Laboratoire 1: Observateur

Durée du laboratoire: 6 périodes. A rendre le jeudi 14 mars sur Cyberlearn.

1. Objectif

Concevoir une application utilisant le modèle *Observateur*.

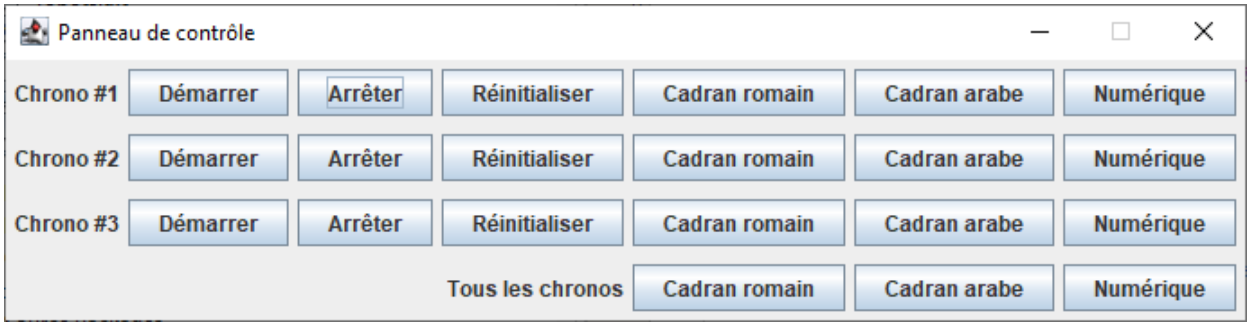
2. Donnée

Définir une application graphique permettant de gérer des chronomètres et d'en afficher la valeur sous différentes représentations.

Chaque chronomètre mesure le temps en heures/minutes/secondes et est identifiable par son numéro, incrémenté à partir de 1.

L'application doit permettre de gérer de 1 à 9 chronomètres, dont le nombre est passé en argument lors du lancement de l'application.

La figure ci-dessous montre l'interface initiale de l'application si elle est lancée avec la valeur 3.

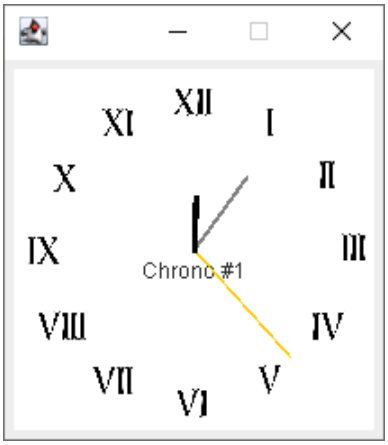


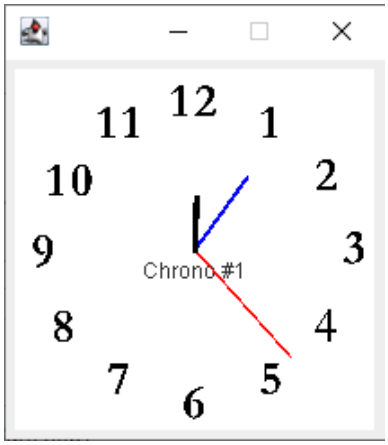
Pour chaque chronomètre il doit être possible, dans l'ordre des boutons affichés, de :

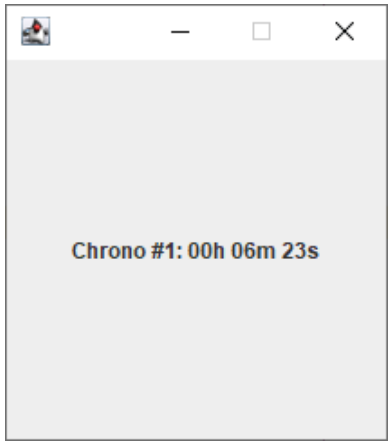
- le (re)démarrer,
- l'arrêter (le mettre en pause),
- le réinitialiser (le remettre à la valeur 0).

Chacun des 3 boutons suivants crée une fenêtre, non redimensionnable, affichant le chronomètre :

