

ALGORITHMIQUE III.7

HÉRÉDITÉ

À propos des exercices individuels Les exercices notés (*) sont obligatoires et individuels. Ils sont à faire avant le prochain *cours*. Ils seront notés. Une réponse fautive, sans commentaires et rendue par plusieurs personnes sera considérée comme de la copie et vaudra 0.

En d'autres termes, *testez vos programmes* !

À propos des exercices collaboratifs Les exercices notés © sont aussi obligatoires et partagés entre toute la classe. Ils sont aussi à faire avant le prochain cours. Ils seront aussi notés mais la note sera appliquée à toute la classe.

Spécialisons Swing

Il y a deux semaines, nous avons défini des groupes de `JPanel`s imbriqués dans d'autres groupes de `JPanel`... C'était bien gentil, mais on fatigue assez vite. Au-delà de trois ou quatre profondeurs, on en a très vite assez.

L'une des conséquences sympathiques de l'héritage est que vous pouvez facilement définir de nouveaux composants à partir de composants existants. La méthode la plus simple est de créer une sous-classe de `JPanel` et, dans le constructeur, d'ajouter des composants à l'aide de `this.add`. Vous pouvez aussi, bien entendu, ajouter des méthodes ou remplacer des méthodes de `JPanel`.

Exercice 1. Créez une nouvelle classe, que vous appellerez, mettons, `JCardinalLabels`. Ce nouveau composant, qui héritera de `JLabel`, contiendra 5 sous-composants : quatre labels placés respectivement au Nord, au Sud, à l'Est et à l'Ouest et un `JPanel` placé au centre.

Cette classe doit implanter l'interface `LabelsOnTheSidePanelInTheCenter`.

Exercice 2. Créez une sous-classe de `JCardinalLabels` appelée `JCardinalLabels2` et qui disposera en plus d'un constructeur

```
/**
 * Construit un JCardinalLabels2
 *
 * @param north Le texte à mettre sur le label placé au Nord
 * @param east  Le texte à mettre sur le label placé à l'Est
 * @param south Le texte à mettre sur le label placé au Sud
 * @param west  Le texte à mettre sur le label placé à l'Ouest.
 */
public JCardinalLabels2(String north, String south, String east, String west)
```

Exercice 3. Créez une sous-classe de `JCardinalLabels2` appelée `JCardinalLabels3` et qui disposera en plus d'un constructeur

```
/**
 * Construit un JCardinalLabels3
 *
 * @param north Le texte à mettre sur le label placé au Nord
 * @param east  Le texte à mettre sur le label placé à l'Est
 * @param south Le texte à mettre sur le label placé au Sud
 * @param west  Le texte à mettre sur le label placé à l'Ouest.
 * @param center Le composant à mettre au centre.
 * @param color La couleur dans laquelle remplir les quatre labels.
 */
public JCardinalLabels3(String north, String south, String east, String west,
    JComponent center, Color color)
```

Remplacez de plus la méthode `setForeground` de `JComponent` de manière à ce qu'elle change la couleur des quatre labels.

Exercice 4. À l'aide de `JCardinalLabels3`, écrivez en une (longue) ligne quelque chose qui ressemble à ce qui était demandé à l'exercice 2 du TD 5.

Exercice 5. Juste pour le défi, à l'aide de `JCardinalLabels3`, écrivez quelque chose qui ressemblera à ce qui était demandé à l'exercice 2 du TD 5 – mais cette fois avec une profondeur de 10 imbrications. Utilisez une boucle, hein, vous gagnerez du temps.

Time and time again

Le répertoire `animation` contient une quinzaine d'images qui, à elles toutes, forment une animation. Nous allons essayer de définir un composant Swing qui sera chargé uniquement de jouer cette animation.

Exercice 6. * Créez une fenêtre Swing composée en tout et pour tout d'un label affichant une image de votre choix parmi les images que je vous ai fournies.

Rappelons que, pour charger une image, on utilise généralement quelque chose du genre

```
Icon icon = new ImageIcon(nom_de_la_classe.class.getResource('nom de l'image'))
```

Exercice 7. * Modifiez ce qui précède de manière à ce que votre programme

- charge toutes les images au démarrage (mettez-les dans un tableau et, par pitié, faites une boucle); puis
- démarre un `Timer` qui prévienne ses `ActionListeners` toutes les 60 millisecondes ;
- à chaque top du `Timer` change l'image affichée sur le label – commencez par la première image, puis la deuxième... puis la dernière, puis la première, puis la deuxième...

Exercice 8. * Complétez ce qui précède à l'aide de deux boutons *démarrer/pause* et *quitter*. Le premier bouton devra interrompre le `Timer` s'il est en marche, le redémarrer s'il est interrompu. Le deuxième bouton se contentera d'appeler `System.exit(0)`; .

Exercice 9. * Réécrivez ce qui précède sous la forme d'une `JFrame` contenant un composant `JAnimationLabel` et les deux boutons *démarrer/pause* et *quitter*. Le composant `JAnimationLabel` sera le seul à connaître le `Timer`, l'`ActionListener` associé à ce `Timer`... et plantera l'interface `Animator`.

Exercice 10. © Mettez les réponses aux exercices 1 à 5 sur le Wiki, y compris ceux qui n'ont pas été corrigés en classe.

Exercice 11. © Mettez le contenu du cours sur le Wiki, y compris la fin du corrigé aux exercices de la semaine dernière. Vous êtes encouragés à redécouper et à reformuler plus clairement.