Architecture Logicielle pour l'Interaction

durée indicative : **2 heures**
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**.
Si le sujet présente des ambiguïtés, précisez vos choix. Il sera tenu compte de vos hypothèses. Lorsqu'il est demandé de produire du code, ne vous focalisez pas sur la correction de la syntaxe.

Généralités (5 points)

Question 1

a) Comparez l'évolution dans le temps des performances des ordinateurs et celles des humains.
b) Situez dans ce cadre la problématique de l'IHM.

Question 2

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 (8 points)

Question 3

Vous devez réaliser l'interface graphique d'un logiciel qui permet de manipuler des photos sur une surface interactive qui permet d'utiliser les doigts comme pointeurs.
Ceux-ci génèrent trois types d'événements : ↓ lorsqu'un doigt entre en contact avec la surface ; D lorsqu'il se déplace au contact de la surface ; et enfin ↑ lorsqu'il quitte cette dernière.
Ces événements contiennent la position (`e.getPoint()`) où ils ont eu lieu ainsi qu'un identifiant du doigt utilisé (`e.getFingerId()`).

On veut réaliser l'interaction suivante :

- avec 1 doigt, la photo est déplacée (déplacement géré par les fonctions données suivantes : `start_drag(Point start)`, `drag(int dx, int dy)`, `end_drag()`) ;

- avec 2 doigts, la photo est déplacée/tournée/zoomée (redimensionnement géré par la fonction donnée suivante : `pinch(Point old_p1, Point p1, Point old_p2, Point p2)` où old_p1 (resp. p1) désigne la position précédente (resp. courante) de l'un des doigt et old_p2 (resp. p2) celle de l'autre doigt).

Dessinez la machine à état qui réalise ces interactions (vous pouvez faire l'hypothèse qu'au plus 2 doigts seront utilisés en même temps).

Question 4

Réalisez en java la machine à état de la question précédente.
Pour cela, vous est donnée la classe `Surface` qui utilisera votre classe `Listener` de la manière suivante :

```
package exam.ui;

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

public class Surface extends JComponent {
   Listener listener;
   public Surface() {
      listener = new Listener(this);
      addMouseListener(listener);
      addMouseMotionListener(listener);
      ...
   }

   public void start_drag(Point start) { ... }
   public void drag(int dx, int dy) { ... }
   public void end_drag() { ... }

   public void pinch(Point old_p1, Point p1, Point old_p2, Point p2) { ... }
   ...
}
```

Votre classe `Listener` pourra commencer ainsi :

```
package exam.ui;

public class Listener ... {
      protected Surface surface;
      public Listener(Surface surface) {
            this.surface = surface;
            ...
      }
      ...
}
```

Les doigts génèrent en effet des événements souris (`MouseEvent`) qui ont simplement une méthode supplémentaire pour connaître l'identifiant du doigt correspondant.
Les `MouseEvent` ont ainsi les méthodes utiles suivantes :

- `Point getPoint()` qui retourne le lieu de l'événement ;
- `int getX()` et `int getY()` qui retournent la même information ;
- `int getFingerId()` qui retourne un entier qui sera le même pour tout événement entre un ↓ et un ↑ (inclus) si et seulement si c'est le même doigt qui l'a provoqué.

Les interfaces fournies par `java.awt.event` permettant de recevoir les événements sont données ci-dessous, elles disposent toutes d'un adaptateur associé (`XxxxAdapter` pour `XxxxListener`).

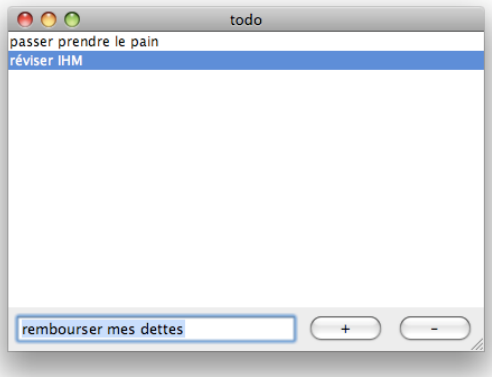
```
public interface MouseListener {
      public void mousePressed(MouseEvent e);
      public void mouseReleased(MouseEvent e);
      ...
}

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

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

Composants interactifs (5 points)

Question 5

L'interface utilisateur de l'application ci-dessous utilise une `JList` pour afficher une liste de tâches à effectuer, un `JTextField` pour en saisir une nouvelle, deux `JButton`, l'un pour ajouter la tâche, l'autre pour supprimer celles sélectionnées.



Écrivez le code qui construit cette interface en complétant ce squelette, en précisant explicitement les *layouts* utilisés (mais sans coder l'interaction) :

```
public class Todo extends JFrame {
   Todo(String title) {
      super(title);
      ...
   }
}
```