

Examen

Sujet 4

Aucun document autorisé. Seul matériel autorisé : stylos, crayons et règle. Aucun autre matériel autorisé. Le barème est donné à titre indicatif. Durée de l'épreuve : 2h00.

Exercice 1 : [Analyse de code] (5 points)

Soit le programme suivant.

----- Début ProgExam.java -----

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class ProgExam extends JApplet {
    private String texte = "34";
    private boolean test = false;

    private JPanel pan;
    private JButton bouton;
    private JLabel lab;
    private JTextField textF;

    public void stop() {
        texte = "JAVA";
        test = true;
        textF.setText("68");
    }

    public void init() {
        pan = new JPanel(new GridLayout(1,3));
        setContentPane(pan);
        lab = new JLabel("");
        textF = new JTextField("");
        bouton = new JButton("OK");
        bouton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                if (test) {
                    texte = texte + ":";
                    lab.setText(texte);
                    test = false;
                }
            }
        });
    }
}
```

```

        }
        else {
            texte = texte + textF.getText();
            lab.setText(texte + "*");
        }
    }
});
pan.add(textF);
pan.add(bouton);
pan.add(lab);
}
}

```

Fin ProgExam.java

Dire ce qui est affiché dans `lab` après *chacune* des actions successives de l'utilisateur :

1. Clic sur le bouton.
2. Saisie de "GO" dans `text`, clic sur le bouton.
3. Clic sur le bouton.
4. Iconification de la fenêtre, restauration de la fenêtre.
5. Clic sur le bouton.
6. Clic sur le bouton.

Exercice 2 : [Programmation IHM] (15 points)

Écrire un programme qui propose une interface permettant d'effectuer des recherches dans une grande base de données. On supposera qu'il existe une méthode statique dont le prototype est `public static Vector loadBase();` (dans une classe `Examen` ; qui ne sont ni l'une ni l'autre à écrire) et qui renvoie un élément de la classe `Vector` initialisé avec les données de la base. Les données de la base sont des identités composées d'un nom, d'un prénom et d'une adresse, regroupés dans une classe `Identity` (qui est, elle, à écrire complètement).

L'interface graphique fonctionnera de la façon suivante. Au centre de la fenêtre, un `JTextArea`, permettra de voir le résultat des recherches dans la base. Au dessus se trouvera un `JTextField` permettant de saisir un nom. En dessous se trouveront deux `JButton`, de titre "Chercher" et "Stop". Lorsque l'utilisateur clique sur le bouton "Chercher", la recherche est lancée : les informations, nom, prénom et adresse, dont le nom correspond au texte contenu dans le `JTextField` seront affichées, dans le `JTextArea`, au fur et à mesure de leur récupération. Si l'utilisateur clique sur "Stop" pendant la recherche, celle-ci doit s'arrêter, et les titres des `JButton` doivent être changés en "Reprendre" et "Annuler". Si l'utilisateur clique alors sur "Reprendre", la recherche doit reprendre là où elle s'était arrêtée, tandis que s'il clique sur "Annuler", le contenu du `JTextArea` et du `JTextField` doivent être effacés ; dans les deux cas, les `JButton` doivent reprendre leur titre d'origine.

Bien veiller à respecter l'architecture MVC vue en cours ; en particulier, ce n'est pas le thread gestionnaire d'événements qui doit faire la recherche. Choisir pertinement le nombre de threads à utiliser. Il est inutile de considérer les cas où l'utilisateur effectue des actions qui ne correspondent pas au déroulement décrit dans l'énoncé.