EPF – Lausanne *COURS DE MATHEMATIQUES SPECIALES*

3 juin 2015

Contrôle d'informatique no 4

Durée : 1 heure 45'

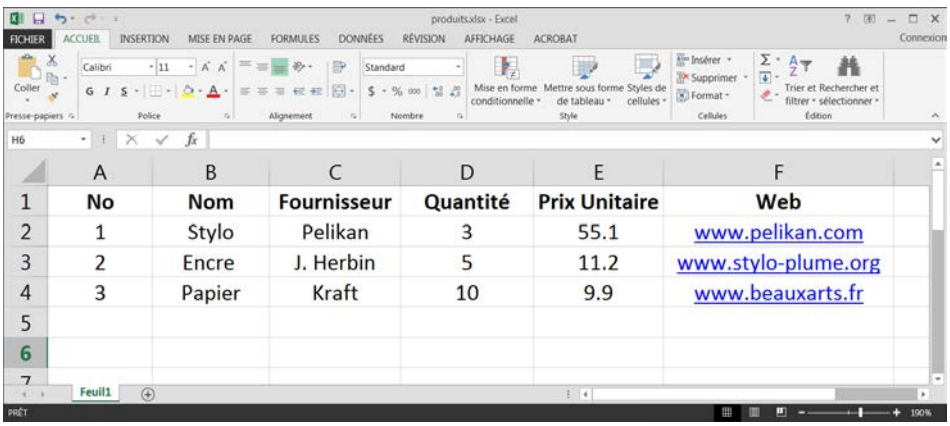
Nom : ……. Groupe :

Prénom : …….

No sujet	**1**	**2**
Nombre points	60 points	(52 + 45) 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.

1. La plupart des tableurs (comme le logiciel Excel de la suite Microsoft Office) permettent de sauver les feuilles de calcul dans un format appelé **csv** (**C**omma-**S**eparated **V**alues). Plus précisément, un tel fichier **csv** est un fichier texte dans lequel chaque ligne d'une feuille de calcul d'origine est transformée en une ligne de texte où les contenus des cellules se retrouvent sans mise en forme, mais dans le "bon ordre" et séparés par des points-virgules.
Prenons l'exemple du fichier Excel **produits.xlsx** ci-dessous :



En sauvant le fichier **produits.xlsx** en format **csv**, on obtient le fichier texte **produits.csv** dont le contenu est reproduit ci-dessous :

```
No;Nom;Fournisseur;Quantité;Prix Unitaire;Web
1;Stylo;Pelikan;3;55.1;www.pelikan.com
2;Encre;J. Herbin;5;11.2;www.stylo-plume.org
3;Papier;Kraft;10;9.9;www.beauxarts.fr
```

Le but du sujet numéro 1 est de concevoir une application Java autonome qui :

- lit un fichier texte en format **csv** donné, à savoir le fichier **produits.csv** ;
- utilise certaines informations lues afin de calculer de nouvelles valeurs ;
- crée un nouveau fichier texte en format **csv**, à savoir **produitsFinaux.csv**, dans lequel elle écrit les informations de départ complétées par les nouvelles valeurs calculées (voir l'exemple qui suit).

Plus précisément, l'application Java doit calculer et ajouter dans le nouveau fichier le prix total de chaque produit, ce qui donne un fichier texte **produitsFinaux.csv** avec le contenu suivant :

```
No;Nom;Fournisseur;Quantité;Prix Unitaire;Web;Prix total
1;Stylo;Pelikan;3;55.1;www.pelikan.com;165.3
2;Encre;J. Herbin;5;11.2;www.stylo-plume.org;56.0
3;Papier;Kraft;10;9.9;www.beauxarts.fr;99.0
```

Il convient de préciser que le fichier texte à lire doit avoir le nom **produits.csv** et qu'il :

- se trouve dans le dossier **temp** situé sur un disque Windows identifié par **C:** ;
- a un nombre total de lignes qui n'est pas connu par le programmeur Java ;
- contient au moins une première ligne.

