Projet IMAC 2 – Java

Feriel.Goulamhoussen@u-pem.fr

Le projet consiste en la création d'un Captcha intelligent en java.
Il s'agit de créer une interface graphique qui permettra de sélectionner les images dans un pool d'images organisé et relié à une bibliothèque en java.

Le fonctionnement du logiciel

1. Le captcha se lance et aléatoirement, le logiciel choisit une catégorie d'images à sélectionner et entre 1 et 4 images de cette catégorie.

Par exemple : sélectionnez toutes les images qui affichent un panneau.

2. L'utilisateur sélectionne les images et un petit algorithme détecte si le résultat est correct.
Pour valider, il est nécessaire de cliquer sur le bouton « Ok » et un Label « Veuillez sélectionner les images qui contiennent …. » est obligatoire.
Le bouton « Ok » doit vérifier que les images sélectionnées sont sont bien celles qui correspondent à la question posée.

En fonction du résultat, une boite de dialogue s'ouvre et affiche si le résultat est le bon ou non.

3. Si le résultat n'est pas correct, alors l'algorithme utilise plus d'images OU des images plus difficiles à sélectionner que celles précédemment utilisées (car il peut s'agir d'un robot). Les images doivent être du même type (par exemple panneau) mais catégorisées comme plus difficile. Il faut utiliser astucieusement l'héritage. Il faut alors sélectionner tous les panneaux ronds, etc.

Il peut y avoir une quantité infinie de difficulté. A vous de voir combien vous souhaitez en ajouter.

Interface graphique

Pour faire cela, l'exemple de code joint va vous aider à créer une interface graphique. Vous ne serez pas noté sur l'interface graphique mais sur l'intelligence que vous allez mettre pour créer ce système.

La classe UI jointe dispose de plusieurs éléments qui permettent de mettre en œuvre votre logiciel de captcha. Il est nécessaire d'utiliser la bibliothèque SWING de java. Cette bibliothèque contient tout ce dont vous avez besoin pour écrire votre logiciel.

1. Pour créer une nouvelle fenêtre, utilisez la classe JFrame.

```
JFrame frame = new JFrame("Captcha"); // Création de la fenêtre principale
```

2. On y ajoute un layout, c'est à dire un « mode d'organisation » du conteneur principal. Ici, vu que nous allons utiliser des images côte à côte, nous allons utiliser un GridLayout.

```
GridLayout layout = new GridLayout(4,3); // Création d'un layout de type Grille avec 4 lignes et 3 colonnes
```

Cf. la documentation : https://docs.oracle.com/javase/7/docs/api/java/awt/GridLayout.html

Il y a 4 lignes et 3 colonnes dans la fenêtre.

Image	Image	Image
Image	Image	Image
Image	Image	Image
Texte d'instruction	Bouton OK	

3. Utilisation du bouton « Ok » et des événements associés

Les boutons en swing, sont de type JButton. Pour plus de simplicité, nous créons ici un bouton dans lequel nous ajoutons une action.

En effet, il est nécessaire de définir le comportement du bouton lorsque l'on clique dessus.
Pour cela, nous utilisons la classe AbstractAction (classe abstraite) dans laquelle nous définissons le code manquant, c'est à dire la méthode actionPerformed :

```
new JButton(new AbstractAction("Vérifier") {

                @Override
                public void actionPerformed(ActionEvent arg0) {
                      EventQueue.invokeLater(new Runnable() {

                            @Override
                            public void run() {
                                  System.out.println("J'ai cliqué sur Ok");
                            }
                      });
                }
          });
```

Nous ajoutons ensuite ce que nous souhaitons afficher dans la console dans la queue des événements. Il est très important de lancer les différentes actions dans la queue des événements afin :
- d'éviter les potentiels interblocages
- d'éviter d'utiliser des threads qui ne serviront à rien dans le cadre de ce projet
- et d'autres choses … mais vous le verrez plus tard.

4. Affichage des images.

Pour afficher des images, il est nécessaire de « hacker » les composants JLabel. Les composants JLabel affichent une ligne de texte. Néanmoins, il est tout à fait possible d'afficher des images.

Pour rechercher une image dans votre projet java, il est nécessaire d'utiliser la méthode getResource de MainUi.class (attention, le tutoriel va venir après).

getResource permet de récupérer l'URL de l'image à afficher. Ensuite, il est nécessaire de lire l'image puis de l'intégrer au JLabel .

```
final URL url = MainUi.class.getResource(imageLocation);

          System.out.println(url);
```

```
            BufferedImage img = ImageIO.read(url);
            Image sImage = img.getScaledInstance(1024/3,768/4,
Image.SCALE_SMOOTH);

            final JLabel label = new JLabel(new ImageIcon(sImage));
```

Enfin, dans le projet il est nécessaire de cliquer sur les images que nous souhaitons sélectionner. S'agissant d'un évènement utilisateur, il est nécessaire d'ajouter un « écouteur » d'évènement de type « souris ». Il s'agit de l'interface MouseListener.

Dans notre cas, il est nécessaire uniquement de sélectionner ou désélectionner des images. Ils s'agit donc de vérifier quand on clique sur un bouton de la souris.

