

IN101 : Sujet du TP8 v1

ESIEE Engineering, Denis BUREAU, mai 2010.
(à n'imprimer qu'une seule fois par poste de travail)

1 Les objectifs

Être capable de lire et écrire dans des fichiers de texte, et de développer des applettes simples.

2 Fichiers de texte

1. Créez un nouveau fichier `Numerote.java` pour contenir la classe `Numerote` sans attribut ni constructeur, mais avec la fameuse méthode `main()`
2. Ce programme devra numéroter les lignes d'un fichier de texte. Pour cela, il commencera par tenter d'ouvrir un `Scanner` sur le fichier dont le nom est passé en 1^{er} argument de la ligne de commande. Ensuite, il tentera d'ouvrir un `PrintWriter`, soit sur le fichier dont le nom est passé en 2^{ème} argument de la ligne de commande, soit sur `System.out` s'il n'y a pas de 2^{ème} argument. Enfin, il appellera la procédure `numerote()` décrite ci-dessous.
3. Écrivez la procédure de classe `numerote()` qui prend en paramètres un `Scanner` et un `PrintWriter`. Pour chaque ligne lue, elle doit écrire le numéro de ligne sur 4 chiffres, suivi d'un deux-points, d'un espace, et de la ligne lue.
4. N'oubliez pas de fermer les fichiers. Quel est le bon endroit ?
Aide : `System.exit(n);` sort immédiatement du programme en retournant le code d'erreur n .
5. Essayez ce programme avec 0, 1, 2, et 5 arguments. Ne serait-il pas souhaitable de signaler les 3 arguments inutiles (donc à ignorer) ? Modifiez/retestez.

3 Applette

Il s'agit ici de créer une applette de toute pièce. Pour cela, suivez scrupuleusement les indications. Par contre, vous devrez trouver par vous-même ce qu'il faut importer, étant entendu que les instructions `import truc.machin.*;` sont interdites.

3.1 La classe `CalcApplet`

1. Créez un nouveau fichier `CalcApplet.java` pour contenir la classe `CalcApplet` sans attribut ni constructeur, qui hérite de `JApplet`.
Cette applette devra permettre d'ajouter ou de retirer à une liste une note / N, afficher la liste courante et la moyenne courante, tout en gérant les erreurs possibles (essayez-la !) ; N représente la note maximale admissible et est **passé en paramètre à l'applette** sous forme d'un entier.
2. La seule méthode à écrire est `init()` qui doit commencer par les lignes suivantes :

```
// Il y a un conflit de securite avec les navigateurs courants (incluant
// Netscape & Internet Explorer) qui interdisent l'accès à la queue
// d'évenements d'AWT -- ce dont les JApplets ont besoin au démarrage.
// Ne pas modifier les 2 lignes suivantes :
JRootPane rootPane = this.getRootPane();
rootPane.putClientProperty("defeatSystemEventQueueCheck", Boolean.TRUE);
```

3. Elle continue ensuite par l'extraction de l'entier passé en paramètre à l'applette, puis son utilisation pour créer un `CalcPanel` (voir 3.2 ci-après), puis l'ajout (`add`) de ce `CalcPanel`.

En cas d'exception, faire :

```
JLabel lab = new JLabel( e.toString() );
lab.setVerticalAlignment( JLabel.BOTTOM );
lab.setHorizontalAlignment( JLabel.CENTER );
add( lab );
```

4. Elle se termine dans tous les cas par `setVisible(true);`

3.2 La classe `CalcPanel`

1. Créez un nouveau fichier `CalcPanel.java` pour contenir la classe `CalcPanel` qui hérite de `JPanel` (voir la JavaDoc de cette classe et de toutes les nouvelles classes évoquées ci-dessous). Ce composant graphique contiendra tout ce que l'applette va afficher. Ne pas oublier de lui ajouter (`add`) tous les composants, et d'ajouter à chaque composant tous ses sous-composants.

2. Les attributs sont :

- `maxN`, la note maximum (entière)
- `ald`, la liste de notes réelles (`ArrayList`)
- des `JLabel` permettant d'afficher du texte : `listeL` (pour afficher la liste de notes), `moyL` (pour afficher la moyenne, précédée de =), et `errL` (pour afficher les messages d'erreurs).
- des `JTextField` permettant de saisir du texte : `noteTF` (pour saisir la note à ajouter) et `indTF` (pour saisir l'indice de la note à retirer).

