Série E : Interfaces

Table des matières

Série E : Interfaces....1
Exercice E.1: FractionList....1
Exercice E.2: Shapes & Swing Component (Comparable, Computable, Iterator)....1
Exercice E.3: Listeners....2
Exercice E.4: MiniDraw....2

Exercice E.1: FractionList

1. Reprenez l'exercice **FractionList.** Faites en sorte qu'une liste de fractions puisse être triée simplement par la méthode **sort** prédéfinie dans **java.util.Collections**.
2. Sachant que **ArrayList** implémente **Iterable**, parcourez la liste avec son itérateur à toutes les occasions dans **FractionList**.
3. Modifiez la classe **Fraction** pour qu'on puisse employer la méthode **clone** (de façon publique). Profitez de la méthode **clone** héritée. Testez la réalisation de **clone** pour la classe **Fraction**. Modifiez, supprimez la fraction originale. Que se passe-t-il dans la copie 'clonée' ?
4. Modifiez la classe **FractionList** pour qu'on puisse employer la méthode **clone** (de façon publique). Profitez de la méthode **clone** héritée. Testez la réalisation de **clone** pour la classe **FractionList**. Modifiez, ajoutez des éléments dans l'original. Que se passe-t-il dans la copie 'clonée' ? Comment peut-on y remédier (consultez aussi l'Internet à ce sujet).

Exercice E.2: Shapes & Swing Component (Comparable, Computable, Iterator)

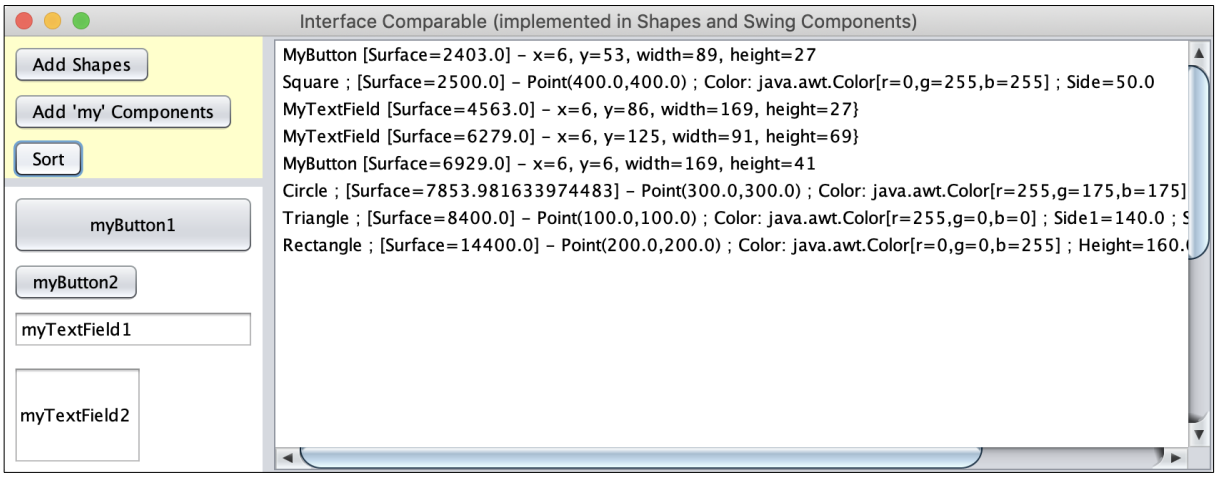
- Réalisez les classes **`Shape`**, **`Circle`**, **`Square`**, **`Rectangle`**, **`Triangle`**. Chaque figure a une position (point supérieur gauche : x,y) et une couleur. Ces attributs et les dimensions (rayons, longueurs de côtés) sont fixés lors de la construction.
- Chaque figure peut afficher son état avec **`toString`**.
- Toutes les figures implémentent l'interface **`Computable`** qui possède les méthodes **`getSurface()`** et **`getPerimeter()`.**
- Ajoutez une classe **`MyTextField`**, dérivée de **`JTextField`**, qui implémente l'interface **`Computable`**.
- Ajoutez une classe **`MyButton`**, dérivée de **`JButton`**, qui implémente l'interface **`Computable`**.
- **`Computable`** doit implémenter aussi l'interface **`Comparable`** basé sur la surface des objets.
- Réalisez la classe **`Computables`** qui sert à gérer une liste **`alComputables`** du type **`ArrayList`**. On peut trier les éléments dans la liste d'après leur ordre interne (c.-à-d. défini par **`Comparable`**).
- Faites en sorte que les éléments de **`Computables`** se laissent parcourir à l'aide d'un itérateur. Testez ceci en utilisant une boucle '*for ... each*'.
- Testez le programme en ajoutant différentes figures ainsi que des **`MyButton`** et **`MyTextField`** à **`Computables`**. (Affichez les éléments et leurs propriétés).

Version 2 : Array & interfaces

>>> Familiarisez-vous d'abord avec les tableaux statiques (**arrays**) <<<
>>> dans le chapitre 'Structures de données' <<<

- Faites une copie du programme et remplacez la structure **`ArrayList`** dans **`Computables`** par un tableau statique **array** nommé **`arComputables`**. Définissez et utilisez un attribut **size** comme dimension effective pour le tableau.
- On peut trier les éléments du tableau d'après leur ordre interne (c.-à-d. défini par **`Comparable`**). Profitez de méthodes prédéfinies dans **`Arrays`**.
- Faites en sorte que les instances de **`Computables`** se laissent aussi parcourir à l'aide d'un itérateur. Définissez et réalisez vous même cet itérateur! Testez-le en utilisant une boucle '*for ... each*'.

Exemple pour une application de test :



Exercice E.3: Listeners

Créez un nouveau projet vide en NetBeans avec un bouton placé sur une fiche.

Regardez le code généré par NetBeans (qui est normalement caché).

Créez un événement pour le cas où on clique sur le bouton. Vérifiez le code généré. Que s'est-il passé ? Recherchez de l'aide sur **ActionListener** et essayez d'expliquer le code généré par NetBeans.

Exercice E.4: MiniDraw

Dans notre exercice **MiniDraw**, nous avons défini une méthode **draw** pour toutes les figures à dessiner. Pourquoi est-ce qu'il n'est pas utile de définir un interface **Drawable** dans notre exercice **MiniDraw** pour dessiner les figures ?

A partir de quel moment est-ce que ce serait utile de définir un tel interface ? Donnez un exemple.

Quelles transformations faudrait-il faire dans la classe **FigureList** ?