EPF – Lausanne *COURS DE MATHEMATIQUES SPECIALES*

1 juin 2011

Contrôle d'informatique no 4

Durée : 1 heure 45'

Nom : ……. Groupe :

Prénom : …….

No sujet	**1**	**2**	**3**
Nombre points	35 points	30 points	30 points

Remarque générale : toutes les questions qui suivent se réfèrent au langage de programmation Java (à partir du JDK 5.0) et les réponses doivent être rédigées à l'encre et d'une manière propre sur ces feuilles agrafées.

Remarques initiales :

- Il est conseillé de lire chaque sujet jusqu'à la fin, avant de commencer la rédaction de la solution.
- N'oubliez pas d'importer les packages et les classes standards dont vous avez besoin.
- Afin de centrer horizontalement le texte affiché par un composant graphique de type ***JLabel***, vous pouvez appeler la méthode d'instance ***setHorizontalAlignment*** avec comme argument le champ statique et final ***CENTER*** de l'interface ***SwingConstants*** (du package ***javax.swing***).
- Pour un composant graphique de type ***JLabel***, vous pouvez changer le texte affiché à l'aide de la méthode d'instance ***setText*** et récupérer le texte affiché à l'aide de la méthode d'instance ***getText***.
- Afin de transformer une chaîne de caractères en nombre entier on peut faire un appel à la méthode statique ***parseInt***, tandis que pour transformer un nombre entier en chaîne de caractères on peut faire un appel à la méthode statique ***toString***.
- Au deuxième sujet, faites une bonne utilisation des méthodes ***print*** et ***println***.

1. On considère un projet Java muni d'un package nommé ***cms*** et qui contient deux classes :

- une classe publique appelée ***Fenetre*** dont l'instanciation correspond à l'interface graphique utilisateur (GUI) présentée ci-dessous :

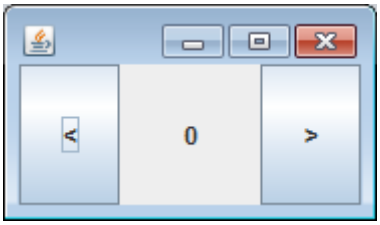

Figure 1

- la classe principale ***CP_Ctr4Exo1*** qui crée une instance de la classe graphique ***Fenetre*** et dont le code source est indiqué ci-dessous.

```
package cms;

public class CP_Ctr4Exo1 {
     public static void main(String[] args)
     {
          new Fenetre();
     }//fin de la méthode main
}//fin de la classe principale
```

Dans ce premier sujet, on vous demande d'écrire le code source de la classe graphique ***Fenetre*** qui fait partie du package ***cms***.

Suite à l'instanciation de cette classe ***Fenetre*** par la méthode ***main*** de la classe principale ***CP_Ctr4Exo1***, une fenêtre (un container graphique de premier niveau) est affichée à l'écran (voir la **Figure 1** ci-dessus). On peut y distinguer :

- un bouton à cliquer dont le texte associé est le signe < (i.e. le signe **strictement plus petit**) et qui est placé dans la région "ouest" ;
- un bouton à cliquer dont le texte associé est le signe > (i.e. le signe **strictement plus grand**) et qui est placé dans la région "est" ;
- une étiquette dont le texte affiché initialement est le chiffre **0** et qui est placée dans la région "centre".

De plus, les deux boutons mentionnés sont réactifs dans le sens où :

- chaque fois que l'utilisateur appuie sur le bouton placé à l'ouest, l'étiquette du centre change d'affichage et indique un nombre entier obtenu en décrémentant d'une unité le nombre entier précédemment affiché ;

- chaque fois que l'utilisateur appuie sur le bouton placé à l'est, l'étiquette du centre change d'affichage et indique un nombre entier obtenu en incrémentant d'une unité le nombre entier précédemment affiché.

A titre d'exemple, on présente ci-dessous l'interface graphique obtenue si, à partir de la situation initiale indiquée dans la **Figure 1**, l'utilisateur clique trois fois de suite sur le bouton placé à l'est :

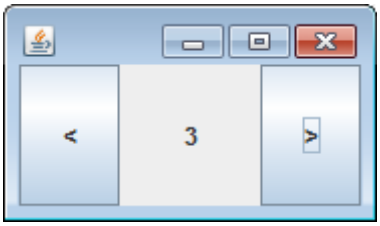

Figure 2

et, ensuite, cinq fois de suite sur le bouton placé à l'ouest :

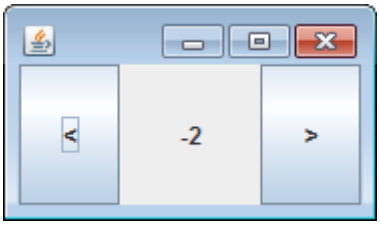

Figure 3

La réalisation du design de l'interface graphique présentée dans la **Figure 1** est détaillée sous forme de commentaires dans le canevas de la classe ***Fenetre*** présenté ci-dessous. Cette classe publique est la fille d'un container graphique swing de premier niveau.

