Polymorphisme

Cours

Les formes géométriques
Classe abstraite AFormeGéométrique

Une forme géométrique est un objet qui a

- une position d'ancrage que l'on peut lire et modifier ;
- une aire ;
- un périmètre.

Sur le modèle donné en cours, proposer une classe abstraite **`AFormeGéométrique`** qui gère la lecture et la modification du point d'ancrage.

Les formes pré-définies

Proposer une classe Java pour réaliser les formes géométriques suivantes :

- rectangle ;
- carré ;
- cercle ;
- ellipse.

Note: l'aire d'une ellipse est donnée par PI.a.b où a et b sont les demi-petits et grands axes. Le périmètre est plus difficile, on considèrera l'approximation suivante: PI.sqrt(2(a^2+b^2)).*

Application

Proposer une application Java qui teste un scénario possible dans lequel 4 formes géométriques sont créées (une de chaque) et qui affiche sur la console les informations sur chacune de ces formes.

Oui vous avez compris, il faut redéfinir la méthode **`toString()`** des classes concrètes, mais desquelles ? Si vous ne savez pas ce que veut dire redéfinir, voir le cours.

Il n'y a pas de tableau ou de liste dans cet exercice !

Applet

Proposer une applet Java qui teste un scénario possible dans lequel 4 formes géométriques sont créées (une de chaque) et sont dessinées dans un navigateur (ou plutôt avec l'appletviewer).

Pour cela:

- Créer une classe qui hérite de Applet (voir le premier cours)
- Etudier attentivement la classe java.awt.Graphics
- Ajouter une méthode **`dessineToi`** à toutes les formes géométriques.
    - Quelle est la signature de cette méthode ?
    - A quelle(s) classe(s) faut-il l'ajouter ?

Les couleurs

Modifier la classe **`AFormeGéométrique`** pour qu'elle gère un champ pour la couleur de fond et un champ pour la couleur de dessin.

- Quel est le type de ces champs ? (cf. la classe java.awt.Color)
- Quel est l'impact sur les classes concrètes ?

Les paramètres de centrage

Pour l'instant, nous avons supposé que le point d'ancrage était:

- le coin supérieur gauche pour le rectangle (et le carré);
- le centre pour l'ellipse (et le cercle).

On veut pouvoir choisir les propriétés d'ancrage et donc les alignements horizontaux et verticaux.

1. Proposer un typé énuméré qui permet d'aligner à gauche, de centrer ou d'aligner à droite.
2. Ajouter les champs et les méthodes nécessaires pour gérer l'alignement.
3. Quelles sont les méthodes et les classes qu'il faut modifier pour faire cela ?

Bien évidemment, à chaque fois il faut créer de nouveaux scénarios de test !

Les nombres complexes

Le codage

On veut encoder les nombres complexes avec les deux codages vus en cours (cartésien et polaire).

1. Implanter l'interface **`IComplexe`** vue en cours;
2. Implanter la classe **`ComplexeCartésien`** qui réalise l'interface **`IComplexe`** en utilisant un codage cartésien (partie réelle et partie imaginaire);
3. Implanter la classe **`ComplexePolaire`** qui réalise l'interface **`IComplexe`** en utilisant un codage polaire (module et argument).
4. S'assurer que chacune de ces classes redéfinit la méthode **`toString`**.

Les opérations

Réaliser une classe **`ComplexeUtil`** qui fournit les opérations suivantes :

1. création d'un nombre complexe à partir de ses coordonnées cartésiennes ;
2. création d'un nombre complexe à partir de ses coordonnées polaires ;
3. conversion d'un nombre complexe quelconque en un nombre complexe polaire ;
4. conversion d'un nombre complexe quelconque en un nombre complexe cartésien ;
5. comparaison de l'égalité arithmétique de deux nombres complexes ;
6. addition de deux nombres complexes (on ne sait pas quel est leur encodage);
7. soustraction de deux nombres complexes ;
8. multiplication de deux nombres complexes ;
9. division de deux nombres complexes.

Attention c'est à vous de choisir attentivement la signature et la façon d'implanter ces opérations pour vous faciliter la vie. Aucune de ces opérations ne devrait prendre plus de 5 ou 6 lignes de Java.
