TP 4 : partie 1 (1,5h)
Exceptions – Lecture de fichier texte – Sérialisation/Désérialisation

Dans cette partie 1, on reprend le sujet du TP2 Java semaine 20/4/2020 que vous connaissez bien, pour mettre en pratique les notions qui sont énoncées dans le Cours 5 : semaine du 27/4/2020

1. Dans la méthode *Supprimer()* de la classe *Liste* :
➔ Si la liste est vide de la liste, cette méthode lève la nouvelle exception que l'on nommera *VideException*, pour le motif que l'action de cette méthode est impossible en cas de liste vide.
➔ Cette méthode propage cette exception *VideException*.

2. Dans la méthode *Suivant()* de la classe *Liste* :
➔ Si la liste est vide, cette méthode lève la nouvelle exception *VideException* (la même que celle de la méthode *Supprimer()* pour le même motif).
➔ Si la position se trouve au dernier élément de la liste, cette méthode lève la nouvelle exception que l'on nommera *DernierException*.
➔ Cette méthode propage les deux exceptions *VideException* et *DernierException*.

3. Définir deux nouvelles classe d'exception *VideException* et *DernierException* pour lesquelles vous pourrez définir des attributs et des méthodes tel que vu en cours.

4. Dans la méthode *Depiler()* de la classe *Pile*, propager l'exception *VideException* qui a pu être provoquée par l'appel de la méthode *Supprimer()* de *Liste*.

5. De même la méthode surchargée *Depiler()* de sa classe fille *PileEntiers* propage l'exception *VideException* qui peut être reçue par la méthode homonyme de la classe mère.

6. Reprendre le ***main()*** de exercice **4** du TP2 mais avec les changements suivants :
➔ Remplir la pile d'entiers *P1*, non pas avec des valeurs aléatoires, mais en lisant des entiers qui se trouvent dans un fichier texte. Pour commencer, on supposera que ce fichier ne contient que des entiers de nombre variable, soit avec un entier par ligne, soit tous dans la même ligne et séparés par un espace.

Pour vous aider à lire des valeurs d'un fichier, aidez-vous de l'« **Exemple 3 : Lecture dans un fichier** » du tutoriel **Exemples avec la classe Scanner** et de l'API de la classe ***File*** **https://docs.oracle.com/javase/8/docs/api/java/io/File.html**

Comme le montre cet « Exemple 3 », essayer d'instancier un objet de la classe ***File*** peut déclencher l'exception ***FileNotFoundException*** si le fichier n'existe pas. Dans ce cas, il faut attraper cette exception et afficher un message d'erreur.

➔ Le dépilement des entiers de la copie de *P1* (eh oui il faut une copie pour garder intacte cette pile) vers les piles *P2* (nombres pairs) ou *P3* (nombres impairs) peut déclencher l'exception *VideException* si la pile de départ est vide. Dans ce cas, il faut attraper cette exception et afficher un message d'erreur.

➔ Dans un second temps, le fichier pourrait contenir des valeurs autres que des entiers. Dans ce cas, la lecture d'un entier avec la méthode ***nextInt()*** de la classe ***Scanner*** déclenche l'exception ***InputMismatchException***. De même si le fichier est vide, la lecture d'un entier déclenche l'exception ***NoSuchElementException***. Dans ces 2 cas d'exceptions, il faut les attraper et afficher un message d'erreur.
https://docs.oracle.com/javase/8/docs/api/java/util/InputMismatchException.html
https://docs.oracle.com/javase/8/docs/api/java/util/NoSuchElementException.html

➔ Pour finir avec ce ***main()***, « sérialiser » la pile d'entiers P1 (donc un objet) dans un fichier binaire. Puis « désérialiser » l'objet de ce fichier pour en extraire une pile d'entiers et afficher ses valeurs.
Sources : le tutoriel **Exemples de sérialisation** et l'API de l'interface ***Serializable***
https://docs.oracle.com/javase/8/docs/api/java/io/Serializable.html

JP Segado

TP 4 : partie 2 (1,5h)
Pattern MVC - Librairie graphique Swing - Programmation évènementielle avec Listeners

Dans cette partie 2, on souhaite mettre en pratique le pattern MVC (Modèle Vue Contrôleur) dans le cadre d'une interface Homme-Machine (IHM) graphique avec la librairie ***javax.swing***, utilisant de la programmation évènementielle avec des Listeners, tels qu'énoncés dans le Cours 6 : semaine du 11/05/2020.
Dans chacun des exercices suivants, le code sera structuré en 3 packages selon le pattern MVC : *modele*, *vue* et *controleur*.

