Université de Nice - Sophia Antipolis
Faculté des Sciences

DEUG MIAS MP1

Programmation 2001-02

6. LE GRAPHISME AVEC AWT

A. LA TOILE DU PEINTRE.

Le peintre ne dessine pas sur le cadre [cadre = `frame` en anglais], il utilise une toile de lin blanche. La classe `Canvas` [toile = `Canvas` en anglais] sert à cela. Un `Canvas` est un composant graphique [`Canvas` est une sous-classe de `Component`] au même titre qu'un `Button`, un `Label`, une `TextBox`. Notre but est d'installer un `Canvas` à l'intérieur d'une `Frame`, tout comme le peintre, afin de pouvoir y dessiner des objets (cercles, lignes, etc.). Avant de vous installer au clavier, vous allez vous renseigner sur la classe `Canvas`.

Exercice 6.1 Que dit la documentation de l'API sur les `canvas` ?

Vous avez 15 minutes pour lire la documentation et répondre sur papier aux questions suivantes

a) Comment créer un objet de type `canvas` ?

b) Quelle méthode donne la *taille préférée* d'un `canvas` ? De quelle classe est-elle héritée ? Quelle autre classe utilise-t-elle ? Quels sont les constructeurs de cette autre classe ?

c) Quelle méthode doit-on modifier pour dessiner sur le `canvas` ? Quelle autre classe utilise-t-elle ? Quelles sont les méthodes de cette autre classe que nous pouvons utiliser pour dessiner une ligne ? un rectangle vide ? un carré vide ? un disque plein ? un point ? Quelle méthode permet de choisir la couleur du tracé ?

d) Comment pouvons-nous utiliser les méthodes trouvées en **b)** et **c)** pour créer un `canvas` ayant une taille fixée et dessiner dessus ? (c'est marqué en haut de la page de documentation !)

B. "DESSINE MOI UN CADRAN".

Nous allons réaliser une petite application classique qui vient avec tous les systèmes d'exploitation: l'horloge. La notre sera analogique avec deux aiguilles (heures et minutes) et un joli cadran gradué comme le montre la capture d'écran à droite. On ajoutera plus tard un affichage digital.

Exercice 6.2 Canvas et Cadran

a) Dans un fichier `MontreFrame.java` implantez la classe `MontreFrame` qui étend la classe `MP1Frame`.

b) Le titre de la fenêtre sera "Tictac".

c) Il n'y a qu'un seul composant à ajouter : un `Canvas`. Pour fixer sa taille et les dessins à y faire, vous allez utiliser une classe anonyme qui surcharge deux méthodes comme on l'a expliqué en **6.1.d**. Le `Canvas` aura une taille de 100×100. Vous dessinerez quatre disques concentriques centrés dans le `Canvas` en respectant l'ordre suivant : un disque plein jaune de rayon *r=48*, un disque plein blanc de rayon *r=45*, un disque plein rouge de rayon *r=4*, un disque plein noir de rayon *r=2*. (on a vu comment choisir une couleur et dessiner un disque à la question **6.1.c**)

d) Ajoutez l'instruction permettant de donner à la fenêtre sa taille minimale (rappelez-vous le TP précédent).

e) Ajoutez une méthode `main` qui crée une `MontreFrame` et l'affiche. Compilez & testez.

Nous avons donc un cadran blanc avec un bord doré et les deux pignons au centre auxquels nous attacherons nos aiguilles un peu plus tard. Au même endroit que vous avez dessinez les cercles pour le cadran de la montre, vous allez ajouter le code pour graduer le cadran des minutes et des heures. Cela doit se faire après le dessin du disque blanc sinon vous ne verrez rien !

Exercice 6.3 Graduations

a) Dessinez 60 petits points rouges pour les minutes, sur un cercle de rayon 40. (on a vu comment choisir une couleur et dessiner un point à la question **6.1.c**)

b) Dessinez 12 carrés à bord noirs pour les heures, eux aussi centrés sur un cercle de rayon 40. (on a vu comment choisir une couleur et dessiner un carré à la question **6.1.c**)

c) Compilez & testez.

C. De fil (du temps) en aiguille (de montre).

Avant de dessiner les aiguilles, il nous faut trouver un moyen de savoir où en est le fil du temps. Pour connaître l'heure nous allons regarder la documentation (on sait que vous aimez ça).

