IN3S02 – TP5 2011/2012 v1

Sujet (n'imprimer le pdf qu'une seule fois par poste de travail)

Durée : 3 h

1 OBJECTIFS

- Maîtriser l'utilisation des tableaux et des boucles
- Apprendre à lancer un programme en ligne de commande et à utiliser les arguments y figurant

2 TRAVAIL A REALISER

Nota : le travail demandé doit être terminé, en séance ou, à défaut, hors séance.

Visualiser le sujet dans un navigateur pour bénéficier des liens.
Créer un répertoire `tp5` dans `In3s02` sur votre compte.

2.1 Exercice 1 : Utilisation de la ligne de commande

1. Créez un nouveau fichier `Moyenne.java` pour contenir la classe `Moyenne` sans attribut ni constructeur, dans laquelle nous construirons **progressivement** une méthode permettant de calculer la moyenne des nombres présents sur la ligne de commande.
   Par exemple, la commande `java Moyenne 9.5 10 15` devra afficher `moyenne=11.5`
2. Définir la fonction `moyenne()` vue en TD. L'essayer par exemple avec `{ 9.5, 10., 15. }`.
3. Modifier cette fonction pour qu'elle accepte maintenant un tableau de `String` au lieu du tableau de `double`. Nous supposerons que chaque `String` représente bien un `double` (par ex.: `"3.14"`).
   Aide : découvrir la méthode `parseDouble()` de la classe `Double` ne sera pas une perte de temps !
   Contrainte pour tout l'exercice 1 : Ne pas utiliser de tableau intermédiaire.
4. Transformer cette fonction en procédure à un seul paramètre (le nombre d'éléments utiles sera considéré comme égal à la taille du tableau). D'autre part, elle ne devra plus retourner le résultat, mais l'afficher. S'il n'y a aucun nombre, afficher le message `pas de nombre !` .
5. Lire l'explication sur la fameuse méthode `main()` et modifier le nom de la procédure pour que votre programme fonctionne à partir de la ligne de commande. L'essayer dans un terminal Linux.

2.2 Exercice 2 : Le crible d'Ératosthène

Cet exercice va consister à réaliser l'exercice décrit dans le sujet du TD5.

2.2.1 Créer un nouveau projet

Créer *dans le répertoire tp5*, précédemment créé, un nouveau projet BlueJ de nom eratosthene.

2.2.2 Créer une nouvelle classe `Eratosthene`

Attention ! Déclarer deux attributs `aTab` et `aMax`, et pour pouvoir tester les différentes méthodes depuis l'extérieur de cette classe, laisser toutes les méthodes en public.
Créer pour l'instant un constructeur à un paramètre entier qui initialise uniquement `aMax`.

2.2.3 Tester la première méthode

Dès que la méthode `initV()` est écrite, **tester** son bon fonctionnement en incorporant par copier/coller la classe `EratostheneTest` dans votre projet, puis exécuter les tests.
Tout est vert ? ***(interdiction de modifier EratostheneTest)***

2.2.4 Continuer et tester d'autres méthodes

Écrire ensuite successivement `raye()` et `prepare()`.

Tester au fur et à mesure le bon fonctionnement de chacune de ces 2 méthodes en décommentant la procédure de test correspondante dans la classe `EratostheneTest`, puis exécuter les tests.
Tout est vert ? **Compléter** maintenant le constructeur comme indiqué dans le TD.

2.2.5 Continuer et tester la classe

Écrire ensuite successivement `estPremier()` et `affiche()`.

Tester au fur et à mesure le bon fonctionnement de chacune de ces 2 méthodes en décommentant la procédure de test correspondante dans la classe `EratostheneTest`, puis exécuter les tests. Tout est vert ?

2.2.6 Créer une méthode `essai()`

Passer un paramètre caractère pC et un paramètre entier pN et tenir compte des consignes suivantes :
- Le paramètre pC sera interprété comme une commande : **d**isplay, **g**reatest, **h**elp
- Le paramètre pN sera interprété différemment selon la commande ci-dessous
- `'d'` appellera `affiche()`
- `'g'` devra trouver le plus grand nombre premier inférieur ou égal à `pN`
- `'h'` affichera un message d'aide listant les commandes possibles (`pN` ne sert à rien dans ce cas)
- tout autre caractère provoquera l'affichage d'un message d'erreur signalant la commande `"h,0"`

Vérifier le bon fonctionnement de ce programme en testant extensivement.

2.3 Exercice 2 : Projet "weblog-analyzer" (tableaux)

Les serveurs web maintiennent habituellement des fichiers d'historique des accès aux pages web qu'ils supportent. Ces fichiers sont appelés fichiers log.
L'analyse de ces fichiers permet d'obtenir des informations utiles comme : quelles sont les pages les plus consultées, quelles sont les périodes de consultation les plus chargées, ...
Le projet weblog-analyzer sur lequel nous allons travailler est un programme qui réalise une analyse élémentaire d'un fichier log simplifié (fourni). C'est dans ce cadre que nous créerons et gérerons des tableaux.

2.3.1 Ouvrir le projet

Télécharger le fichier `weblog-analyzer.jar` lié à cet énoncé, et l'enregistrer dans le répertoire `tp4` précédemment créé.