De plus, il faut tenir compte des consignes suivantes :

- la première ligne du fichier **produits.csv** correspond aux noms des colonnes dans le fichier Excel de départ ;
- le nombre, la succession et les noms de toutes les colonnes du fichier Excel de départ ne sont pas connus par le programmeur Java ;
- cependant, le programmeur sait que le fichier Excel de départ contenait une colonne dont le nom est **`Quantité`** et une colonne dont le nom est **`Prix Unitaire`** (mais ne connaît pas les positions exactes de ces deux colonnes).

Le fichier texte produit par l'application Java doit avoir le nom **produitsFinaux.csv** et il doit :

- être créé dans le même dossier **temp** situé sur le disque Windows identifié par **C:** ;
- avoir le même nombre de lignes que le fichier lu **produits.csv**.

De plus, chaque ligne du fichier **produitsFinaux.csv** correspond à une ligne du fichier **produits.csv** à la fin de laquelle le programme Java ajoute une information supplémentaire précédée par un point-virgule, à savoir :

- sur la première ligne, le nom d'une nouvelle colonne **`Prix total`** ;
- sur les lignes suivantes, la valeur du prix total de chaque produit obtenu en multipliant le nombre d'unités du produit (correspondant à la colonne **`Quantité`**) avec le prix unitaire du produit (correspondant à la colonne **`Prix Unitaire`**).

En conclusion, il s'agit d'un projet Java muni d'un package nommé ***cms_ctr4*** et qui contient une seule classe principale, publique et appelée ***CP_Ctr4Exo1***. Dans la méthode ***main*** de cette classe, le programme doit lire, ligne après ligne, le fichier texte **produits.csv**, traiter les informations lues et écrire, au fur et à mesure, les lignes du nouveau fichier texte **produitsFinaux.csv**.

Indications :

- chaque ligne lue dans le fichier **produits.csv** doit être séparée en tokens adéquats (par une opération de "tokenisation" en fonction d'un séparateur convenablement choisi) ;
- la lecture de la première ligne permet d'identifier et de stocker dans des variables prévues à cet effet, appelées par exemple ***positionQuantite*** et ***positionPrix***, les "positions" des tokens correspondant aux colonnes **`Quantité`** et **`Prix Unitaire`** ;
- pour chaque nouvelle ligne lue, c'est-à-dire pour chaque produit :
    - le nombre d'unités du produit correspond au token de "position" ***positionQuantite*** ;
    - le prix unitaire du produit correspond au token de "position" ***positionPrix*** ;
- afin de faire des calculs mathématiques (dans notre cas des multiplications), les opérandes doivent être des nombres réels (et non pas des chaînes de caractères).

Vous devez écrire ci-dessous le contenu du fichier ***CP_Ctr4Exo1.java*** (à savoir : la déclaration de package, les éventuelles instructions d'importations des packages prédéfinis, ainsi que l'en-tête et le corps de la classe principale publique ***CP_Ctr4Exo1***). La solution que vous allez proposer doit utiliser le mécanisme de "tokenisation".

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

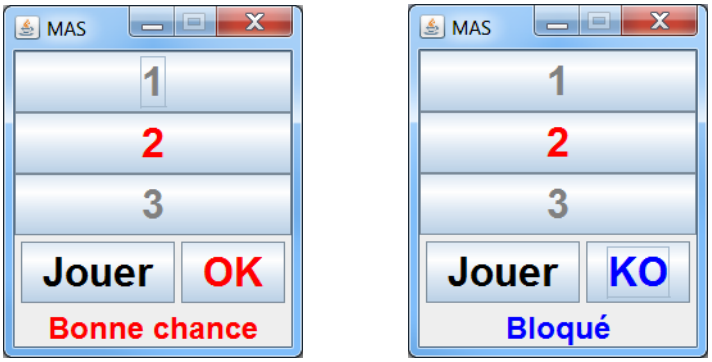
.....