Exercice 6.4 **Que dit la documentation de l'API sur les `Calendar`?**
a) Dans quel `package` se trouve la classe `Calendar` ? Comment créer un objet de type `Calendar` ?
b) Quelle méthode et quel paramètre doit-on utiliser pour récupérer le nombre de minutes de l'heure courante en utilisant un objet de type `Calendar` ?
c) Quelle méthode et quel paramètre utiliser pour récupérer le nombre d'heures ? Quelle est la différence entre les paramètres possibles pour récupérer l'heure et lequel choisir ?
d) Testez ces deux méthodes au top-level.

Vous avez maintenant tous les outils pour dessiner les aiguilles, alors à vos pinceaux !

Exercice 6.5 **Dessiner les aiguilles**
a) Dessinez un trait rouge de longueur 38 juste après le dessin du petit disque rouge. Ce trait sera l'aiguille des minutes, à vous de bien le positionner.
b) Dessinez un trait noir de longueur 20 juste après le dessin du petit disque noir. Ce trait sera l'aiguille des heures, à vous de bien le positionner.
c) Compilez & testez... vous devez pouvoir lire l'heure.

Il reste un problème : la montre ne se met pas à jour toute seule mais uniquement lorsqu'elle est affichée. Donc si vous voulez l'heure exacte, il faut, par exemple, réduire la fenêtre puis la réagrandir. Cela est dû au fait que nous n'affichons les aiguilles que lorsque la méthode `paint` est appelée. Si la fenêtre n'est pas recouverte, ou sa taille modifiée, cette méthode n'est pas appelée et notre montre n'avance pas. Afin de pallier à ce défaut tout en évitant d'utiliser des méthodes de programmation trop compliquées (telles que les `Threads`) nous allons tricher:

Exercice 6.6 **Que dit la documentation de l'API sur la méthode `repaint` pour les `canvas` ?**
a) Regardez la méthode `repaint` pour les canvas. De quelle classe est-elle héritée ? Quels sont les arguments possibles ? Laquelle vous paraît intéressante pour résoudre notre problème de remise à jour régulière de l'affichage ?
b) Ajoutez l'appel à `repaint` pour qu'après chaque affichage on programme un autre affichage pour dans 1 seconde.
c) Compilez & testez : votre aiguille doit bouger maintenant. (si si... un peu de patience)
d) L'affichage étant rafraîchit toutes les secondes, vous pouvez ajouter un trotteuse bleue de longueur 40.

D. Le temps moderne: l'affichage digital.

Ayant dessiné les aiguilles, nous allons maintenant ajouter un label pour afficher l'heure au format numérique HH:MM. Le problème c'est que l'on souhaite placer cet affichage sous la montre. Or jusqu'à maintenant on a utilisé un `FlowLayout` qui place les éléments sur la même ligne.

Exercice 6.7 **Que dit la documentation de l'API sur les différents `Layout`?**
a) Quels sont les différents `Layout` disponibles ? (commencez par regarder l'interfaces `LayoutManager`)
b) Nous voulons juste empiler deux composants verticalement (le `canvas` et le `Label`), quel `Layout` choisir ? Regardez son constructeur et vérifiez son `package` ! (ici nous allons utiliser `swing` car c'est plus simple)

Exercice 6.8 **Ajouter l'affichage digital.**
a) Modifiez le `Layout` de votre `Frame` et ajoutez un `Label` initialisé avec "00:00" et dont le contenu est centré. Puisqu'il affiche l'heure, ce label sera modifié nous allons donc le déclarer comme variable d'instance : `labelTemps`.
b) Faites en sorte que l'affichage du `Label` soit remis à jour en même temps que les aiguilles. Compilez & testez !
c) Il y a un petit truc qui cloche : il manque le zéro devant les minutes et les secondes lorsqu'elles sont inférieures à 10 L'affichage ressemble à "8:5:1". Essayez de trouver comment palier à ce problème.

E. Le temps d'affichage: attention peinture fraiche !

Vous remarquerez peut-être que l'image de la montre "saute/flash" de temps en temps. Cela est dû au fait que le système utilise la méthode `update` qui efface la fenêtre et appelle `paint`. Or sur certaines machines, le balayage de l'écran est plus rapide que notre programme et commence à rafraîchir l'écran alors que nous n'avons pas fini de dessiner la montre, laissant voir un morceau du fond de la fenêtre. La méthode `paint` est appelée lorsque toute la fenêtre doit être dessinée sinon le système appelle uniquement `update`. La signature de la méthode `update` est la même que celle de `paint`.

Exercice 6.9 **Surcharger `update`**
a) Déplacez le code de dessin dans la méthode `update` sauf le cadre jaune qui est inchangé lors des rafraîchissements.
b) Compilez & vérifiez que les flashs ont disparu.