Liste de travaux pratiques

Les TP permettent de mettre en application les différentes notions abordées dans les cours sur les variables, tableaux simples, chaînes avec String et boucles. Tous ne sont pas encore faisable.

Exercice 1 : manipulation de chaînes (String)

a. Écrire une méthode qui affiche toutes les lettres d'un mot, une ligne par lettre.

```
FORMATION-JAVA :
F
O
R
M
A
T
I
O
N
-
J
A
V
A
```

b. Écrire une méthode qui affiche toutes les lettres d'un mot, une ligne par lettre mais cette fois-ci en commençant par la fin !

```
AVAJ-NOITAMROF (à l'envers) :
A
V
A
J
-
N
O
I
T
A
M
R
O
F
```

c. Écrire une méthode qui fait la même chose que **b** mais plutôt que d'afficher directement le résultat dans la console, vous utiliserez le mot `return` dans votre

méthode pour ensuite afficher le résultat depuis votre méthode main.

Il vous suffit de retourner une chaîne qui concatène chaque lettre avec un saut de ligne et de la retourner.

Exemple :

```
// on écrit string au début pour indiquer le type de l'objet retourné par la méthode
public static String maMethode(String minus){
  String upper = minus.toUpperCase();
  return upper;
}

// voici comment l'utiliser
public static void main (String[] args){
  String unMot = "coucou";
  String resultat = maMethode(unMot); // le résultat est maintenant dans la variable résultat
  System.out.println(resultat);

// on pourrait aussi l'écrire en une seule instruction de cette manière (Exercices étant la classe qui contient la méthode static)
System.out.println(Exercices.maMethode(unMot));

}
```

Exercice 2 : Recherche d'une sous-chaîne dans une chaîne

1. Écrire une méthode qui retourne `true` si un bout de chaîne donnée en paramètre est trouvée dans le mot.

Votre méthode doit prendre 2 paramètres, un String du mot et un String du bout de mot à rechercher.

1. Écrire une méthode palindrome qui indique (true ou false) si un mot peut se lire de la même façon de gauche à droite et de droite à gauche.

N'hésitez pas à chercher sur internet ou sur le site.

Exemples :

```
"kayak" : true
"xanax" : true
"sommes" : false
```

Exercice 3 : Manipulation de tableau simple de nombre avec des calculs

Créer un package nommé **nombrehasard** puis la classe **Hasard**

Pré-requis : savoir utiliser la boucle **for**

Liste des variables de la classe :

- **tableau[ ]**
- **moyenne**
- **somme**

Ajouter 2 constructeurs :

- Un *constructeur sans argument* qui appelle le constructeur en lui envoyant une dimension de 10 par défaut.
- Un *constructeur avec un argument* qui initialise le tableau avec la dimension reçue en paramètre.

Liste des méthodes :

- **calculerMoyenne()** : permet d'effectuer le calcul de la moyenne
- **afficher()** : permet d'afficher le contenu du tableau, moyenne et somme dans la console
- **afficherNombreSupMoyenne()** : permet d'afficher la liste des nombres supérieurs à la moyenne.
- **ajouter(int nombre)** : permet d'ajouter une valeur entière à chaque nombre du tableau.

La classe Hasard est définie par 3 attributs :

- **tableau[]** qui définit un tableau de nombres.
- **moyenne** qui est la moyenne des nombres du tableau.
- **somme** qui est la somme des nombres du tableau.
- Le constructeur avec argument met des nombres tirés au hasard dans le tableau.

Exemple d'une partie du code d'une méthode avec un argument dimension (entier) pour remplir un tableau avec des nombres aléatoires :

Méthode **random()** de la classe **Math** :

```
public Hasard(int dimension)
{

    tableau=new int[dimension];
    for (int i=0;i<tableau.length;i++)
    {
        nombreAleatoire=Math.random()*100;
        tableau[i]=(int)nombreAleatoire;
    }
}
```

Méthode **nextInt()** de la classe Random :

Attention car cette méthode peut tirer aléatoirement un nombre ayant la valeur **zéro** !

```
nombre = new Random().nextInt(100); // ici on veut un nombre inférieur à 100
```

Exercice 4 : Les cadeaux du père Noël (classes, tableau, variables statiques,…)

Exercice qui nécessite des notions de base en POO.

Enoncé :

Le père Noël doit préparer la distribution des cadeaux aux enfants.

Chaque enfant a un nom, un prénom, un âge et une liste de cadeaux.

Il ne peut pas avoir 2 fois le même cadeau. Une fois qu'un cadeau est donné, il n'est plus disponible.

Cette application doit permettre au Père Noël de gérer une liste d'enfants et une liste de cadeaux avec des méthodes pour affecter à chaque enfant un cadeau.

Le Père Noël doit posséder 2 listes d'objets Enfants et Jouets. Un Jouet est caractérisé par un libellé (son intitulé) et un état pour dire s'il est ou pas distribué.

Le Père Noël est le seul capable de distribuer des cadeaux pour chaque enfant.

Au final, l'application doit permettre de créer des enfants, des jouets et d'effectuer des distributions de cadeaux aux différents enfants.