.....

2. Le but de cet exercice est de permettre à l'utilisateur d'interagir avec une interface graphique qui peut être vue comme une machine à sous avec une seule "colonne".

Plus précisément, au début de l'exécution du projet à réaliser, une interface graphique comme celle présentée dans la **Figure 2.1.a)**) doit apparaître au coin supérieur gauche de l'écran. Cette fenêtre doit avoir comme titre le texte **MAS** et doit respecter les consignes suivantes :

- les chiffres ***1*** et ***3*** apparaissent en gris (***gray*** en anglais) ;
- le chiffre ***2*** et les textes ***OK*** et ***Bonne chance*** apparaissent en rouge (***red*** en anglais) ;
- le texte ***Jouer*** apparaît en noir (***black*** en anglais) ;
- les trois chiffres ainsi que les textes ***Jouer*** et ***OK*** sont écrits avec une police ***Arial***, grasse (***bold*** en anglais) et de taille 32 ;
- le texte ***Bonne chance*** est écrit avec une police ***Arial***, grasse et de taille 24.



a) b)

Figure 2.1.

Quand l'utilisateur appuie sur le bouton de droite qui affiche initialement ***OK*** :

- si le texte affiché est ***OK*** rouge :
  - ce texte change en ***KO*** bleu (***blue*** en anglais) ;
  - le texte affiché par l'étiquette tout en bas devient ***Bloqué*** bleu (voir la **Figure 2.1.b)**) ;
  - de plus, si les trois chiffres étaient en train de changer (comme expliqué ci-dessous), tout changement cesse ;
- si le texte affiché est ***KO*** bleu :
  - ce texte change en ***OK*** rouge ;
  - le texte affiché par l'étiquette tout en bas devient ***Bonne chance*** rouge (voir la **Figure 2.1.a)**).

Quand l'utilisateur appuie sur le bouton de gauche qui affiche ***Jouer*** :

- si le bouton à sa droite affiche ***KO*** (bleu), rien ne se passe ;

- par contre, si le bouton à sa droite affiche ***OK*** (rouge), les chiffres affichés au-dessus changent simultanément un nombre aléatoire de fois et de "manière circulaire", c'est-à-dire : chaque chiffre augmente chaque fois d'une unité et s'il arrive à 9, il repasse ensuite à 0, augmente à nouveau d'une unité et ainsi de suite ; au début, les trois chiffres affichés se succèdent et cette propriété reste valable aussi par la suite (dans la mesure où on accepte que le successeur de ***9*** est bien ***0***) ;
- de plus, quand le changement des chiffres s'arrête :
    - si le chiffre du milieu (c'est-à-dire en deuxième position) est impair, le texte affiché par l'étiquette tout en bas change en ***Gagné !*** rouge (voir la **Figure 2.2.a)**) ;
    - si le chiffre du milieu est pair, le texte affiché par l'étiquette tout en bas change en ***Perdu !*** bleu (voir la **Figure 2.2.b)**).

a) b)

Figure 2.2.

On considère un projet Java qui implémente les consignes mentionnées ci-dessus et qui est muni d'un package nommé ***cms_ctr4*** contenant deux classes publiques :

- la classe graphique ***MachineASous*** qui correspond à l'interface graphique décrite auparavant ;
- la classe principale ***CP_Ctr4Exo2*** dont le code source est donné ci-dessous et qui contient :
    - la méthode publique et statique ***main*** (qui crée une instance de la classe graphique ***MachineASous***) ;
    - la méthode publique et statique ***hazarder*** (qui est utilisée dans la classe graphique ***MachineASous***).