Pour plus d'informations sur le pattern MVC, consulter les sources suivantes :
https://openclassrooms.com/courses/apprenez-a-programmer-en-java/mieux-structurer-son-code-le-pattern-mvc
https://fr.wikipedia.org/wiki/Mod%C3%A8le-vue-contr%C3%B4leur

Exercice 1 - Première fenêtre

Pour créer une fenêtre on dispose dans le package ***javax.swing***, d'une classe standard nommée ***JFrame***, possédant un constructeur sans argument : voir l'API
https://docs.oracle.com/javase/8/docs/api/javax/swing/JFrame.html

- Dans le package *vue*, créer votre première fenêtre en définissant la classe suivante héritant de ***JFrame*** :

```
import javax.swing.*;
public class MaFenetre extends JFrame
{
     /**
         Constructeur par défaut
      */
     public MaFenetre ()   { // constructeur
         setSize (300, 150); // donne une taille en hauteur et largeur à la fenêtre
         setTitle ("Ma premiere fenetre"); // donne un titre à la fenêtre
     }
}
```

- Dans le package *controleur*, définissez une classe *TestFenetre* avec le ***main()*** qui crée et rend visible la fenêtre avec un objet de la classe de la *vue* ci-dessus.

- Dans le package *modele*, définissez une classe *ModeleFenetre* contenant les propriétés suivantes :
  - Attributs **private** : les dimensions et le titre de la fenêtre.
  - Un constructeur par défaut qui initialise les dimensions et le titre avec des valeurs par défaut.
  - Une méthode *saisirFen()* qui saisit au clavier les dimensions et le titre, et les copie dans les attributs. Si les dimensions saisies sont incorrectes (négatives ou de mauvais format), cette méthode propage l'exception générale ***Exception***.
  - Les getters de tous les attributs.

- Dans la classe du package *vue*, définissez un second constructeur avec les dimensions et le titre en paramètre, pour respectivement dimensionner et intituler la fenêtre.

- Dans le ***main()*** de la classe *TestFenetre* du package *controleur*, créer un objet de la classe *ModeleFenetre* du package *modele* et appeler sa méthode *saisirFen()*. Créer une nouvelle fenêtre du package *vue* avec les dimensions et le titre de cet objet. En cas de mauvaise saisie des dimensions de l'objet de *ModeleFenetre*, il faut attraper l'exception ***Exception*** et afficher une erreur.

Exercice 2 - Evénements

Les interfaces graphiques sont fondées sur le principe des événements. La plupart des événements sont créés par des composants qu'on aura introduits dans la fenêtre (menu, boutons, etc).
Il est nécessaire pour cela d'utiliser le package ***java.awt.event***. Les fenêtres doivent utiliser les événements via des écouteurs qu'ils implémentent comme **ActionListener**, ainsi on aura :

```
package vue ;
import javax.swing.*;
import java.awt.event.*; // pour ActionEvent et ActionListener

public class VueFenetre extends JFrame implements ActionListener
{
  public VueFenetre () {
     addActionListener (this); // la fenêtre sera son propre écouteur d'évènements
  }

 // Méthode appelée en fonction de chaque événement :
 public void actionPerformed(ActionEvent ev) {…} // à compléter
}

package controleur;
import vue.*;

public class Clic1
{
  public static void main (String args[]) {
       VueFenetre fen = new VueFenetre();
       fen.setVisible(true);
  }
}
```

- Dans la classe *VueFenetre* du package *vue* ci-dessus, compléter la méthode *actionPerformed* qui réagit au clic de la souris dans une fenêtre en affichant "clic souris" dans la fenêtre de commande.

- Dans un nouveau package *modele*, définissez une classe *ModeleSouris* contenant les propriétés suivantes :
- Attributs **private** : les coordonnées de la souris.
- Un constructeur avec les coordonnées de la souris copiées dans les attributs.
- Les getters de tous les attributs.

- Dans la classe *VueFenetre*, définir en attribut **public** un objet de *ModeleSouris* et une méthode *afficherSouris()* qui affiche les coordonnées de l'attribut. Compléter la méthode *actionPerformed* qui récupère les coordonnées d'un clic généré par l'événement ev par **ev.getX()** et **ev.getY()**, puis instancie l'objet de *ModeleSouris* avec ces coordonnées.

- Le ***main()*** du *controleur* récupère les coordonnées la souris, par l'intermédiaire de l'objet de *ModeleSouris* défini en attribut de l'objet *fen* de la classe *VueFenetre*. Puis il appelle la méthode *afficherSouris()* pour afficher les coordonnées.

Sources :
➔ les tutoriels **Les interfaces graphiques Swing** ; **Exemples de gestion evenementielle avec Swing**
➔ les API **https://docs.oracle.com/javase/8/docs/api/java/awt/event/ActionListener.html** et **https://docs.oracle.com/javase/8/docs/api/java/awt/event/ActionEvent.html**

Exercice 3 - Dessin

Le programme suivant permet de dessiner une droite. Modifiez le pour dessiner des droites avec la souris.

```
import javax.swing.*;
import java.awt.*;
public class MaFenetre extends JFrame
{
  private JPanel pan ;
  public MaFenetre ()  {
      setTitle ("Essai dessins");
      setSize (300, 150);
      pan = new Paneau();
      getContentPane().add(pan);
      pan.setBackground(Color.yellow); // couleur de fond = jaune
  }
}

public class Paneau extends JPanel
{
   public void paintComponent(Graphics g)  {
      super.paintComponent(g);
      g.drawLine (15, 10, 100, 50);
  }
}

public class PremDes
{
   public static void main (String args[]) {
       MaFenetre fen = new MaFenetre();
       fen.setVisible(true);
   }
}
```