L'3 – Programmation Java

TP Swing

2015-2016

Préambule :

Consignes communes à tous les TP :

- Développement en Java 7 sous Eclipse.
- Sauf exception explicitement autorisée, tous les attributs seront déclarés `private`.
- Utiliser Eclipse pour générer automatiquement des éléments de code classiques (constructeurs, getters, setters, méthodes equals, hashcode, toString(), …), mais aussi les comprendre (il n'y aura pas d'Eclipse disponible en contrôle écrit) !
- Ne pas s'autoriser de "warnings" (avertissements de compilation).
- La bonne exécution du code développé ne suffit pas, encore faut-il qu'il soit proprement conçu et implémenté.
- Tous les exercices doivent être terminés, en ou hors séance.

Quelques sources d'informations en ligne :

- Documentation des classes Java standard : http://docs.oracle.com/javase/7/docs/api/
- Tutoriels Java : http://docs.oracle.com/javase/tutorial/

  Ou plus ciblé : taper sous Google les trois mots Oracle Java7 tutorial suivi – en anglais ! – de un ou quelques mots-clés (par exemple le nom d'une classe ou d'une notion).
- Documentation Eclipse : http://help.eclipse.org

CONSIGNES TRES IMPORTANTES

1) Pour construire une interface graphique, respecter la méthodologie étudiée en cours et mise en oeuvre dans les exemples du cours.

2) Pour la gestion d'évènements, sauf indication contraire, le choix est libre entre la technique des interfaces de type Listener ou celle de la classe AbstractAction.

Exercice 1 : Exemple de JFrame

1) Écrire une classe *SimpleFrame* permettant d'afficher une fenêtre.
On lui donnera simplement un titre et une taille minimale.
2) Changer le comportement par défaut de la fenêtre lors de sa fermeture pour que l'application soit quittée.
3) Dans un second temps, ajouter à cette fenêtre un label, représentant une étiquette (un texte court). Récuperer le conteneur correspondant à l'espace client de la fenêtre avec la méthode *getContentPane*, et ajouter le label à ce conteneur.
4) Centrer le texte du label en utilisant la méthode *setHorizontalAlignment*.
5) L'algorithme de placement des composants de la fenêtre est, pour une *JFrame*, géré par la classe *BorderLayout*. En lisant attentivement la description de la méthode *add(Component, Object)* de la classe *Container*, ajouter un bouton "OK" au dessous du label.

Exercice 2 : A la découverte de nouveaux composants

1) Créer une nouvelle *JFrame* nommée "Découverte" et y inclure les composants suivants :

- un bouton
- un champ de texte
- une zone de texte

2) Ajouter une bulle d'aide contextuelle (*setToolTipText()*) sur le bouton ; rendre le bouton inopérant; changer la couleur du texte et du fond du bouton.
3) Ajouter un texte dans le champ de texte et dans la zone de texte; rendre le champ de texte non éditable.
4) Ajouter une bordure au champ de texte en fournissant un objet type *javax.swing.border.Border* créé via une méthode de la classe *javax.swing.BorderFactory,* par exemple la méthode *createLineBorder(Color color, int thickness)*.

Exercice 3 : Associer des actions aux boutons

Le but de cet exercice est de créer une fenêtre contenant un bouton de label "Test" qui, lorsqu'il reçoit un clic, affiche "Test clic" sur la sortie standard

1) Créer la fenêtre et le bouton.

2) Créer une classe (elle doit être anonyme) implémentant l'interface *ActionListener* et effectuant l'affichage dans la méthode dédiée.

3) Relier le listener au bouton grâce à la méthode du bouton dédiée.

4) Utiliser la même technique avec trois boutons, "Un", "Deux" et "Trois", qui permettent de changer le titre de la fenêtre principale de l'application en respectivement "*Un*", "*Deux*" et "*Trois*". Produire un code qui permette de rajouter facilement un bouton.

