EPF – Lausanne *COURS DE MATHEMATIQUES SPECIALES*

5 juin 2013

Contrôle d'informatique no 4

Durée : 1 heure 45'

Nom : ……. Groupe :

Prénom : …….

No sujet	**1**	**2**	**3**
Nombre points	45 points	95 points	20 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 et rappels :

- Il est conseillé de lire chaque sujet jusqu'à la fin, avant de commencer la rédaction de la solution.
- Les couleurs indiquées dans le sujet numéro **2.** sont représentées en nuances de gris dans les copies d'écran données comme exemples dans les figures.
- Le cycle de vie d'une applet est géré grâce à quatre méthodes standards appelées automatiquement par l'environnement d'exécution (call-back methods) : ***init()***, ***start()***, ***stop() et destroy()***.
- Afin de savoir quel objet a généré un certain événement, on peut appeler la méthode ***getSource()*** pour cet événement.
- Une fenêtre de type applet swing est un container de premier niveau.

1. Soit le fichier ***CP_Ctr4Exo1.java*** contenant le code source suivant :

```
package cms_ctr4;

class TropBasException extends Exception
{
	String noteBasse;
	TropBasException(String note)
	{
		super("Fréquence basse !");
		this.noteBasse = note;
		System.out.println("Une note trop basse !");
	}
}

class TropHautException extends RuntimeException
{
	String noteHaute;
	TropHautException(String note)
	{
		super("Fréquence haute !");
		this.noteHaute = note;
		System.out.println("Une note trop haute !");
	}
}

public class CP_Ctr4Exo1
{
	static String jouer(String note) throws TropBasException
	{
		try
		{
			if(note.equals("do") || note.equals("re"))
			{
				System.out.println("Quel début !");
				throw new TropBasException(note);
			}
			else if(note.equals("la") || note.equals("si"))
			{
				throw new TropHautException(note);
			}
			else
			{
				System.out.println("OK");
				return note;
			}
		}
```

```
            catch(TropBasException e)
            {
                  System.out.println(e.getMessage());
                  if(e.noteBasse.equals("do"))
                  {
                        return "doDièse";
                  }
                  else
                  {
                        throw e;
                  }
            }finally
            {
                  System.out.println("La note finale !");
            }
      }//fin de la méthode jouer

      public static void main(String[] args)
      {
            String notes = "mi-re-la-fa-si-do";

            for(int i=0; i<notes.length(); i=i+3)
            {
                  try
                  {
                        System.out.println(jouer(
                                    notes.substring(i, i+2)));
                  }catch(TropBasException tbe)
                  {
                        System.out.println(tbe.getMessage());
                        System.out.println(tbe.noteBasse + "Bécarre");
                  }catch(TropHautException the)
                  {
                        System.out.println(the.getMessage());
                        System.out.println(the.noteHaute + "Bémol");
                  }
                  System.out.println("**************************");

            }
            System.out.println("Au revoir !");
      }//fin de la méthode main
}//fin de la classe principale
```

Préciser les messages qui seront affichés à l'écran suite à l'exécution du projet correspondant au code source indiqué ci-dessus.

2. On considère un projet Java muni d'un package nommé ***cms_ctr4*** et qui contient une seule classe publique appelée ***Chrono*** dont l'instanciation dans l'AppletViewer (le visualiseur d'applets) de l'environnement de programmation intégré Eclipse correspond initialement à l'interface graphique utilisateur (GUI) de type applet swing présentée ci-dessous :

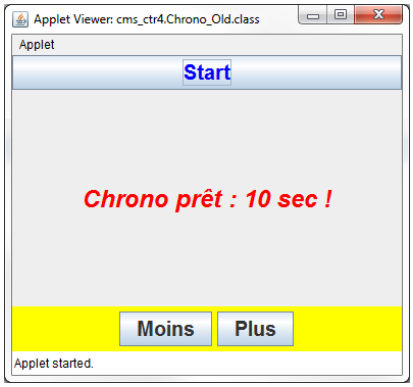

Figure 1

Dans ce deuxième sujet, on vous demande d'écrire le code source de cette classe graphique. On ne vous demande pas d'écrire le fichier ***.html*** qui sert à télécharger et à lancer l'applet swing dans un navigateur.

Suite à l'instanciation de la classe ***Chrono*** par l'AppletVewer, une fenêtre (un container graphique de premier niveau) de type applet swing est affichée à l'écran (voir la **Figure 1** ainsi que les autres figures ci-dessous). On peut y distinguer :

- un bouton poussoir (à cliquer) swing placé dans la partie haute de la fenêtre et dont le texte associé est ***Start*** ;
- une étiquette swing placée au centre de la fenêtre et qui affiche initialement (avec une police précisée plus loin) le texte ***Chrono prêt : 10 sec !*** ;
- un conteneur swing intermédiaire placé dans la partie basse de la fenêtre, qui a la couleur de fond jaune et qui contient deux boutons poussoir (à cliquer) swing dont les textes associés sont ***Moins*** et, respectivement, ***Plus***.

