Dessin, création de composant et modèle

Exercice 1 - Création d'un composant

Le but de cet exercice est de créer un composant et de dessiner dessus.
Dans un premier temps, nous allons étudier le dessin en utilisant les classes `Graphics` et `Graphics2D`

1 Créer un composant `JPortView` héritant de `JComponent` et redéfinissant la méthode `paintComponent()`.
Dessiner un rectangle bleu au centre du composant (la méthode `drawRect` de `Graphics` puis `draw` de `Graphics2D` en passant un rectangle en paramètre).

2 Mettre le composant au centre du `contentPane` de la frame. Que se passe-t'il ? Pourquoi ? Comment corriger ?

3 Modifier la méthode `paintComponent()` pour afficher un rectangle avec un dégradé du bleu vers le rouge. (Utiliser la méthode `setPaint()` de `Graphics2D` en passant en paramètre un objet de la classe `GradientPaint`)

4 Modifier la méthode `paintComponent` pour afficher le rectangle penché avec une rotation de 45 degrés. (la méthode `rotate()` sur `Graphics2D` ou `AffineTransform`).

Vous pouvez tester d'autres transformations : `rotate, scale, shear, translate`.

Exercice 2 - Visualisateur de switch

On cherche à créer un composant permettant d'afficher si les ports d'un switch sont activés (UP) ou non (DOWN). Nous allons pour cela programmer le composant suivant le principe du MVC : Modèle/Vue/Controleur.

- Ecrire un modèle (l'interface `PortModel`) permettant de refléter l'état (activé ou non) d'un nombre fixé de ports.
- Ecrire une implémentation (la classe `DefaultPortModel`) qui possède un constructeur capable d'indiquer le nombre de ports, les méthodes de l'interface `PortModel` et une méthode permettant de changer l'état d'un port dans le modèle.
- Enfin, implémenter un composant nommé `JPortView` qui se comportera comme une vue du modèle. Avec un constructeur prenant en paramètre le modèle et une méthode `paintComponent` affichant les ports sous forme de rectangle vert ou rouge en fonction de leur état.

Créer un exemple visualisant un switch avec 4 ports avec tous les ports "UP" sauf le port 2.

Exercice 3 - Visualisateur de switch (2)

Réutiliser le programme précédent et ajouter une serie de boutons à état (`JToggleButton`) permettant de contrôler l'état du switch.

1 Faire en sorte qu'il y ait autant de boutons à état que de ports. Ajouter, de plus, des listeners à chaque bouton pour que lorsque celui-ci soit enfoncé, l'état du port correspondant passe à "UP" (en vert).
PS: si vous avez des problèmes de raffraichissement, passer à la question suivante.

2 Ajouter un mécanisme de listener (`PortListener`) au modèle pour informer la vue

d'un changement effectué sur le modèle.

Exercice 4 - Timer et SwingUtilities

On cherche à ajouter au visualisateur de switch un bouton permettant d'effectuer un "reset" du switch.
Lorsque l'utilisateur cliquera sur le bouton "reset", celui-ci deviendra inutilisable (`setEnabled()`) et un compteur se déclenchera, celui-ci affichera à la place du texte du bouton "reset" le temps restant. Passé un délai de 10 secondes, tous les ports du switch passeront en état "DOWN".
Il existe deux façons de faire cela :

1. Utiliser un `Thread`, la méthode `sleep()`, les méthodes `invokeAndWait()` et `invokeLater()` de la classe `SwingUtilities`.
2. Utiliser plutôt la classe `java.awt.Timer` et ses méthodes `start()` et `stop()`.