Le but de la méthode ***hazarder*** est de générer et retourner un nombre aléatoire entier compris entre une borne inférieure entière donnée comme premier argument et une borne supérieure entière donnée comme deuxième argument.

```java
package cms_ctr4;
public class CP_Ctr4Exo2
{     public static void main(String[] args)
      {
            new MachineASous();
      }//fin de la méthode main

      public static int hazarder(int min, int max)
      {
            return (int)(min + Math.random()*(max-min));
      }//fin de la méthode hazarder
}//fin de la classe principale
```

Dans ce deuxième exercice, on vous demande d'écrire le code source de la classe graphique ***MachineASous***, en complétant le canevas proposé plus loin et en fonction des indications données ci-dessous.

A) Afin de réaliser la **partie design** :

- la classe publique ***MachineASous*** doit correspondre à un container swing de premier niveau personnalisé (c'est-à-dire à une fenêtre graphique principale affichable directement à l'écran) ;
- dans la partie "du haut" du container de premier niveau, on dispose un container swing intermédiaire créé pour l'occasion, identifié dans le code par le champ ***panAffichage*** et dont on change le gestionnaire de mise en forme (Layout Manager) par défaut pour qu'il corresponde à une grille avec trois lignes et une seule colonne ;
- dans les trois cellules de cette grille, on place trois boutons personnalisés de type classe interne ***MonBouton*** (et cette classe sera détaillée ci-dessous) ; ces trois boutons sont identifiés dans le code par les champs ***jbHaut***, ***jbCentre*** et ***jbBas*** ; ils ne sont pas sensibles aux clics de la souris et ont comme rôle de permettre l'affichage des trois chiffres dont on a déjà parlé ;
- dans la partie "centrale" du container de premier niveau, on dispose un container swing intermédiaire créé pour l'occasion, identifié dans le code par le champ ***panControles*** et dans lequel on place le bouton personnalisé ***Jouer***, identifié dans le code par le champ ***jbJouer***, et celui de sa droite, identifié dans le code par le champ ***jbBloquer*** ; ces deux boutons sont de type classe interne ***MonBouton*** et ils sont sensibles aux clics de la souris ;
- dans la partie "du bas" du container de premier niveau on place une étiquette swing identifiée dans le code par le champ ***jlResultat***.

En outre, afin d'obtenir l'effet d'animation qui fait défiler les trois chiffres affichés, il faut prévoir un moteur de l'animation sous la forme d'un objet non graphique ayant un type approprié et identifié dans le code par le champ ***moteur***. Cet objet doit être capable d'envoyer des événements de type ***ActionEvent*** au container de premier niveau (à la fenêtre graphique) avec une périodicité de ***100*** millisecondes.

La classe interne privée ***MonBouton*** :

- est définie au début de la classe englobante ***MachineASous***, juste après les déclarations des champs privés ;
- doit hériter d'une classe prédéfinie appropriée afin de correspondre à un bouton poussoir (à cliquer) swing personnalisé ;
- définit un constructeur (sans modificateur d'accès) avec deux arguments : le premier nommé ***s*** de type chaînes de caractères et le deuxième nommé ***c*** de type classe couleur prédéfinie ; avec ce constructeur on doit pouvoir créer un bouton personnalisé qui affiche le texte ***s*** avec la couleur ***c*** et avec une certaine police ; plus précisément, le constructeur doit respecter les consignes suivantes :
    - par un appel au constructeur de la classe mère (avec un argument de type chaîne de caractères), la valeur du premier argument ***s*** devient le texte associé au bouton personnalisé ;
    - on crée une variable locale nommée ***f*** de type "police prédéfinie" et on y stocke l'adresse d'un nouvel objet créé pour l'occasion et correspondant à une police ***Arial***, ***grasse*** et de taille ***32*** ;
    - on indique que la police ***f*** définie ci-dessus doit être utilisée pour afficher le texte associé au bouton personnalisé ;
    - on indique que la couleur correspondant au deuxième argument ***c*** doit être utilisée pour afficher le texte associé au bouton personnalisé.

