Devoir à remettre - Programmation en java

Méthodes d'instance et de classe, Chaines de caractères et énumérations

Objectifs : On s'intéresse à la manipulation de:

- méthodes d'instance et de classe (méthodes déclarées static ou non)
- chaines de caractères (les classes **String** et **StringBuilder**)
- types énumérés (les classes de type **enum**)

Pré- requis : Les notions nécessaires à la réalisation du travail se trouvent dans le chapitre 3 et le chapitre 4 du cours.

Vous devez connaitre les notions suivantes :

- Implémentation et instanciation d'une classe en java (déjà vu)
- Implémentation et manipulation d'un tableau d'objets (déjà vu)
- La référence « this » (déjà vu)
- Les méthodes statiques (déclarées avec le modificateur static – voir chapitre 3)
- Les méthodes de la classe String et StringBuilder (voir chapitre 4)
- Les types énumérés (voir chapitre 4)

Exercice 1

Indication : méthodes d'instance et de classe (static) – voir le chapitre 3 pour les méthodes static

Donner l'implémentation d'une classe **Date** comportant trois attributs (jour, mois, année). Il faut définir la classe.

La classe doit comporter :

Un constructeur sans paramètres et un constructeur avec paramètres,

Une méthode *vérif ()* qui vérifie si une date est correcte ou non (le cas du mois de février doit être vérifié aussi).

Une méthode *saisir ()* qui saisit une date au clavier en vérifiant que la date est correcte et demande de refaire la saisie jusqu'à introduire une date correcte.

Une méthode *afficher()* qui affiche une date (au format JJ-mm-aaaa)

Une méthode *an_Bissextile ()* qui vérifie si la date comporte une année bissextile ou non.

Une méthode *inférieure (Date d2)* qui vérifie si une date est inférieure à une autre

Une méthode *supérieure (…)* qui vérifie si une date est supérieure à une autre

Une méthode *statique compare (…)* (static int compare(Date d1, Date d2)) qui compare deux dates d1 et d2, retourne 0 si les deux dates sont égales, -1 si la première est inférieure à la deuxième et 1 sinon.

Une méthode *compareTo (…) – une méthode d'instance est une méthode qui n'est pas statique*

Une méthode Aujourd'hui () qui récupère la date système et la retourne dans votre objet de type Date

Une méthode lendemain () qui calcule la date du lendemain et la retourne

Une méthode *dateAprès* (int nb) qui calcule la date après un nombre nb de jours et la retourne.

Une méthode statique *dateAprès* (Date d, int nb) qui calcule la date d après un nombre nb de jours et la retourne.
Une méthode *dateAvant* (int nb) qui calcule la date avant un nombre nb de jours et la retourne.
Une méthode toString () qui convertit une date en chaine de caractères (String)

Dans une classe ProgDate, écrire une méthode main qui :

- crée un vecteur V de type Date comportant 20 objets de type Date (par exemple T[0] = new Date (15, 3, 2020))
- affiche le vecteur
- trie le vecteur par *ordre croissant* des dates et l'affiche
- remplace chaque date du vecteur par la date après nb jours (nb donné). Utiliser la méthode *dateAprès* déclarée static
- affiche le vecteur
- crée un vecteur S de type String contenant les dates au format String et l'affiche

Exercice 2

Indication : Utiliser les méthodes de la classe **String** – voir le chapitre 4

On considère un tableau T de taille n (donnée) contenant des chaînes de caractères (de type String) que l'on doit remplir par saisie au clavier.
Ecrire une classe **ProgTest** contenant une méthode main qui :

- Crée le tableau T et le remplit avec des chaines de caractères quelconques
- Affiche toutes les chaînes du tableau en majuscules (il s'agit d'afficher la forme majuscule de chaque chaine).
- Affiche toutes les chaînes du tableau en minuscules
- Affiche toutes les chaînes ayant S pour suffixe (toutes les chaines se terminant par S. S est une chaîne de caractères donnée par exemple S = "ant")
- Affiche toutes les chaînes supérieures à une chaine S donnée (utiliser la méthode compareTo).
- Remplace toutes les chaines commençant par 'a' dans le tableau, par la chaine "Hello".

Exercice 3

Indications
Utiliser les méthodes de la classe **String** – voir chapitre 4
Par exemple : **ch.charAt(i)** renvoie le caractère qui se trouve à l'indice i de la chaine ch
Utiliser les méthodes de la classe **Character** – voir les compléments « Classes Enveloppes »
Par exemple : **Character.isLetter (char c)** renvoie *true* si le caractère *c* est une lettre alphabétique et *false* sinon.