Pour vous aider, voici l'extrait du contenu de la méthode main() :

```
public static void main(String[] args) {

                // on va d'abord crée un Père Noel ! (même s'il n'existe pas)
                PereNoel papaNono = new PereNoel();

                // Ensuite on va crée 3 enfants :

                // Noémie Truc à 8 ans :
                Enfant nono = new Enfant("Bidule","Noémie",8);
                // Joachim Machin à 6 ans :
                Enfant jojo = new Enfant("Machin","Joachim",6);
                // Soufiane Touti à 9 ans :
                Enfant souf = new Enfant("Truc","Soufiane",9);

                Enfant doublon = new Enfant("Bidule","Noémie",8);

                // Ensuite on crée les jouets à distribuer :
                // on pourrait en mettre davantage

                Jouet joujou1 = new Jouet("Ferrari 308GTB");
                Jouet joujou2 = new Jouet("BarbiZou la poupée qui gazouille ");
                Jouet joujou3   = new Jouet("Super puzzle avec 180 pièces ");
                Jouet joujou4 = new Jouet("Mikado");
                Jouet joujou5 = new Jouet("Rubik's Cube");
                Jouet joujou6   = new Jouet("iPad");

                // le père Noel met dans sa hote des jouets :

                papaNono.ajouterJouet(joujou1);
                papaNono.ajouterJouet(joujou2);
                papaNono.ajouterJouet(joujou3);
                papaNono.ajouterJouet(joujou4);
                papaNono.ajouterJouet(joujou5);
                papaNono.ajouterJouet(joujou6);

                // le père Noel met dans sa liste des enfants :

                papaNono.ajouterEnfant(nono);
                papaNono.ajouterEnfant(jojo);
                papaNono.ajouterEnfant(souf);

                // il affiche la liste des enfants à visiter et
                // les cadeaux à distribuer (ça va, il a pas trop de boulot):

                papaNono.afficherListeDesEnfants();
                papaNono.afficherListeDesJouets();

                // il affecte des cadeaux aux 3 enfants :
                papaNono.distribuer(nono,joujou1);
```

```
                papaNono.distribuer(jojo, joujou2);
                papaNono.distribuer(souf,joujou3);
                papaNono.distribuer(souf,joujou4);
                papaNono.distribuer(nono,joujou5);
                papaNono.distribuer(jojo,joujou6);

                // il va afficher les enfants avec leurs cadeaux :
                nono.afficher();
                jojo.afficher();
                souf.afficher();

                // peut-il ajouter un autre jouet(joujou2) à nono ?
                papaNono.distribuer(nono,joujou2);
}
```

Exercice 5 : Analyser une phrase en profondeur (voyelles, consonnes, nombre de mots, espaces,...)

Notions abordées :

- Classe, objet
- Saisie clavier
- Tableau
- Boucles
- découverte de la classe StringTokenizer() et de diverses méthodes : split(), charAt(), indexOf(),...

Vous avez des exemples dans le cours (sinon sur le web)

Voici la liste des variables de la classe **Chaîne.java** que vous devez créer :

- maChaine : Chaîne de caractères
- nbVoyelles : entier
- nbMots : entier
- nbConsonnes : entier
- nbEspaces : entier
- nbLettres : entier

Cette classe aura le constructeur ci-dessous permettant la saisie dans la console de la chaîne à analyser :

```
public Chaine()

{
    System.out.println("Donnez une suite de mots puis validez (entrée)");
    try
    {
        InputStreamReader entree=new InputStreamReader(System.in);
        BufferedReader clavier=new BufferedReader(entree);
        maChaine=clavier.readLine();
    }
    catch(IOException e)
    {
        System.out.println("Problème de lecture en entrée !";
    }
}
```

Liste des méthodes de la classe :

Il vous faudra créer des méthodes pour trouver le nombre de voyelles, de consonnes, de mots, d'espaces et de lettres concernant un objet de type Chaine :

- **extraire()** ou **analyser()** : méthode permettant d'analyser une chaine et d'effectuer tous les calculs
- **afficher()** : méthode permettant d'afficher la chaine et les résultats de l'analyse dans la console.

Pour les plus fort(e)s : **ajouter une méthode pour déterminer la langue de la phrase saisie**.

Exercice 6 : Bibliothèque avec Collection de type HashMap

Conseils : Vous pouvez utiliser des HashMap, Iterator, Map, Entry, MapEntry, méthodes put(), remove(),...

vous allez créer un package nommé **biblio**.

Vous allez créer une classe **Livre** qui possède les caractéristiques ci-dessous :

- numéro ISBN (longueur 4 pour l'exercice, en réalité c'est plus long)
- titre (Chaîne de caractères)
- prix (nombre flottant)

Vous ajoutez le ou les constructeurs, les méthodes get() et set() ainsi que la méthode toString() à redéfinir.

Ensuite, vous allez créer la classe **GestionBiblio** qui doit contenir un tableau de Livre de type HashMap.