Le design effectif de l'interface graphique est réalisé grâce au constructeur public sans argument de la classe (englobante) ***MachineASous***.

B) Afin de réaliser la **gestion des événements**, la classe ***MachineASous*** doit :

- implémenter une interface Java appropriée afin d'être à l'écoute des boutons **Jouer** et ***OK/KO*** ainsi que du moteur de l'animation ;
- (re)définir le gestionnaire d'événements correspondant à l'interface implémentée.

C) Afin de réaliser la **partie métier**, la (re)définition du gestionnaire d'événements doit assurer les comportements décrits ci-dessous.

Chaque fois que l'utilisateur appuie sur le bouton ***Jouer*** :

- la valeur du champ ***index*** est remise à zéro ;
- une valeur entière aléatoire comprise entre les valeurs des champs ***MIN*** et ***MAX*** est générée (à l'aide de la méthode prévue à cet effet dans la classe principale) et stockée dans le champ ***indexFin*** ;
- le moteur de l'animation est (re)démarré.

Chaque fois que l'utilisateur appuie sur le bouton situé à droite du bouton ***Jouer*** :

- le moteur de l'animation est arrêté ;
- si le texte associé au bouton appuyé était ***OK*** (écrit en rouge) :
    - le texte associé au bouton devient ***KO*** écrit en bleu ;
    - le texte apparaissant en bas de la fenêtre principale devient ***Bloqué*** écrit en bleu ;
- autrement si le texte associé au bouton appuyé était ***KO*** (écrit en bleu) :
    - le texte associé au bouton devient ***OK*** écrit en rouge ;
    - le texte apparaissant en bas de la fenêtre principale devient ***Bonne chance*** écrit en rouge.

Chaque fois que le moteur de l'animation envoie un nouvel événement :

- si le texte associé au bouton situé à droite du bouton ***Jouer*** est ***OK*** :
    - les valeurs des trois chiffres affichés en haut de la fenêtre principale augmentent d'une unité "de manière circulaire " (dans le sens où, comme déjà expliqué, le successeur du chiffre 9 est le chiffre 0) ;
    - la valeur du champ ***index*** est incrémenté d'une unité ;
    - si la (nouvelle) valeur du champ ***index*** est strictement plus grande que la valeur du champ ***indexFin*** :
        - le moteur de l'animation est arrêté ;
        - si le chiffre affiché "au milieu" (c'est-à-dire en deuxième position) est impair, le texte apparaissant en bas de la fenêtre principale devient ***Gagné !*** écrit en rouge ;
        - autrement, le texte apparaissant en bas de la fenêtre principale devient ***Perdu*** écrit en bleu.
- autrement (c'est-à-dire si le texte associé au bouton situé à droite du bouton ***Jouer*** est ***KO***), il n'y a rien qui se passe.