Il est donc nécessaire d'implémenter uniquement la méthode mouseClicked de l'interface MouseListener et d'ajouter dans la queue d'évènement le comportement que nous souhaitons avoir sur notre JLabel .

```
label.addMouseListener(new MouseListener() {
                  private boolean isSelected = false;


                  @Override
                  public void mouseReleased(MouseEvent arg0) {
                  }

                  @Override
                  public void mousePressed(MouseEvent arg0) {

                  }

                  @Override
                  public void mouseExited(MouseEvent arg0) {

                  }

                  @Override
                  public void mouseEntered(MouseEvent arg0) {

                  }

                  @Override
                  public void mouseClicked(MouseEvent arg0) {
                        EventQueue.invokeLater(new Runnable() {

                              @Override
                              public void run() {
                                    if(!isSelected){

      label.setBorder(BorderFactory.createLineBorder(Color.RED, 3));
                                          isSelected = true;
                                          selectedImages.add(url);
                                    }
                                    else {

      label.setBorder(BorderFactory.createEmptyBorder());
                                          isSelected = false;
                                          selectedImages.remove(url);
                                    }
```

```
                    }
                });

            }
        });
```

Enfin, le JLabel est prêt et il peut être ajouté dans l'interface.

5. Pour ajouter des composants dans l'interface, avec le GridLayout, il faut les ajouter dans l'ordre, cf. le code.

```
frame.add(createLabelImage("centre ville.jpg"));
...
frame.add(createLabelImage("voie pieton.jpg"));
frame.add(new JTextArea("Cliquez n'importe où ... juste pour tester l'interface !"));

frame.add(okButton);
```

6. Rendre l'interface visible.

Quand toute l'interface est visible, il est nécessaire de la rendre visible et cela simplement avec :

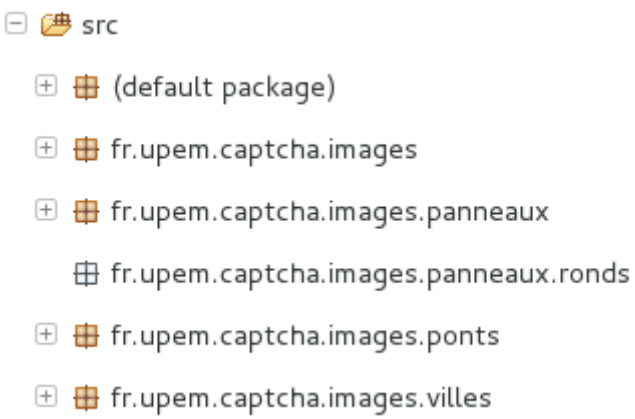
```
frame.setVisible(true);
```

Architecture

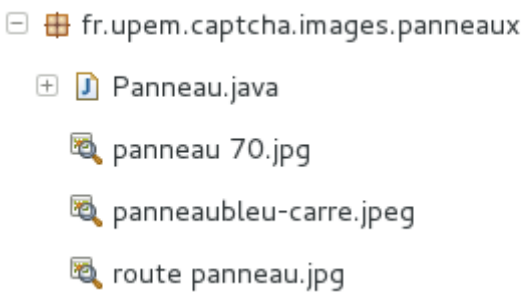
Tout le projet doit être dans un le package fr.upem.captcha

Chaque type de question (panneaux, animaux, bâtiment etc.) doit faire l'objet d'un package. Le package doit contenir les classes nécessaires au contrôle du type par catégories.
La gestion doit être intelligente, c'est à dire que vous devez utiliser toute la puissance objet de java dont l'héritage, l'implémentation d'interface, le sous-typage et les collections.

Chaque bibliothèque d'image doit être organisée dans un package :



Ce système ne nécessite pas de bases de données, mais une utilisation intelligente de java.
Chaque package doit contenir la classe de gestion ainsi que les images



Pour rendre compatible toutes les bibliothèques d'images. Les classes qui recensent les images peuvent contenir des listes ou d'autres types de structures de données qui vous permettent d'obtenir des liens vers les images que vous avez préalablement catégorisées.

Vous devez créer une interface unique. Vous devez également vous aider des classes abstraites pour factoriser le code.

Toutes les classes Panneau et autres (Ponts, Voitures etc) doivent implémenter l'interface « Images » décrites si dessous. Cette interface « Images » peut être agrémentée de méthodes si besoin mais pas nécessairement..

Coup de pouce – Introspection en Java

En java, il est tout à fait possible de créer des objets dynamiquement ainsi que d'interroger les classes en java.

La classe Class<T> (https://docs.oracle.com/javase/7/docs/api/java/lang/Class.html) représente les classes en java sur laquelle on a besoin d'avoir des informations et/ou de l'instancier.

Ces quelques exemples vous seront très probablement utiles.

Instancier des classes de manière dynamique :

```
Class.forName("nom de la classe").newInstance();
```

ou

```
Panneau.class.newInstance();
```

Trouver le package d'une classe

```
Panneau.class.getPackage().getName()
```

etc.

Dans le projet, utiliser l'introspection est fortement conseillé.

Modalités de réalisation

Les projets doivent s'effectuer par binôme. Une exception ne peut être accordée qu'avec l'accord écrit d'un chargé de TD.

Ils seront évalués en deux temps :
- **une soutenance d'une version bêta**, environ 2 semaines avant le rendu final, au cours de laquelle les évaluateurs pourront vous donner des conseils et vous faire des remarques pour poursuivre ou modifier l'implémentation de votre projet. Cette soutenance donne lieu à la moitié de l'évaluation globale et vous avez tout intérêt à être le plus avancé dans votre projet à cette occasion.
- **une version finale**, à rendre à la fin du semestre. Elle intègre dans une seule archive ZIP une documentation technique, un petit mémoire expliquant les choix retenus dans votre projet – dont une section spécifiquement dédiée à la prise en compte des remarques qui vous ont été faites lors de la soutenance bêta, et bien sûr le code source. Les noms doivent apparaître dans chaque classe, chaque fichier et sur le mini mémoire. Le code doit être commenté et la javadoc doit être générée proprement.

Le logiciel doit pouvoir s'exécuter directement en ligne de commande. Un projet rendu uniquement sous eclipse aura une note divisée par 2.