IN101 – TP5 2009/2010

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

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 `In101` sur votre compte.

2.1 Exercice 1 : Calcul de moyenne

1. Créez un nouveau fichier `Moyenne.java` pour contenir la classe `Moyenne` sans attribut ni constructeur, mais avec la fameuse méthode `main()` (lire les explications).
   Contrainte : Ne pas utiliser de tableau intermédiaire.
2. Les arguments sur la ligne de commande seront supposés être des nombres.
3. Ce programme devra calculer et afficher la moyenne des notes réelles passées sur la ligne de commande. Par exemple, la commande `java Moyenne 9.5 10 15` doit afficher `moyenne=11.5`
   Aide : découvrir la méthode `parseDouble()` de la classe `Double` ne sera pas une perte de temps !
4. S'il n'y a aucun nombre, afficher le message `pas de nombre !` .

2.2 Exercice 2 : Le crible d'Ératosthène

Cet exercice va consister à réaliser l'exercice décrit dans le sujet du TD5 (non planifié).

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 ! Pour pouvoir tester les différentes méthodes depuis l'extérieur de cette classe, laisser toutes les méthodes en public. Déclarer 2 attributs aTab et aMax.
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 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 3 (à commencer en séance) : Le mini-projet « groupes »

Cet exercice va consister à réaliser l'exercice décrit dans le sujet du devoir. Tout étudiant qui le terminera en travail individuel pourra le rendre au titre de MA101(Math.Gene.)+IN101.

2.3.1 Créer un nouveau projet

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

2.3.2 Créer une nouvelle classe `TableGroupe`

- L'objectif est de programmer des méthodes pour tester les différentes propriétés que doit vérifier un ensemble fini muni d'une loi pour être un groupe. La loi sera donnée par une table de Cayley (à double entrée) contenant des caractères représentant les éléments de l'ensemble.
- Cette classe possèdera les **attributs suivants** : `aCard` (le cardinal de l'ensemble), `aTab` (la table de Cayley de la loi), `aLoi` (le caractère représentant la loi), et `aNeutre` (le caractère représentant l'élément neutre).
- **Créer** un constructeur à un paramètre *tableau de* `String` et un accesseur `getLoi()`.
  Le constructeur devra extraire de chaque `String` une ligne de la table, sachant que la première ligne et la première colonne contiendront dans le même ordre les éléments de l'ensemble, et que la première case du tableau contiendra le caractère choisi pour la loi (différent des éléments de l'ensemble !).
- **Contrainte pour toute la suite :** tous les tests d'égalité ou de différence entre deux éléments devront être effectués en utilisant les méthodes `U.egal(x,y)` ou `U.diff(x,y)` au lieu de `x==y` ou `x!=y` .
  Le nombre de comparaisons effectuées sera ainsi automatiquement comptabilisé.

2.3.3 Écrire des méthodes booléennes de vérification de base

- `loiOK()` : doit vérifier que le caractère est bien différent de ceux de l'ensemble. (1)
- `tabEstEnsemble()` : doit vérifier que la première ligne constitue bien un ensemble. (2)
- `c0EstIdentique()` : doit vérifier que la première colonne est identique à la première ligne. (3)
- `estElement(char)` : doit vérifier que le paramètre est bien élément de l'ensemble.
- `loiEstinterne()` : doit vérifier que tout élément de la table appartient bien à l'ensemble. (4)
- Construire pour chacun des 4 cas ci-dessus un contre-exemple et les tester.

2.3.4 Terminer la classe et tester extensivement

- Écrire une procédure `affiche()` pour afficher la table.
- Pour chaque propriété mathématique à tester, écrire une fonction booléenne et éventuellement une fonction auxiliaire pour faciliter l'écriture de la première.
- Écrire éventuellement une fonction `getResults()` pour rassembler tous les résultats en une String avant de les afficher.
- Écrire une procédure `test1(mes,tab)` qui affiche le message `mes` identifiant le test en cours, crée une `TableGroupe` à partir de `tab`, puis effectue tous les tests et les affichages.
- Écrire une procédure `test()` qui regroupe tous les appels à `test1()`.
- Suivre toutes les autres instructions données dans le sujet du devoir.

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 Projet, choix Quitter]. Se déloger.

Ce sujet a été élaboré par Denis Bureau (avec Leïla Reille pour le 2.3).