ajouterEmploye(Employe) qui ajoute un employé à la collection.
- void afficherSalaires() qui affiche le salaire de chacun des employés de la collection.
- double salaireMoyen() qui affiche le salaire moyen des employés de la collection.

```
class Salaires {
    public static void main(String[] args) {
        Personnel p = new Personnel();
        p.ajouterEmploye(new Vendeur("Alex", "Terrieur", 45, "1995", 30000));
        p.ajouterEmploye(new Informaticien("Alain", "Terrieur", 25, "2001", 20000));
        p.ajouterEmploye(new Technicien("Yves", "Bosseur", 28, "1998", 1000));
        p.ajouterEmploye(new Manutentionnaire("Jeanne", "Stocketout", 32, "1998", 45));
        p.ajouterEmploye(new TechnARisque("Jean", "Flippe", 28, "2000", 1000));
        p.ajouterEmploye(new ManutARisque("Al", "Abordage", 30, "2001", 45));
        p.afficherSalaires();
        System.out.println("Le salaire moyen dans l'entreprise est de " + p.salaireMoyen() + "
euros.");
    }
}
```

Vos notes

Exercice 20 Contrôler Saisie

Objectifs

- *Découvrir le mécanisme de la gestion des exceptions.*
- *Clause try, catch, finally.*
- *Clause throw, throws.*

Enoncé

On veut écrire la fonction saisieCorrecte qui permet de saisir correctement un entier. Si l'utilisateur saisit une donnée dont le format n'est pas celui d'un entier, le programme lève l'exception InputMismatchException.

- **Question 1**

La fonction devra traiter cette erreur en fournissant une solution alternative. Un message d'erreur sera affiché avec la proposition d'effectuer une nouvelle saisie.

Note :

La classe InputMismatchException appartient au package java.util.

Exemple d'exécution :

```
Donnez un entier :
java
Erreur de saisi
Fin
Donnez un entier :
-7
L'entier saisi est : -7
Fin
```

- **Question 2**

L'entier saisi doit être impérativement supérieur à 10.

On demande donc de créer une classe d'exception adaptée à cette erreur, puis de modifier le programme afin de traiter ce cas d'erreur.

Note

On aurait pu utiliser l'exception prédéfinie IllegalArgumentException pour vérifier que l'entier saisi est supérieur à 10.

Exemple d'exécution :

```
Donnez un entier :
java
Erreur de saisi
Donnez un entier :
4
valeur < 10
Donnez un entier :
34
L'entier saisi est : 34
```


Exercice 21 — Lecture d'un fichier properties

Objectifs

- Utiliser la classe Properties
- Utiliser la classe InputStream et FileInputStream

Enoncé

Récupérer les valeurs des différentes propriétés du fichier data.properties puis les afficher à la console
Ce fichier contient les propriétés suivantes :

```
fichier.personne=C:\data\personne.properties
fichier.article=C:\data\article.propertie
fichier.input.personnes=C:\data\input\personnes.txt
fichier.output.personnes =C:\data\input\ personnes.txt
fichier.input.articles=C:\data\input\listeArticles.txt
fichier.outpput.articles=C:\data\output\listeArticles.txt
formateur.java=davidJ
formateur.android=michel
formateur.test=davidG
```

Vos notes

Exercice 22 Modifier les données d'une personne dans un fichier

Objectifs

- *Etre capable de lire et d'écrire dans un fichier*
- *Utiliser la classe FileInputStream, BufferedReader, FileReader, BufferedWriter, FileWriter, PrintWriter*

Enoncé

Récupérer toutes les lignes du fichier qui correspond à la propriété fichier.input.personnes. du fichier data.properties
Chacune des lignes du fichier doivent être stocké dans un objet de la classe personne.
La classe personne contient les attributs suivants :

- numSS, nom, prenom, date de naissance, ville, sexe

Il doit être possible d'accéder directement à la personne à partir de son numSS dans le code.

Si le numSS correspond à une femme mettre "F" dans sexe sinon mettre M
Ecrire dans un fichier en sortie (propriété fichier.output.personnes) toutes les informations de la personne (pour chaque personne)

Description du fichier fichier.input.personnes

```
numSS
nom
prenom
date de naissance
ville
```

Description du fichier fichier.output.personnes

```
numSS
nom
prenom
sexe
annee de naissance
ville
```

Chaque attribut est séparé par un " ; " dans ces 2 fichiers.

Enoncé

Créer les classes nécessaires en respectant la programmation orientée objet
Et créer un programme permettant d'utiliser les classes créées tout en respectant l'énoncé

Vos notes