L'applet swing décrite ci-dessus correspond à une sorte de chronomètre (un compte à rebours) qui doit fonctionner ainsi :

a. tout au début et après chaque intervalle chronométré, la GUI doit être affichée comme dans la **Figure 1** et doit annoncer une durée à chronométrer proposée par défaut de 10 secondes ;

b. quand le chronomètre est arrêté, avec chaque clic sur les boutons ***Plus*** ou ***Moins***, l'utilisateur peut augmenter ou, respectivement, diminuer d'une seconde la durée à chronométrer et l'affichage doit changer en conséquence et devenir ***Chrono prêt : m sec !*** où ***m*** est le nombre modifié de secondes à chronométrer (voir **Figure 1.bis** ci-dessous) ;

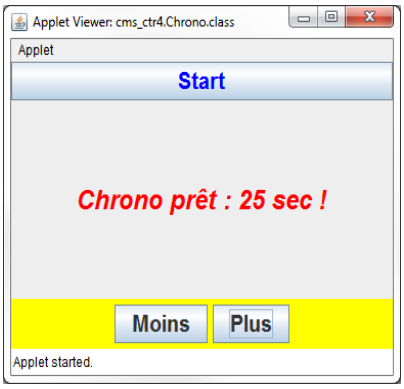

Figure 1.bis

c. quand le chronomètre est en marche, les clics sur les boutons ***Plus*** et ***Moins*** ne doivent pas avoir d'effet ;

d. le même comportement décrit aux points b) et c) doit pouvoir être obtenu à l'aide du clavier, en appuyant les touches "***Flèche en haut***" ou, respectivement, "***Flèche en bas***" ;

e. le chronomètre peut être démarré en cliquant sur le bouton ***Start*** ;

f. une fois démarré le chronomètre, le message central change chaque seconde (voir la **Figure 2** ci-dessous) et affiche ***Chrono en marche : n sec !***, où ***n*** est un nombre entier (de plus en plus petit) qui correspond au nombre de secondes restantes ;

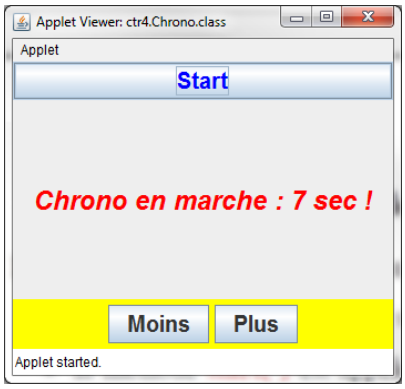

Figure 2

g. à la fin de la durée chronométrée, c'est-à-dire quand le nombre de secondes devient 0, le message affiché pendant une seconde devient ***Game over : 0 sec !*** (voir la **Figure 3** ci-dessous)

Figure 3

et, ensuite, le message redevient ***Chrono prêt : 10 sec !*** (comme dans la **Figure 1**) ;

h. de plus:

- quand la fenêtre de l'applet swing n'est plus visible (par exemple, quand elle est iconifiée), le chronomètre doit s'arrêter automatiquement (sans l'intervention de l'utilisateur) ;
- quand la fenêtre de l'applet swing redevient visible (par exemple, quand elle est désiconifiée), le chronomètre doit redémarrer automatiquement (sans l'intervention de l'utilisateur).

Afin de réaliser le "mécanisme du chronomètre", vous pouvez utiliser un objet Java d'un type adéquat. Cet objet envoie périodiquement (dans ce cas concret, à chaque seconde) un événement de type ***ActionEvent*** vers le conteneur de premier niveau qui, grâce à la méthode "gestionnaire d'événements" adéquate, change l'affichage central.

La gestion des événements produits quand l'utilisateur clique avec la souris sur les boutons ou appuie sur les touches du clavier est assurée par l'instance même de la classe graphique qui implémente des "interfaces écouteur" appropriées. Par conséquent, la classe graphique doit s'inscrire à la fois comme écouteur des boutons et comme écouteur du clavier auprès de ses trois boutons et doit (re)définir les méthodes "gestionnaires d'événements" appropriées.