Définir une classe java appelée **Chaine** comportant un attribut *ch* de type String (chaine de caractères)

Dans cette classe, ajouter :

- un constructeur avec un paramètre
- une méthode *longueur()* qui retourne la longueur de la chaine
- une méthode *affiche()* qui affiche la chaine caractère par caractère (les caractères doivent être séparés par une tabulation '\t')
- une méthode *appartient (char x)* qui vérifie si x appartient à la chaine
- une méthode *nbChiffres()* qui retourne le nombre de caractères *numériques* dans la chaine

- une méthode *nbLettres()* qui retourne le nombre de caractères *alphabétiques* dans la chaine
- une méthode *nbAutres()* qui retourne le nombre de caractères qui sont ni lettre ni chiffre dans la chaine
- une méthode *estEgal (Chaine S2)* qui vérifie si la chaine contenue dans S2 (l'attribut ch de S2) est la même que la chaine contenue dans S1 (l'attribut ch de S1 qui est référencée par « this»).
- une méthode estPalind() qui vérifie si une chaine (l'attribut ch) est palindrôme.
- une méthode *estInverse (Chaine S2)* qui vérifie si la chaine contenue dans S2 (l'attribut ch de S2) est l'inverse de la chaine contenue dans S1 (l'attribut ch de S1 qui est référencée par « this »).
- Une méthode *String inverse ()* qui inverse l'attribut ch de l'objet et la retourne dans un String.

Définir une classe **ProgChaine** comportant une méthode main() qui :

- crée un vecteur VS d'objets de type Chaine (pas de type String) de taille n (donnée)
- crée un vecteur VT d'entiers comportant dans chaque case i la taille de la chaine i en utilisant la méthode *longueur*.
- affiche toutes les chaines qui comportent des caractères spéciaux (ni chiffre, ni lettre)
- affiche toutes les chaines qui comportent le caractère 'z'
- vérifie s'il existe dans le vecteur une chaine qui est inverse d'une autre
- compte le nombre de chaines palindrômes dans le vecteur VS et les affiche
- vérifie si la chaine qui se trouve à VS [0] se répète dans le vecteur et combien de fois elle se répète.

NB : chaque chaine doit être affichée dans une ligne à part (saut de ligne ('\n') après l'affichage d'une chaine)

Exercice 4

Indications
Utiliser les méthodes de la classe **StringBuilder** – voir chapitre 4
Les objets de la classe StringBuilder peuvent être modifiés contrairement aux objets de la classe String qui sont non-modifiables (on dit qu'ils sont « immuables »)

On considère un tableau T de taille n (donnée) contenant des chaînes de caractères (de type StringBuilder) que l'on doit remplir par saisie au clavier.
Ecrire une classe **ProgStrBuilder** contenant une méthode main qui :

- Crée le tableau T et le remplit avec des chaines de caractères quelconques
- Affiche toutes les chaînes du tableau en majuscules (il s'agit d'afficher la forme majuscule de chaque chaine).
- Remplace le troisième caractère de chaque chaine par le caractère '-' (un tiret) et affiche la chaine modifiée.
- Concatène l'entier 2020, à toute chaine du tableau qui a une longueur inférieure à 5 et affiche de nouveau le contenu du tableau (les chaines doivent être séparées par une tabulation)

Exercice 5 - les types énumérés (voir - chapitre 4)

1. Définir un type énuméré nommé **Pays** permettant de représenter vingt pays de votre choix (ALGERIE, MAROC, ALLEMAGNE, EGYPTE, …).

Remarque : Dans la suite, les méthodes de la classe d'énumération Pays (enum Pays) doivent être déclarées **static**.

- Ajouter une méthode *afficher ()* qui affiche les éléments de l'énumération Pays.
- Ajouter une méthode *accepteChaine (String ch)* qui vérifie si la chaine ch correspond à un objet de type Pays (que vous avez considéré dans votre énumération).
- Ajouter une méthode *getPays (int numéro)* qui reçoit un entier en entrée (numéro d'ordre) et rend le nom de pays correspondant, si la valeur ne correspond à aucun pays l'objet null est retourné.

2. Écrire un programme ProgEnumPays réalisant :

- l'affichage des éléments de l'énumération Pays.
- le remplissage d'un tableau de *n* éléments de type pays en utilisant un objet Scanner. On insère l'objet ALGERIE si la chaine saisie ne correspond pas un objet de type **Pays (de votre énumération)**.
- affiche les éléments du tableau.

3. Compléter la classe Pays précédente, de manière à associer à chaque nom de pays :
   - Le nom de la capitale,
   - La langue officielle du pays
   - Le nombre d'habitants,

- Ajouter une méthode *afficher* (Pays p) qui permet d'afficher le nom de la capitale, la langue officielle et le nombre d'habitants du pays p donné en paramètre.
- Une méthode *rechercherParCap* (String cap) qui retourne l'objet Pays ayant pour capitale, une chaine *cap* donnée en paramètre si elle existe, sinon on retourne null.

4. Modifier le programme (classe **ProgEnumPays**) pour réaliser :
   - L'affichage des objets de l'énumération Pays (en spécifiant les attributs de chaque pays) pour lesquels la langue officielle est l'*arabe*.

Remarque : Il n'est pas exigé mais IL serait intéressant de présenter l'exécution des différents exercices à l'aide de menus (utilisation de l'instruction switch).

NB : le travail doit être fait par binôme (ou monôme, le cas échéant).

Le travail doit être envoyé à l'adresse recupspace@gmail.com, dans 3 semaines (9 avril 2020)

Les programmes écrits doivent être lisibles et bien commentés.

Spécifier dans l'objet du mail DevoirPOO-L2 et préciser les noms des étudiants

PS : Il faudra envoyer dans un fichier .zip, le code source de chaque classe et un jeu de tests (exécution du programme) illustré avec des captures d'écran dans un fichier pdf avec des affichages clairs et bien faits.