Lancer *BlueJ* et ouvrir, le fichier `.jar` sauvegardé ci-dessus. [menu *Projet*, choix *Ouvrir non-BlueJ* ...].

2.3.2 Prendre connaissance du projet weblog-analyzer

Le projet weblog-analyzer est composé de 4 classes. Nous nous intéresserons essentiellement à la classe LogAnalyzer. Ce programme réalise une analyse temporelle d'un petit fichier log.

Nommé weblog.txt, il se trouve maintenant dans le répertoire tp5/weblog-analyzer. Il contient une suite de dates au format "année mois jour heure minute", chaque ligne étant censée correspondre à un accès à une page web du serveur. Visualiser le contenu de ce fichier et en prendre connaissance.

Editer et prendre connaissance de la classe LogAnalyzer. La méthode analyzeHourlyData compte combien d'accès ont été réalisés dans chaque tranche horaire pendant toute la durée couverte par le log. Le résultat est mémorisé dans le tableau hourCounts.

Créer un objet de la classe LogAnalyzer et exécuter la méthode analyzeHourlyData puis la méthode printHourlyCounts. Quels sont les moments les plus chargés et les moins chargés d'une journée ?

2.3.3 Compléter la classe LogAnalyzer

On se propose d'ajouter à la classe LogAnalyzer de nouvelles méthodes d'analyse. Aucune de ces méthodes ne devra comporter d'instruction d'affichage.

1. **Écrire** une fonction numberOfAccesses ***(combien de paramètres ?)*** qui, en sommant les éléments du tableau hourCounts, retourne le nombre total d'accès enregistrés dans le fichier log.
   Tester cette méthode en incorporant à votre projet la classe `WeblogAnalyzerTest`, et cliquer sur le bouton « exécuter les tests ». Tout est vert ? ***(interdiction de modifier WeblogAnalyzerTest)***
2. Si ce premier test s'est bien passé, **écrire** une fonction busiestHour ***(combien de paramètres ?)*** qui, en analysant le tableau hourCounts, renvoie la première tranche horaire la plus chargée.
   Tester cette méthode en décommentant la méthode de test correspondant, et « exécuter les tests ». Tout est vert ? ***(interdiction de modifier WeblogAnalyzerTest)***
3. **Écrire** une fonction quiestHour ***(combien de paramètres ?)*** qui, en analysant le tableau hourCounts, renvoie la première tranche horaire non vide la moins chargée. Vérifier notamment le bon fonctionnement de cette méthode dans le cas d'un tableau hourCounts contenant plusieurs zéros, en particulier au début (*ce qui est le cas du fichier weblog.txt fourni*), et retourner -1 s'il n'y a que des zéros.
   Contrainte : ne parcourir le tableau qu'une seule fois.
   Tester cette méthode en procédant comme pour la précédente. Tout est vert ?
4. Écrire une fonction entière busiestTwoHours ***(combien de paramètres ?)*** qui, en analysant le tableau hourCounts, détermine (et retourne) la première période de 2 heures consécutives la plus chargée (désignée par la 1ère heure de cette période).
   Tester cette méthode en procédant comme pour les précédentes. Tout est vert ?
5. **Écrire** une classe HourCount et une fonction rankHours2 :
   - Définir dans le projet weblog-analyzer une nouvelle classe de nom HourCount, comportant deux attributs : un entier hour et un entier count, de constructeur HourCount(int pHour, int pCount), et offrant deux méthodes d'accès : getHour() et getCount().
   - Dans la classe LogAnalyzer, définir une fonction rankHours2 ***(combien de paramètres ?)*** qui, en analysant le tableau hourCounts, renvoie un tableau de type HourCount[] des tranches horaires classées par valeurs décroissantes de compteur (mais tranches croissantes si compteurs égaux).

   - Exemple : Supposons que le tableau hourCounts soit le suivant :

Elément	45	12	8	72	45	3	75	80	10	12	33	80
Indice	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*

   Le tableau rankHours2 serait initialisé comme suit :

Elément	0;45	1;12	2;8	3;72	4;45	5;3	6;75	7;80	8;10	9;12	10;33	11;80
Indice	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*

   Le résultat de rankHours2 trié serait le tableau suivant :

Elément	**7**;80	**11**;80	6;75	3;72	**0**;45	**4**;45	10;33	**1**;12	**9**;12	8;10	2;8	5;3
Indice	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*

   - **Aide :** Pour trier le tableau, on peut employer le « tri à bulles ». Cette méthode consiste à répéter un « parcours » jusqu'à ce que celui-ci n'ait produit aucun échange de place. Un parcours consiste à parcourir tout le tableau en comparant les valeurs situées dans 2 cases consécutives et en les échangeant si elles ne sont pas dans le bon ordre.
   - **Tester** cette méthode en décommentant les 2 dernières méthodes de test. Tout est vert ?

2.4 Terminer la séance

Si pas fait antérieurement, **générer** (après l'avoir complétée !) la documentation des 3 exercices, puis sauvegarder les projets ouverts, puis fermer BlueJ [menu Project, choix Quit]. Se déloger.

Ce sujet a été élaboré par Denis Bureau (d'après un sujet d'Albin Morelle pour le 2.3).