Exercice 3bis : Associer des actions aux boutons

Reprendre l'exercice 3, en utilisant la classe *AbstractAction* au lieu de l'interface *ActionListener.*

Exercice 4 : Auditeurs

1) Créer une fenêtre contenant un bouton intitulé "Add". A chaque clic sur ce bouton, il faudra ajouter un bouton dans la fenêtre, les uns à la suite des autres, numérotés à leurs créations. Quand on clique sur l'un d'entre eux, cela le fait disparaître.

2) Ajouter (à la suite de "Add") un bouton "Reset" qui retire tous les boutons ajoutés.

Exercice 5 : Factorielle

Ecrire un programme permettant de calculer la factorielle d'un entier saisi dans un champ texte. La saisie doit être recommencée si l'entier est supérieur ou égale à 17, dans ce cas une boîte de dialogue s'ouvre délivrant un message adéquat à l'utilisateur. Le résultat s'affiche dans un label suite à un clic sur un bouton dont le nom doit pouvoir être changé en "5!" si 5 est l'entier saisi.

Exercice 6 : Mais quel est donc votre age?

I Maquette IHM

En réfléchissant bien aux panneaux à insérer et à leurs gestionnaires de mise en page, réaliser la maquette suivante. Les 3 champs texte en blanc sont éditables, celui en bleu non.

Vos sources figureront dans un **package ageIHM**.

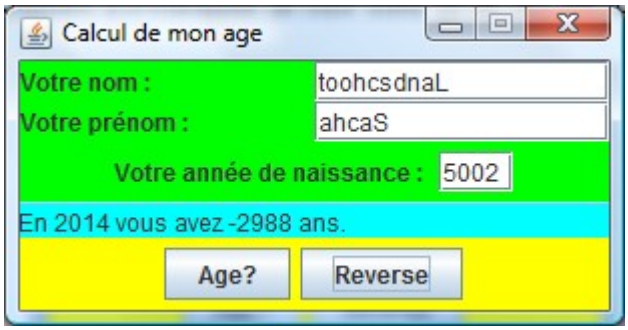
II Gestionnaire d'évènements

Installer des gestionnaires d'évènements sur les boutons.

Vos sources figureront dans un **package calculage**.

1) Bouton Age?

2) Bouton Reverse



aide :

Pour le calcul de l'age affiché dans le champ bleu, l'année en cours doit être retrouvée.

La classe Calendar le permet :

```
Calendar cal = Calendar.getInstance();
int annee = cal.get(Calendar.YEAR);
```

rappel :

La méthode de classe ***int Integer.parseInt(String s) throws NumberFormatException***

permet de convertir une chaîne de caractères numérique en un entier.

Exercice 7 : Composant JTable

1) Ecrire une classe *Etudiant* dont les attributs sont les chaînes de caractères *nom* et le *prénom*.

2) Ecrire une classe *EtudiantLangage* dont les attributs sont un étudiant et une chaîne de caractères dénotant son langage de programmation préféré (par exemple : C, C++, Java …).

3) Ecrire une classe *LangageService* mettant en oeuvre le design pattern Singleton et disposant d'une méthode retournant la liste des langages préférés des étudiants, soit le type *List*<*EtudiantLangage*>. Les étudiants et leur langages de programmation préférés seront entrés en dur dans le programme.

4) Créer une classe *LangageModele* sous-classe de la classe *AbstractTableModel*.

Ce "Table Model" est lié au tableau de données que l'on souhaite afficher. Les entêtes de ce tableau sont le nom et le prénom de l'étudiant ainsi que son langage préféré.

La classe *LangageModele* a 3 attributs :

- *entetes* de type un tableau de chaînes de caractères pour les entêtes
- *langageService* de type *LangageService*
- *etudiantsLangages* de type *List*<*EtudiantLangage*>

La liste des langages préférés des étudiants est à récupérer grâce à l'attribut de type *LangageService*. Une méthode retournant cette liste est à écrire.

Les méthodes *getColumnCount*, *getColumnName*, *getRowCount*, *getValueAt*, *getColumnClass* sont à écrire.

5) Ecrire une application graphique permettant d'afficher le tableau de données dans une JTable.

6) Trier les données par ordre lexicographique des langages préférés.

7) Faire afficher le langage préféré Java en rouge et les autres en bleu.