La syntaxe correspond à celle-ci : **HashMap<Integer, Livre>**

- **Integer** : c'est la clef du livre, ici l'ISBN (raccourci)
- **Livre** : c'est l'objet Livre correspondant à l'iSBN.

Cette classe va vous permettre :

- d'ajouter des livres
- de suprimer des livres
- retourner le nombre total de livres,
- retourner le montant total de tous les livres stockés.
- rechercher un livre par son numéro isbn.

Pour les plus fort(e)s, ajoutez une méthode qui permet de retourner les informations d'un livre en fonction de son ISBN saisie au clavier.

Dans votre méthode **main()**, vous devez réaliser les étapes suivantes :

Créer 3 livres :

Remarque : ici, l'ISBN est volontaire bidon mais vous pouvez utiliser un vrai ISBN et ajouter d'autres propriétés à votre classe Livre.

- livre 1 : isbn = 1234,titre = titre1, prix = 12.00
- livre 2 : isbn = 5678, titre = titre2, prix = 13.00
- livre 3 : isbn = 6789,titre = titre3, prix = 14.00

Listes des tâches que vous devez lancer :

1. Ajouter les 3 livres dans votre bibliothèque
2. Recherche du livre 2 avec l'ISBN n°5678 et l'afficher
3. Montant total des livres en biblio en euros
4. Afficher le nombre de livres en biblio.
5. Ajouter un livre 4 9876, Titre4, 18.00 euros
6. Lister de tous les livres
7. afficher le nouveau montant total des livres en biblio.
8. Afficher le nouveau nombre de livres en biblio.
9. Lister tous les livres
10. Supprimer le livre avec l'ISBN 5678
11. Lister de tous les livres pour vérifier que le n°5678 est bien détruit.

Exercice 7 : Gestion de comptes bancaires

en console (classes, héritage, polymorphisme,...)

Objectif : Mettre en oeuvre les notions de classes, encapsulation, constructeurs, Collections.

Pour ce faire nous allons créer une classe **Compte**. Ensuite,nous créerons des instances de **Compte** pour tester les différentes méthodes développées dans ce TP.

Enoncé

A - Mise en place de la classe **Compte**

1) Définir une classe publique **Compte**

2) Définir des variables de classe (partagées par tous les instances de Compte)... le mot clé est **static**

```
a) sous forme de variables non modifiables (constantes : mot clé **final**) :
- Nom de l'enseigne dans un String (Choisissez un beau nom :-)
- Début du RIB sous cette forme "12345 6789" dans une chaîne de caractères.

b) sous forme de variable modifiable :

- Nombre de clients dans un entier
```

3) Définir des variables d'instance

- Nom du titulaire dans une chaîne
- Numéro de compte dans un entier - Solde du compte dans un double

Remarques :

Le numéro de compte sera calculé en fonction du nombre de clients de la banque déjà enregistrés.
Toutes les variables doivent être **private**.

4) Définir plusieurs constructeurs

- Les constructeurs ont pour but d'initialiser les attributs : nom du titulaire, numéro de compte, solde initial, nombre de clients.

  a) Définir un constructeur demandant comme paramètre un nom de titulaire. Dans ce cas le solde initial sera fixé à 0 par défaut.

  b) Définir un second constructeur demandant comme paramètres un nom de titulaire + un solde initial.

Remarques :

- Penser à incrémenter le nombre de clients à chaque création d'un nouveau compte.
- Pour économiser le codage, faire en sorte que le premier constructeur fasse appel au second.
- A chaque création de nouveau compte, le numéro de compte dépend du nombre de clients.

5) Définir des méthodes d'instance

```
a) prévoir une méthode qui renvoie le solde du compte
b) prévoir une méthode qui affiche le solde
c) prévoir une méthode de dépôt d'argent qui met à jour le solde et affiche le
nouveau solde.
d) prévoir une méthode de retrait qui met à jour le solde seulement si retrait
n'est pas supérieur au solde. Dans le cas d'un refus, faire afficher un message,
sinon afficher le nouveau solde.
e) Définir une méthode qui affiche les caractéristiques d'un compte :
```

ENSEIGNE (nom de la banque) Numéro complet du compte (Début de RIB + numéro)
Nom du titulaire
Solde du compte

B - Classe **GererComptes**

1) Ajouter à votre projet une seconde classe munie d'une méthode **main()**
2) Dans cette méthode main(), réaliser les opérations suivantes :

```
a) Créer un 1er compte en ne spécifiant que le nom du titulaire.
b) Effectuer un dépôt sur ce compte d'un montant de 3000 euros.
c) Afficher les renseignements complets de ce compte.
d) Créer un deuxième compte en spécifiant le nom du titulaire et solde initial.
e) Effectuer un dépôt de 80 euros sur ce compte
f) Effectuer un retrait inférieur au solde.
g) Effectuer un retrait supérieur au solde.
h) Afficher les renseignements complets de ce compte.
```

3) Compiler et tester

4) Bonus : Mettre en place une classe **Titulaire** et modfier votre code.

Un titulaire possède un prénom, un nom et un numéro de téléphone.

Ouf ! You are the best !