Architecture Logicielle pour l'Interaction

durée indicative : **2h**
documents autorisés : **1 feuille A4 recto-verso manuscrite de votre main**

consignes :
Le barème est **indicatif**. La qualité de la **présentation**, de l'**expression**, de l'**orthographe** sera prise en compte dans la notation de **manière significative**.

Généralités (8 points)

Question 1

Il existe deux sortes de boîtes à outils de construction d'interface : celle qui ont une approche minimaliste et l'approche maximaliste.
a) Expliquez en une phrase ce qui les distingue.
b) Donnez les avantages et les inconvénients de ces deux approches.
c) Donnez le nom d'une boîte à outils de chaque sorte.

Question 2

Selon le modèle de Seeheim, le code de l'interface utilisateur peut se décomposer en trois parties : la présentation, le contrôleur de dialogue et l'interface du noyau fonctionnel, correspondant respectivement aux niveaux lexical, syntaxique et sémantique de l'interaction.
a) Dans un explorateur de fichiers, on utilise la souris pour glisser l'icone d'un fichier sur celle d'un répertoire pour y déplacer le fichier. Expliquez à quoi correspondent les trois niveaux ci-dessus pour l'interaction en entrée à la souris sur cet exemple.
b) Le(s)quel(s) de ce(s) niveau(x) sont fournis par les boîtes à outils de construction d'interface ?

Question 3

Vous faites partie d'une équipe chargée de développer une nouvelle interface pour un logiciel existant. Dans le code du noyau fonctionnel, vous découvrez le passage suivant :

```
package nf;

class UIUpdater {
   ...
   public void loop() {
      while(not ended) {
         // met à jour l'affichage de la valeur mesurée par le capteur
         ui.sensor_label.setText(sensor.getValue().toString());
         // attend 1/10 de seconde avant de remesurer
         Thread.sleep(100);
      }
   }
   ...
}
```

a) Quels sont les problèmes que vous identifiez dans ce code ?
b) Quelles modifications allez-vous suggérer à l'équipe chargée du noyau fonctionnel ?

Programmation par événements (6 points)

Question 4
Vous allez spécifier et réaliser le comportement d'une interaction qui permet de dessiner des segments dans une application de dessin.
Lorsqu'on enfonce le bouton de la souris (↓), on débute un segment à cet endroit.
Lorsqu'on déplace ensuite la souris, l'autre extrémité du segment suit ce déplacement, et quand on relâche le bouton (↑), le segment est terminé.
Si, alors qu'on bouge l'extrémité du segment, la touche *shift* est maintenue enfoncée, le déplacement est contraint pour que le segment soit horizontal, vertical ou incliné à ±45º.
La touche *shift* envoie les évènements P (*press*) quand elle est enfoncée et R (*release*) quand elle est relâchée.

Le segment est géré par une classe fournie qui dispose des méthodes suivantes :
- le constructeur `Segment(Point origin)` qui crée un segment au début du déplacement avec `origin` le point initial qui sera utilisé pour la première extrémité ;
- `void move(Point end)` qui déplace la seconde extrémité du segment, de manière non-contrainte ; et
- `void constrained_move(Point end)` qui déplace la seconde extrémité du segment de manière contrainte.

La touche shift peut être enfoncée ou relâchée à tout moment pendant qu'on dessine un segment.

a) **Dessinez la machine à états qui spécifie cette interaction.**

On rappelle que les évènements ont une méthode `getPoint()` qui retourne un `Point` avec la position du curseur.

b) **Réalisez en java cette machine à états**.

Pour cela, vous est donnée une classe `Canvas` qui utilisera votre classe `Interaction`, et à laquelle on peut ajouter de segments grâce à la méthode `void addSegment(Segment s)`.
On donne également une classe `Segment` (page suivante).
C'est la classe `Interaction` que vous devez réaliser.

```
package exam.ui;

import javax.swing.JComponent;
import java.awt.Point;

public class Canvas extends JComponent {
      Interaction interaction;
      public Canvas() {
            interaction = new Interaction(this);
            addKeyListener(interaction);
            addMouseListener(interaction);
            addMouseMotionListener(interaction);
      }
      public void addSegment(Segment s) { ... }
}
```

```
package exam.ui;

import java.awt.Point;

public class Segment {
      public Segment(Point origin) { ... }
      public void move(Point end) { ... }
      public void constrained_move(Point end) { ... }
}
```

Les interfaces fournies par `java.awt.event` permettant de recevoir les événements sont données ci-dessous.

Pour écouter la **souris** :

```
public interface MouseListener {
      public void mousePressed(MouseEvent e);
      public void mouseReleased(MouseEvent e);
      ...
}

public interface MouseMotionListener {
      public void mouseDragged(MouseEvent e);
      ...
}
```

L'interface `MouseInputListener` qui réunit ces deux interfaces est également disponible.

L'interface suivante permet d'écouter le **clavier** :

```
public interface KeyListener {
      public void keyPressed(KeyEvent e);
      public void keyReleased(KeyEvent e);
      ...
}
```

Les `MouseEvent` ont une méthode `Point getPoint()` qui retourne le lieu de l'événement. Les `KeyEvent` ont une méthode `int getKeyCode()` dont le résultat peut être comparé à la constante `KeyEvent.VK_SHIFT` pour savoir s'il s'agit bien de la touche *shift*.

Votre classe `Interaction` pourra commencer ainsi :

```
package exam.ui;

public class Interaction ... {
      protected Canvas canvas;
      public Interaction(Canvas canvas) {
            this.canvas = canvas;
            ...
      }
      ...
}
```

Composants interactifs (6 points)

Question 5

Les figures ci-dessous donnent la fenêtre de l'éditeur de texte TextWrangler sous Mac OS X avant et après un redimensionnement. Sachant que la largeur de la colonne de gauche peut être modifiée par l'utilisateur, donnez l'arbres des widgets que vous utiliseriez pour coder cette interface avec Java/SWING en spécifiant si nécessaire pour les conteneurs les gestionnaires de géométrie utilisés et la localisation de leurs fils.