La gestion des événements produits quand l'utilisateur clique avec la souris sur les boutons est assurée par l'instance même de la classe graphique qui implémente une "interface écouteur" appropriée. Par conséquent, la classe graphique doit s'inscrire comme écouteur auprès de ses deux boutons et doit (re)définir la méthode "gestionnaire d'événements".

On présente ci-dessous le squelette de la classe ***Fenetre*** et il faut compléter ce canevas en fonction des indications données en commentaire.

//déclaration de package

.....

.....

//instructions pour importer les packages nécessaires

.....

.....

.....

.....

.....

//en-tête de la classe graphique ***Fenetre*** (qui est publique, hérite d'un container graphique swing de
//premier niveau et implémente une interface lui permettant d'écouter des événements produits par
//des clicks sur des boutons)

.....

.....

.....

{

//déclaration de deux champs privés de type "bouton" swing appelés ***jbOuest*** et ***jbEst***
//(sans valeurs initiales explicites)

.....

.....

//déclaration d'un champ privé de type "étiquette" swing appelé ***jlbCentre***
//(sans valeur initiale explicite)

.....

.....

//en-tête du constructeur public sans argument de la classe graphique

.....

.....

{

//instruction qui assure que la fermeture de la fenêtre graphique produit aussi l'arrêt
//de l'exécution de la JVM

.....

.....

//instruction qui crée un nouvel objet de type "bouton" swing ayant comme texte
//affiché le signe < et stocke son adresse dans le champ ***jbOuest***

.....

.....

//instruction qui indique la taille préférée du bouton ***jbOuest*** sous la forme d'un nouvel
//objet de type ***Dimension*** créé pour l'occasion avec la largeur de 50 pixels et la
//hauteur de 70 pixels

.....

.....

//instruction qui ajoute la fenêtre graphique comme écouteur du bouton ***jbOuest***

.....

.....

//instruction qui ajoute le bouton ***jbOuest*** dans la région "ouest" de la fenêtre

.....

.....

//instruction qui crée un nouvel objet de type "bouton" swing ayant comme texte
//affiché le signe > et stocke son adresse dans le champ ***jbEst***

.....

.....

//instruction qui indique la taille préférée du bouton ***jbEst*** sous la forme d'un nouvel
//objet de type ***Dimension*** créé pour l'occasion avec la largeur de 50 pixels et la
//hauteur de 70 pixels

.....

.....

//instruction qui ajoute la fenêtre graphique comme écouteur du bouton ***jbEst***

.....

.....

//instruction qui ajoute le bouton ***jbEst*** dans la région "est" de la fenêtre graphique

.....

.....

//instruction qui crée un nouvel objet de type "étiquette" swing ayant comme texte
//affiché le chiffre zéro et stocke son adresse dans le champ ***jlbCentre***

.....

.....

//instruction qui centre horizontalement le texte affiché par l'étiquette ***jlbCentre***

.....

.....

//instruction qui indique la taille préférée de l'étiquette ***jlbCentre*** sous la forme d'un
//nouvel objet de type ***Dimension*** créé pour l'occasion avec la largeur de 70 pixels et
//la hauteur de 70 pixels

.....

.....

//instruction qui ajoute l'étiquette ***jlbCentre*** dans la région "centre" de la fenêtre

.....

.....

//instruction qui demande à la fenêtre graphique de s'adapter aux tailles préférées de
//ses composants (tout en prenant la plus petite taille possible)

.....

.....

//instruction qui rend la fenêtre graphique visible à l'écran

.....

.....

}//fin du constructeur

@Override
//en-tête et corps du gestionnaire des événements produits par des clicks sur des boutons
//(rédaction selon les indications données auparavant)

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

}//fin de la classe graphique

2. Dans le sujet numéro 2, on vous demande d'écrire une application Java autonome qui lit un fichier texte donné, modifie le contenu lu et crée un nouveau fichier texte qui contient le texte avec les modifications (voir les exemples qui suivent).
On donne ci-dessous, à titre d'exemple, le contenu d'un fichier texte ***citations.txt*** qui doit être lu.

```
Bonjour, voilà une ligne qui ne contient aucune citation.

     Une ligne entre deux lignes vide.

     Une "citation", ensuite une "deuxième" et on s'arrête.
     "Le début" pour commencer et "Le milieu" pour continuer.
     On ne commence pas bien mais "On finit bien !"
"Bien commencer" et "Bien continuer", c'est "Bien finir !"

          Au revoir !
```