Afin d'écrire le code source de la classe ***Chrono*** (qui est accessible partout, correspond à une applet swing et est capable de gérer les événements produits par les boutons et le moteur du chronomètre ainsi que par les touches du clavier) vous devez respecter les consignes suivantes :

A. déclarer le package qui contient la classe ***Chrono*** ;

B. importer les packages contenant les classes et les interfaces standard nécessaires ;

C. préciser l'en-tête complet de la classe ***Chrono*** ;

D. déclarer les champs privés suivants :

1. un champ appelé ***etiquette*** de type étiquette swing, sans valeur initiale explicite;
2. un champ appelé ***panneau*** de type conteneur intermédiaire swing, sans valeur initiale explicite ;
3. trois champs appelés ***boutonPlus***, ***boutonMoins*** et ***boutonStart*** de type bouton poussoir swing, sans valeurs initiales explicites ;
4. un champ appelé ***moteur*** d'un type approprié, sans valeur initiale explicite et qui représente le "mécanisme du chronomètre";
5. un champ appelé ***nbSecondes*** de type primitif entier, avec la valeur initiale explicite ***10*** et qui sert à stocker le nombre courant de secondes du chronomètre ;
6. un champ appelé ***enMarche***, de type primitif booléen, avec la valeur initiale correspondant à "faux" et qui sert à savoir si le chronomètre est arrêté ou en marche ;

E. redéfinir (en précisant son en-tête et son corps) la méthode qui pour toute applet est appelée une seule fois, immédiatement après la création de l'applet ; cette méthode précise le design de la GUI et fait les liens entre les éléments graphiques écoutés et les écouteurs correspondants ; procéder ainsi :

1. créer un objet de type étiquette swing (avec ***Chrono prêt : 10 sec !*** comme texte initial affiché) et stocker son adresse dans le champ ***etiquette*** ; l'affichage doit être fait avec une police de la famille ***Arial***, en ***italique*** et ***gras*** et avec la taille 24 pixels ; le texte doit être centré horizontalement et apparaître en rouge (en anglais red) ; placer cette étiquette dans la zone centrale du container de premier niveau ;
2. créer un objet de type bouton poussoir swing (avec ***Start*** comme texte associé) et stocker son adresse dans le champ ***boutonStart*** ; placer ce bouton dans la partie supérieure du container de premier niveau ; inscrire le container de premier niveau comme écouteur des événements de type ***ActionEvent*** et des événements de type "touche clavier" auprès de ce bouton (par deux instructions à part) ;
3. créer un objet de type container intermédiaire et stocker son adresse dans le champ ***panneau*** ; la couleur de fond de ce container doit être jaune (en anglais

yellow); placer ce container intermédiaire dans la partie inférieure du container de premier niveau ;

4. créer un objet de type bouton poussoir swing (avec ***Moins*** comme texte associé) et stocker son adresse dans le champ ***boutonMoins*** ; placer ce bouton dans le container intermédiaire ***panneau*** ; inscrire le container de premier niveau comme écouteur des événements de type ***ActionEvent*** et des événements de type "touche clavier" auprès de ce bouton (par deux instructions à part) ;
5. créer un objet de type bouton poussoir swing (avec ***Plus*** comme texte associé) et stocker son adresse dans le champ ***boutonPlus*** ; placer ce bouton dans le container intermédiaire ***panneau*** ; inscrire le container de premier niveau comme écouteur des événements de type ***ActionEvent*** et des événements de type "touche clavier" auprès de ce bouton (par deux instructions à part) ;
6. créer un objet de type approprié pour qu'il soit le "mécanisme du chronomètre" et stocker son adresse dans le champ **moteur** ; cet objet doit être capable d'envoyer des événements de type ***ActionEvent*** au container de premier niveau avec une fréquence de 1 Hz (c'est-à-dire à des intervalles réguliers d'une seconde) ;

F. redéfinir (en précisant son en-tête et son corps) la méthode gestionnaire des événements de type ***ActionEvent*** pour qu'elle assure le fonctionnement suivant :

1. si le "bouton ***Plus***" est appuyé et le chronomètre est arrêté (c'est-à-dire le champ ***enMarche*** vaut "faux"), la valeur du champ ***nbSecondes*** est incrémentée d'une unité et le texte affiché par ***etiquette*** change de manière appropriée ;
2. autrement, si le "bouton ***Moins***" est appuyé et le chronomètre est arrêté (c'est-à-dire le champ ***enMarche*** vaut "faux"), la valeur du champ ***nbSecondes*** est décrémentée d'une unité et le texte affiché par ***etiquette*** change de manière appropriée ;
3. autrement, si le "bouton ***Start***" est appuyé et le chronomètre est arrêté (c'est-à-dire le champ ***enMarche*** vaut "faux"), le mécanisme du chronomètre (représenté par le champ ***moteur***) est démarré et la valeur du champ ***enMarche*** est changée à "vrai" ;