3. Le constructeur (à un paramètre entier) comprendra :

- l'initialisation des attributs `maxN` et `ald`
- `setLayout(new GridLayout(4,1));` pour demander une colonne de 4 sous-`JPanel`
- un `JPanel` haut qui comprendra un `JLabel` (pour afficher note), `noteTF` (de largeur 5), et un `JButton` ("ajoute" pour ajouter une note)
- un `JPanel` milieu qui comprendra `listeL` (initialisé à "[]") et `moyL` (initialisé à "= ?")
- un `JPanel` bas qui comprendra un `JLabel` (pour afficher indice), `indTF` (de largeur 2), et un `JButton` ("supprime" pour supprimer une note)
- un `JPanel` `err` qui comprendra `errL` (initialisé à "ok")

4. Compilez tout. Utilisez largement la javadoc pour y parvenir.

3.3 Le fichier `applette.html`

1. Créez un nouveau fichier `applette.html` pour pouvoir lancer l'applette. Y mettre :

```
<html>
  <head>
    <title>CalcApplet</title>
  </head>
  <body>
    <h1>CalcApplet</h1>
    <hr>
    <applet code="CalcApplet.class">
```

```

        width=400 height=200
        codebase="."
        alt="Votre navigateur comprend la balise <APPLET> mais
            n'exécute pas l'applet, pour quelque autre raison."
    >
    <PARAM NAME=max VALUE=20>
    Votre navigateur ignore la balise <APPLET> !
</applet>
</body>
</html>

```

2. Vous pouvez maintenant exécuter l'applette par la commande :
`appletviewer applette.html`
 S'affiche-t-elle correctement ? Que se passe-t-il lorsqu'on clique sur un bouton ?

3.4 La gestion des évènements

1. Modifiez la classe `CalcPanel` pour qu'elle implémente l'interface `ActionListener`.
2. La méthode à écrire est `actionPerformed()` (voir la javadoc). Elle devra distinguer 3 cas : clic sur le bouton `ajoute`, clic sur le bouton `supprime`, ou évènement inconnu. Dans ce dernier cas, il suffit d'écrire `errL.setText("Bouton inconnu !?");` Lorsqu'une exception survient, il suffit d'"afficher" l'exception dans `errL`.
3. Dans le cas `ajoute`, il faut récupérer la note saisie, générer une exception si elle n'est pas dans l'intervalle $[0, \text{max}]$, l'ajouter à la liste, effacer `noteTF`, puis mettre à jour l'affichage (voir la procédure `maj()` ci-dessous).
4. Dans le cas `supprime`, il faut récupérer l'indice saisi, supprimer de la liste la note correspondante, effacer `indTF`, puis mettre à jour l'affichage (voir la procédure `maj()` ci-dessous).
5. La procédure `maj()` doit mettre à jour l'affichage de `listeL`, `moyL` (en faisant appel à la fonction `moyenne()` ci-dessous), et `errL`.
6. La fonction `moyenne()` doit parcourir la liste de notes pour en calculer la moyenne, puis la retourner.
7. Compilez tout. Utilisez largement la javadoc pour y parvenir.
8. Exécutez à nouveau l'applette. Se comporte-t-elle correctement ? Testez tous les cas d'erreur. Comparez ce comportement avec celui de l'applette de référence indiquée au 3.1.1.

3.5 Les classes anonymes

1. La méthode `actionPerformed` précédemment réalisée n'est pas très "OO" car elle fait des tests à chaque clic de bouton alors que l'on pourrait associer à chaque bouton le bon `ActionListener`.
2. Modifiez la classe `CalcPanel` pour qu'elle n'implémente plus l'interface `ActionListener`, donc plus la méthode `actionPerformed`, mais attache à chaque bouton une classe anonyme implémentant l'interface `ActionListener` donc la méthode `actionPerformed`.
3. Pour ne pas avoir à répéter 2 fois le contenu du `catch`, surchargez la procédure `maj()` en lui passant maintenant un paramètre `Exception`.
4. Compilez. Testez. Observez la plus grande modularité du code.