Il convient de remarquer que, dans ce fichier texte à lire :

- une ligne peut contenir aucune, une ou plusieurs citations ;
- chaque citation est délimitée par une paire de guillemets ;
- il n'y a pas de citation qui s'étale sur deux ou plusieurs lignes ;
- une ligne peut commencer et/ou finir par des **caractères blancs** (comme les espaces ou les tabulations) ;
- une citation peut se trouver au début, à l'intérieur ou à la fin d'une ligne.

Suite à l'exécution du programme que vous devez concevoir, un nouveau fichier texte ***mesCitations.txt*** sera créé et il aura le contenu suivant :

```
Bonjour, voilà une ligne qui ne contient aucune citation.

     Une ligne entre deux lignes vide.

Une <<citation>>, ensuite une <<deuxième>> et on s'arrête.
<<Le début>> pour commencer et <<Le milieu>> pour continuer.
On ne commence pas bien mais <<On finit bien !>>
<<Bien commencer>> et <<Bien continuer>>, c'est <<Bien finir !>>

          Au revoir !
```

Plus précisément, il s'agit d'un projet Java muni d'un package nommé ***cms*** et qui contient une seule classe (principale), publique et appelée ***CP_Ctr4Exo2***. Dans la méthode ***main*** de cette classe, le programme doit :

- lire, ligne après ligne, le fichier texte ***citations.txt*** qui se trouve dans le dossier ***temp*** situé sur un disque Windows identifié par ***C:*** ;
- modifier, le cas échéant, le contenu lu ligne après ligne :
    - en éliminant tous les éventuels **caractères blancs** situés au début et/ou à la fin de chaque ligne qui contient au moins une citation ;
    - en remplaçant le caractère guillemet au début de chaque citation par deux caractères successifs << (c'est-à-dire deux fois **strictement plus petit**) ;
    - en remplaçant le caractère guillemet à la fin de chaque citation par deux caractères successifs >> (c'est-à-dire deux fois **strictement plus grand**) ;
- écrire les lignes lues, à la fois celles laissées inchangées et celles modifiées, les unes après les autres, dans un nouveau fichier texte nommé ***mesCitations.txt*** qui sera créé dans le même dossier que celui qui contient le fichier initial ***citations.txt***.

Astuces :

- afin de savoir si une ligne contient au moins une citation, il suffit de tester si la ligne contient, au moins une fois, le caractère guillemet ;
- chaque ligne contenant au moins une citation peut être "nettoyée" en supprimant les éventuels **caractères blancs** situés au début et/ou à la fin de la ligne par un appel à la méthode d'instance ***trim*** ;
- afin de savoir si une ligne "nettoyée" des caractères blancs commence par une citation, il suffit de tester si le premier caractère est bien le caractère guillemet.

Indications :

- le fichier texte initial ***citations.txt*** sera lu grâce à une boucle englobante ;
- si la ligne lue ne contient pas de citation, elle sera écrite telle quelle dans le nouveau fichier texte ***mesCitations.txt*** ;
- autrement, c'est-à-dire si une ligne lue contient au moins une citation :
    - elle sera d'abord "nettoyée" des éventuels **caractères blancs** apparaissant au début ou à la fin de la ligne ;
    - elle sera ensuite séparée en tokens adéquats (par une opération de "tokenisation" en fonction d'un séparateur convenablement choisi) ;
    - le nombre de tokens sera déterminé ;
    - deux cas seront traités en fonction du fait que la ligne commence ou non par une citation (voir la ***Remarque*** présentée ci-dessous) ;

- pour chaque cas mentionné, chaque token sera lu grâce à une nouvelle boucle (interne à la boucle englobante) ;
- si un token correspond à une citation, il sera écrit dans le fichier ***mesCitations.txt*** précédé et suivi par les nouveaux marqueurs de citations ;
- autrement, le token sera écrit tel quel dans le fichier ***mesCitations.txt***.

Remarque :

- si une ligne qui contient au moins une citation et qui a été nettoyée et tokenisée correctement commence par une citation, le premier token lu correspond à une citation, le suivant à du texte "normal", le suivant à une citation et ainsi de suite ;
- autrement, le premier token lu correspond à du texte "normal", le suivant à une citation, le suivant à du texte "normal" et ainsi de suite.