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

```
//déclaration du package
……
……
//importations des packages prédéfinis
……
……
……
……
……
……
//en-tête complet de la classe graphique
……
……
……
……
……
{     /*début de la classe graphique*/
      /*déclarations des champs PRIVES*/
      //déclarer deux champs de type "container intermédiaire" sans
      //valeurs initiales et nommés panAffichage et panControles
……
……
……
      //déclarer trois champs de type MonBouton sans
      //valeurs initiales et nommés jbHaut, jbCentre et jbBas
……
……
……
      //déclarer deux champs de type MonBouton sans
      //valeurs initiales et nommés jbJouer et jbBloquer
……
……
……
      //déclarer un champ de type "étiquette" sans valeur initiale
      //et nommé jlResultat
……
……
```

```
    //déclarer un champ de type "moteur de l'animation" sans valeur
    //initiale et nommé moteur
……
……
    //déclarer deux champs de type numérique entier nommés MIN et
    //MAX dont les valeurs initiales 10 et, respectivement, 20 sont
    //figées par la suite
……
……
……
    //déclarer deux champs de type numérique entier nommés index et
    //indexFin dont les valeurs initiales sont 0 et,
    //respectivement, la valeur du champ MIN
……
……
……

    //définition (en-tête et corps) de la classe interne MonBouton
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
```

```
     //l'en-tête du constructeur de la classe graphique
……
……
     {     /*début du corps du constructeur
           //appeler une méthode qui assure que la fermeture de la
           //fenêtre principale arrête aussi la JVM
……
……
……
           //donner le titre MAS à la fenêtre principale
……
……
           //rendre la fenêtre principale non rédimensionnable
……
……
           //créer un container intermédiaire et stocker son adresse
           //dans le champ panAffichage
……
……
           //associer au container ci-dessus un LayoutManager créé
           //pour l'occasion et correspondant à une grille avec trois
           //lignes et une colonne
……
……
           //créer un bouton personnalisé avec 1 comme texte affiché
           //en gris et stocker son adresse dans le champ jbHaut
……
……
           //ajouter le bouton ci-dessus au container panAffichage
……
……
           //créer un bouton personnalisé avec 2 comme texte affiché
           //en rouge et stocker son adresse dans le champ jbCentre
……
……
           //ajouter le bouton ci-dessus au container panAffichage
……
……
           //créer un bouton personnalisé avec 3 comme texte affiché
           //en gris et stocker son adresse dans le champ jbBas
```

……

……

//ajouter le bouton ci-dessus au container ***panAffichage***

……

……

//ajouter le container ***panAffichage*** "en haut" de la
//fenêtre principale

……

……

//créer un container intermédiaire et stocker son adresse
//dans le champ ***panControles***

……

……

//créer un bouton personnalisé avec ***Jouer*** comme texte
//affiché en ***noir*** et stocker son adresse dans le champ
//***jbJouer***

……

……

//rendre le bouton ***jbJouer*** sensible en le mettant sous
//l'écoute de la fenêtre principale

……

……

//ajouter le bouton ***jbJouer*** au container ***panControles***

……

……

//créer un bouton personnalisé avec ***OK*** comme texte affiché
//en ***rouge*** et stocker son adresse dans le champ ***jbBloquer***

……

……

//rendre le bouton ***jbBloquer*** sensible en le mettant sous
//l'écoute de la fenêtre principale

……

……

//ajouter le bouton ***jbBloquer*** au container ***panControles***

……

……

//ajouter le container ***panControles*** "au centre" de la
//fenêtre principale

……

……

```
            //créer une étiquette qui affiche le texte Bonne chance
            //et stocker son adresse dans le champ jlResultat
……
……
            //créer un nouvel objet de type "police" correspondant à
            //une police Arial, grasse et de taille 24 et stocker son
            //adresse dans une variable locale créée à cet effet et
            //appelée f
……
……
            //indiquer que le texte de l'étiquette jlResultat doit
            //être affiché avec la police précisée ci-dessus
……
……
            //indiquer que le texte de l'étiquette jlResultat doit
            //être centré horizontalement à l'affichage
……
……
            //indiquer que le texte de l'étiquette jlResultat doit
            //être affiché en rouge
……
……
            //ajouter l'étiquette jlResultat "en bas" de la fenêtre
            //principale
……
……
            //appeler une méthode qui assure que la taille de la
            //fenêtre principale s'adapte (à l'exécution et de manière
            //optimale) à son contenu
……
……
            //rendre la fenêtre principale visible à l'écran
……
……
            //créer le moteur de l'animation qui envoie ses signaux
            //avec une périodicité de 100 ms à la fenêtre principale
……
……
      }//fin du constructeur
```

```
//définition (en-tête et corps) de la méthode gestionnaire
//d'événements (selon les indications données au point C))
```

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

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

}//fin de la classe graphique ***MachineASous***

A part ce texte, cette page doit rester vide, mais vous pouvez l'utiliser si la place prévue pour vos réponses n'a pas été suffisante.