4. autrement, si l'événement a été généré par le mécanisme du chronomètre :
    i. si la valeur du champ ***nbSecondes*** n'est pas encore 1, la valeur du champ ***nbSecondes*** est décrémentée d'une unité et le texte affiché par le champ ***etiquette*** change de manière appropriée (voir la **Figure 2**) ;
    ii. autrement, si la valeur du champ ***nbSecondes*** vaut 1, la valeur du champ ***nbSecondes*** est décrémentée d'une unité et le texte affiché par le champ ***etiquette*** change de manière appropriée (voir la **Figure 3**) ;
    iii. autrement, le mécanisme du chronomètre (représenté par le champ ***moteur***) est arrêté, le champ ***enMarche*** est remise à "faux", le champ ***nbSecondes*** est remis à 10 et le texte affiché par le champ ***etiquette*** change de manière appropriée (voir la **Figure 1**) ;

G. redéfinir les trois méthodes gestionnaires des événements "touche clavier" de sorte qu'une d'entre elles assure le fonctionnement suivant :

1. si la touche du clavier "***Flèche en haut***" (dont le code virtuel est ***VK_UP***) est appuyée et le champ ***enMarche*** vaut "faux", la valeur du champ ***nbSecondes*** est incrémentée d'une unité et le texte affiché par ***etiquette*** change de manière appropriée;
2. autrement, si la touche du clavier "***Flèche en bas***" (dont le code virtuel est ***VK_DOWN***) est appuyée et le champ ***enMarche*** vaut "faux", la valeur du champ ***nbSecondes*** est décrémentée d'une unité et le texte affiché par ***etiquette*** change de manière appropriée;

H. redéfinir (en précisant son en-tête et son corps) la méthode qui est appelée automatiquement quand une applet n'est plus visible de sorte qu'elle arrête le chronomètre (représenté par le champ ***moteur***) si celui-ci est en marche (c'est-à-dire, si le champ ***enMarche*** vaut "vrai") ;

I. redéfinir (en précisant son en-tête et son corps) la méthode qui est appelée automatiquement quand une applet redevient visible de sorte qu'elle redémarre le chronomètre (représenté par le champ ***moteur***) si celui-ci est arrêté (c'est-à-dire, si le champ ***enMarche*** vaut "faux").

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

```
//déclaration du package
……
……
……
//importations des packages
……
……
……
……
……
……
……
//en-tête de la classe Chrono
……
……
……
……
……
{     //début du corps de la classe Chrono
      //déclarations des champs (voir le point D. des consignes)
……
……
……
……
……
……
……
……
……
……
……
      //en-tête de la méthode précisée au point E.
……
……
……
      {     //début du corps de la méthode précisée au point E.
```

//le point **E.1.** (le champ ***etiquette***)

//le point **E.2** (le champ ***boutonStart***)

//le point **E.3.** (le champ ***panneau***)

//le point **E.4.** (le champ ***boutonMoins***)

//le point **E.5** (le champ ***boutonPlus***)

……

……

……

……

……

……

……

……

//le point **E.6.** (le champ ***moteur***)

……

……

……

……

……

}//fin du corps de la méthode précisée au point **E.**

//en-tête et corps de la méthode précisée au point **F.**

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

//en-têtes et corps des trois méthodes précisées au point **G.**

```
    //en-tête et corps de la méthode précisée au point H.
```
……

……

……

……

……

……

……

……

……

```
    //en-tête et corps de la méthode précisée au point I.
```
……

……

……

……

……

……

……

……

……

```
}//fin de la classe Chrono
```

Ceci n'est pas la dernière page du Contrôle. Voir la page suivante, s'il vous plaît.

3. Indiquer si les affirmations suivantes sont vraies ou fausses (par VRAI ou FAUX à l'endroit prévu à cet effet) :

a) Un gestionnaire d'exception (un bloc **catch**) qui reçoit en paramètre et traite une exception d'un type classe **UneException** peut lancer (par exemple, à l'aide d'une instruction **throw**) une exception d'un type classe **AutreException**.

.....

b) Le gestionnaire de disposition **BorderLayout** ne peut être utilisé que pour la mise en forme des containers de premier niveau.

.....

c) Tous les composants Swing sont des composants légers (lightweight components).

.....

d) Il est possible qu'un objet graphique soit à la fois la source d'un certain événement et l'écouteur d'un autre événement.

.....

e) Tout fichier HTML contient au moins une balise (tag) **<APPLET>**.

.....

Remarque : chaque réponse correcte vaut +4 points, chaque réponse fausse vaut –2 points et chaque question sans réponse vaut 0 point.