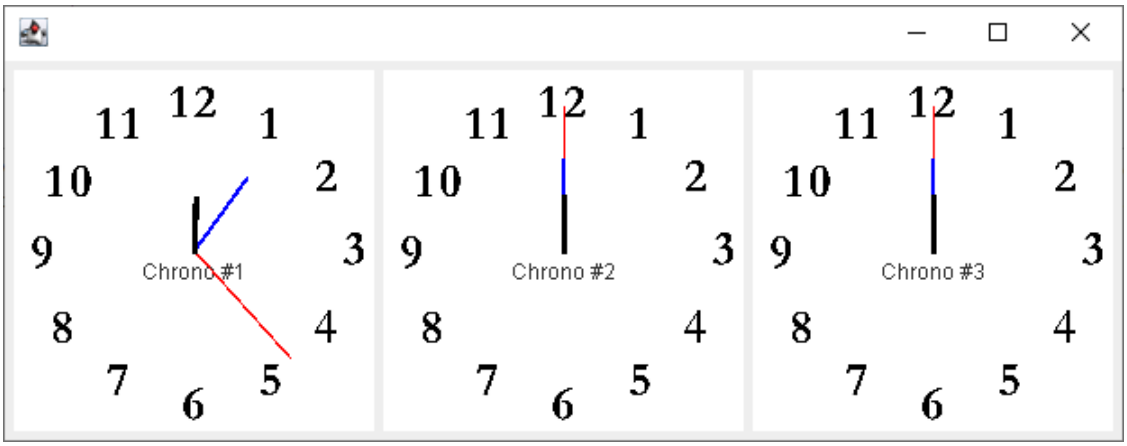
- sous forme analogique (aiguille noire pour le heures, grise pour les minutes et orange pour les secondes) sur un cadran de chiffres romains,
- sous forme analogique (aiguille noire pour le heures, bleue pour les minutes et rouge pour les secondes) sur un cadran de chiffres arabes,
- sous forme numérique (affichage digital).







La dernière « ligne » du panneau de contrôle permet d'ouvrir des fenêtres affichant, chacune, tous les chronomètres sous la forme voulue. Ces fenêtres doivent être redimensionnables, les chronomètres se réalignant selon la place disponible (voir affichage vertical en annexe).



La taille de chaque chronomètre est fixe (200x200 pixels) et son nom (Chrono + #numéro) y est affiché :

- avant sa valeur dans un affichage numérique,
- juste en dessous du centre dans les affichages analogiques.

Il doit être possible de créer autant de fenêtres d'affichage de chronomètre(s) que souhaité et donc de gérer de multiples clicks sur un même bouton d'affichage de chronomètre(s). A chaque fois s une nouvelle fenêtre doit être créée, sans fermer celles existantes.

Cliquer sur l'affichage d'un chronomètre met ce dernier en pause s'il était en marche ou le (re)démarre s'il était à l'arrêt.

Chaque fenêtre doit pouvoir être fermée.

Fermer le panneau de contrôle doit terminer l'application.

3. Implémentation

- Implémenter une version du modèle *Observateur* (ne pas utiliser celle fournie dans l'API Java).
- Pour gérer le temps qui s'écoule dans chaque chronomètre, utiliser un objet `Timer` de l'API Java.
- Les images des cadrans avec chiffres romains et arabes sont fournies.

4. Indications AWT/Swing

- `Toolkit.getDefaultToolkit().getImage(String fileName)` permet d'obtenir un objet `Image` à partir du contenu du fichier ayant le nom `fileName`.
- `getScaledInstance(int width, int height, int hints)` permet ensuite de redimensionner l'objet de type `Image`.
- Dessiner l'objet `Image` dans un `Graphics`, en utilisant la méthode `drawImage(Image img, int x, int y, ImageObserver observer)`.
- Définir chaque fenêtre affichant un ou plusieurs chronomètres au moyen d'une `JFrame` contenant un `JPanel` par chronomètre.
- Afficher le texte de l'horloge numérique (nom + valeur) dans un `JLabel`.
- Afficher le nom du chronomètre dans un affichage avec un cadran, en utilisant la méthode `drawString(String s, int x, int y)` sur un `Graphics` (celui du `JPanel` contenant l'`Image` du cadran).
- Pour le panneau de contrôle des chronomètres, faire un `JPanel` par « ligne ».
- Positionner les éléments dans les `JFrame` et `JPanel` en invoquant la méthode `setLayout(LayoutManager mgr)` et en lui passant en paramètre un `layout` adapté (`FlowLayout`, `GridLayout`, …).
  Ceci est notamment utile pour positionner les éléments dans le panneau de contrôle et pour que les chronomètres puissent se repositionnent lors d'un redimensionnement des fenêtres.
- La gestion des clicks sur les boutons est réalisée en redéfinissant la méthode `actionPerformed` d'un objet implémentant `ActionListener` qui a été abonné à l'objet `JButton`.
  Exemple implémenteant anonymement `ActionListener` :

```
JButton xxx = new JButton("…");
xxx.addActionListener(new ActionListener() {
  @Override
  public void actionPerformed(ActionEvent ae) {
    /* code à exécuter après un click sur le bouton xxx */
  }
});
```

5. Rendu

- Diagramme de classes.
- Code source.

6. Annexe

Affichage de 3 chronomètres après redimensionnement de la fenêtre pour avoir un affichage vertical.