Vous devez écrire ci-dessous le contenu du fichier ***CP_Ctr4Exo2.java*** (à savoir : la déclaration de package, les éventuelles instructions "***import***", ainsi que l'en-tête et le corps de la classe principale publique ***CP_Ctr4Exo2***). La solution que vous allez proposer doit utiliser le mécanisme de "tokenisation".

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

3. Soit le fichier ***CP_Ctr4Exo3.java*** contenant le code source suivant :

```
package cms;

import java.awt.*;    import javax.swing.*;        import java.awt.event.*;

class TropFonceeExp extends Exception
{       Color champ;
        TropFonceeExp(Color arg)
        {       super("est trop foncée !");
                champ = arg;
        }
}//fin de la classe TropFonceeExp

class TropClaireExp extends RuntimeException
{       Color champ;
        TropClaireExp(Color arg)
        {       super("a la couleur trop claire !");
                System.out.println("Quelle couleur !");
                champ = arg;
        }
}//fin de la classe TropClaireExp

public class CP_Ctr4Exo3 implements KeyListener
{
        static JFrame fen;
        static JPanel tab[ ] = new JPanel[4];
        static Color couleurs[ ] = {Color.WHITE, Color.RED, Color.GREEN, Color.BLACK};

        public static void main(String[ ] args)
        {       fen = new JFrame( );
                fen.setLayout(new GridLayout(2,2));
                for(int i = 0; i < 4; i++)
                {       tab[i] = new JPanel( );
                        tab[i].setPreferredSize(new Dimension(70, 70));
                        tab[i].setBackground(couleurs[i]);
                        fen.add(tab[i]);
                }
                fen.addKeyListener(new CP_Ctr4Exo3( ));
                fen.pack( );
                fen.setVisible(true);
        }//fin de la méthode main

        private static void critiquer(Color coul,  String pos) throws TropFonceeExp
        {       try
                {       if(coul == Color.BLACK || coul == Color.RED)
                                throw new TropFonceeExp(coul);
                        else if(coul == Color.WHITE || coul == Color.GRAY)
                                throw new TropClaireExp(coul);
                        System.out.println("Le panneau situé " + pos + " est parfait !");
```

```
            }catch(TropFonceeExp e)
            {       if(e.champ == Color.BLACK)
                        System.out.println("Le panneau situé " + pos + e.getMessage());
                    else
                        throw e;
            }catch(TropClaireExp e)
            {       System.out.println("Le panneau situé " + pos + e.getMessage());
            }finally
            {       System.out.println("Le bloc \"Finally\" de la méthode critiquer !");
            }
            System.out.println("On cesse de critiquer !");
        }//fin de la méthode critiquer

        @Override
        public void keyPressed(KeyEvent ev)
        {       try
            {       switch( ev.getKeyChar( ) )
                    {       case '0' :  critiquer(tab[0].getBackground( ), "au nord-ouest ");
                                break;

                            case '1' :  critiquer(tab[1].getBackground( ), "au nord-est ");
                                break;

                            case '2' :  critiquer(tab[2].getBackground( ), "au sud-ouest ");
                                break;

                            case '3' :  critiquer(tab[3].getBackground( ), "au sud-est ");

                            default : critiquer(Color.GRAY, "nulle part ");
                    }
            }catch(TropFonceeExp e)
            {       System.out.println("Vraiment, c'est trop foncé !");
            }finally
            {       System.out.println("C'est la fin !");
                    System.out.println("**************************");
            }
        }//fin de la méthode keyPressed

        @Override
        public void keyReleased(KeyEvent arg0) { }
        @Override
        public void keyTyped(KeyEvent arg0) { }
}//fin de la classe principale
```

Suite à l'exécution de la méthode ***main*** de la classe principale ***CP_Ctr4Exo3*** par la machine virtuelle Java (JVM), l'utilisateur dispose d'une interface graphique (GUI) comme celle montrée dans la figure ci-dessous, où le panneau placé au nord-ouest a la couleur de fond blanche (WHITE), celui placé au nord-est rouge (RED), celui placé au sud-ouest verte (GREEN) et celui placé au sud-est noire (BLACK) :

Indiquer quels seront les messages affichés dans la fenêtre console si, juste après l'affichage de la GUI suite (chaque fois) à une nouvelle exécution de la classe ***CP_Ctr4Exo3***, l'utilisateur clique sur une touche ou une combinaison de touches correspondant :

- **a)** au chiffre **0** ;
- **b)** au chiffre **1** ;
- **c)** au chiffre **2** ;
- **d)** au chiffre **3** ;
- **e)** au caractère **$**.

Tous les cas ci-dessus, notés de **a)** à **e)**, sont indépendants et chacune de vos 5 réponses doit indiquer clairement la lettre qui lui correspond.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....