Grands Moulins

24/06/2021 - 8h30-10h30

Initiation à la Programmation (IP2)

Aucun document autorisé. Dans vos réponses n'utilisez **aucune des librairies de java** qui sont en rapport avec les listes, vous devez écrire tout ce qui vous est utile.

On rappelle que vous disposez de brouillon pour réfléchir, et que les réponses bien conçues tiennent en quelques lignes seulement. Cela ne vous prendra pas beaucoup plus de temps de n'écrire au propre que lorsque vous avez les idées claires : d'ailleurs voir des erreurs raturées, des choses confuses, peu lisibles, montrerait que vous maîtrisez mal votre sujet.

Les exercices sont largement indépendants ; ils apparaissent dans l'ordre que nous avons suivi pour la progression du cours. Le barème est indicatif, il vous donne une idée du temps à consacrer aux exercices. S'il évolue légèrement ce sera pour donner de l'importance aux exercices traités complètement et correctement. Inutile donc de vous précipiter : si vous faites bien la moitié des exercices vous aurez certainement une meilleur note que si vous accumulez des erreurs.

Exercice 1 (5 points)

1. (1.5 points) Ecrivez une méthode alterne qui prend en argument un tableau de chaînes de caractères et qui les affiche en prenant d'abord la première, puis la dernière puis la seconde puis l'avant dernière etc... chaque chaîne étant affichée une et une seule fois seulement.

2. (1.5 points) On considère des tableaux d'entiers, par exemple $[2; 5; 3; 7; 10; 5; 2]$, sur lesquels on effectue un calcul qui consiste à faire d'abord une addition $2 + 5 = 7$, puis une soustraction en reprenant le résultat précédent $7 - 3 = 4$ puis une multiplication $4 * 7 = 28$, puis une division $28/10 = 2.8$ et on recommence ainsi de suite tant que c'est possible (une addition, une soustraction, une multiplication, une division etc ...) c'est à dire qu'ici on produit encore 7.8 puis 5.8 à la fin.

   Ecrivez une méthode calculette qui prend en argument un tableau d'entier et qui effectue ce travail.

3. (2 points) On considère des tableaux de longueurs paires contenant des entiers strictement positifs (inutile de le vérifier). Par exemple $[2; 3; 4; 1; 1; 7]$, que l'on interprète en se disant qu'il faut répéter 2 fois l'entier 3, puis 4 fois l'entier 1, puis 1 fois l'entier 7. On souhaite obtenir le résultat sous la forme d'un tableau, c'est à dire ici $[3; 3; 1; 1; 1; 1; 7]$. Ecrivez la méthode expand qui effectue ce travail sur son argument. Expliquez votre démarche.

Exercice 2 (4 points)

Ecrivez (présenté clairement) une structure de liste chaînée ListeResultat qui permet de stocker les résultats de chaque étudiant à un examen (le nom de l'étudiant et sa note). Ecrivez les constructeurs, une méthode d'ajout qui interdit les doublons de nom, une méthode qui permet de consulter la note d'un étudiant. Expliquez vos choix si nécessaire.

Exercice 3 (6.5 points)

Les rendez-vous pour des séances de consultation de copies sont enregistrés dans une liste qui prend comme information : (nom d'étudiant, heure de début, durée). Par nature, on fera en sorte que ces rendez-vous soient toujours stockés dans l'ordre chronologique.

Pour simplifier la gestion du temps, et ne pas s'occuper des problèmes de conversion heures/minutes, on utilisera les entiers pour modéliser le temps : le premier rendez vous aura lieu à l'instant 0, et on restera dans le domaine des minutes.

Les rendez-vous stockés dans une liste concernent un même examen, et on prévoit la règle suivante : si un étudiant a une note supérieure ou égale à 10 il n'aura besoin que de 5 minutes de consultation, et s'il n'a pas validé il aura droit à 10 minutes. Seuls ceux qui le souhaitent prendront rendez-vous.

1. (1.5 points) Ecrivez une classe RendezVous (ainsi qu'une pour ses cellules). Munissez là d'un constructeur RendezVous(ListeResultat res). Ecrivez constructeurs et accesseurs pour les cellules. (On s'intéressera particulièrement à ce que vous choisissez de faire de cette liste res ; ListeResultat a été défini à l'exercice précédent)

2. (1 point) Ecrivez une méthode supprime(String nom) qui servira à supprimer le rendez-vous de l'étudiant identifié (sans toucher aux autres)

3. (2 points) On planifie les rendez-vous individuellement, c'est à dire qu'il n'y aura personne d'autre de prévu sur la période $[debut, debut + duree]$. Entre deux rendez-vous successifs on aménage même une pause de 2 minutes. L'inscription se fait par la méthode int prendRendezVous(String nom), le moment où il aura lieu doit être fixé automatiquement : le plus tôt possible, sans décaler les autres rendez-vous, et en tenant compte des contraintes déjà expliquées. Ecrivez cette méthode, elle retourne l'instant où est programmé le rendez-vous.

4. (2 points) On souhaite ensuite répartir les rendez-vous entre enseignants : on va en extraire une section, la plus grande possible, dans laquelle la somme des durées de consultation reste inférieure à une heure. Ecrivez une méthode RendezVous decoupeBlocUneHeureConsultation() qui permette cette fonctionnalité, expliquez clairement votre démarche.

Exercice 4 (5 points) On s'intéresse à des arbres binaires pour encoder une forme d'alphabet morse. On rappelle rapidement qu'un mot en morse est une suite de points et de traits ; comme pour les mots en français on séparera les mots en morse par des espaces, ce qui nous permet d'écrire un texte complet.

Dans la classe String on peut trouver une méthode split qui sépare une longue chaîne de caractères en mots. Ainsi si s="un petit test", le résultat de s.split(" ") retourne ["un","petit","test"].

On rappelle également que la méthode char charAt(int i) retourne le ième caractère d'un mot, et que String substring(int i) retourne la sous chaine qui commence à l'index i

Les noeuds de l'arbre utilisés pour l'alphabet morse possèdent deux fils : filsP et filsT, respectivement pour fils-point, et fils-trait. Lorsqu'on décodera une suite de points et de traits, on suivra le chemin des fils correspondants en partant de la racine. Le noeud obtenu au bout du chemin contiendra un caractère qui est le décodage du mot morse.

Si une suite de points et de traits ne donne rien dans l'arbre de l'alphabet, on considérera qu'elle représente '#' en français. Inversement, s'il y a un caractère en français qu'on ne peut pas trouver dans l'alphabet morse alors il sera simplement ignoré lors du codage.

1. (0.5 point) Ecrivez la déclaration des classes Alphabet et Noeud (juste les attributs, pas la peine d'écrire ses constructeurs)

2. (2 points) Ecrivez une méthode traduitReception qui prend un texte en morse en argument (on suppose qu'il est bien formé, ce n'est pas la peine de vérifier) et qui retourne le texte correspondant en français.

3. (2.5 points) On s'intéresse aussi à la traduction dans l'autre sens : écrivez une méthode String emetSimpleWord(String m) qui prend en argument un mot en français (sans espace, pas la peine de le vérifier) et qui retourne son codage en morse.