CHAPUS 17/05/2023

TP n°3 Java

Louka 3A IE

Exercice 1 :

1./

Une image contenant texte, capture d’écran, Police, logiciel

Description générée automatiquement

Figure [1] classe FormeGeometrique

On déclare des méthodes abstraites, ainsi que l’épaisseur du trait en protected pour pouvoir y avoir accès dans les classes filles.

2./

On met l’attribut protected pour que les classes filles aient accès aux variables de la classe mère.

3./

Pour le constructeur, on initialise simplement l’épaisseur du trait puisque c’est le seul paramètre qui est commun aux deux classes

4./

Une image contenant texte, capture d’écran, Police, logiciel

Description générée automatiquement

Forme [2] classe Cercle

Une image contenant texte, capture d’écran, logiciel, Police

Description générée automatiquement

Forme [3] classe Rectangle

Là aussi on déclare simplement les classes avec les getters bon getters. Les deux classes héritent de la classe FormeGeometrique c’est pourquoi on appelle la fonction super() dans le constructeur.

5./

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Figure [4] méthodes de la classe Cercle

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Figure [5] méthodes de la classes Rectangle

A chaque fois on calcule le périmètre et la superficie des formes géométriques avec la bonne formule pour le rectangle et pour le cercle.

7./

Une image contenant texte, capture d’écran, Police, noir

Description générée automatiquement

Figure [6] erreur FormeGeometrique

Lorsqu’on essaye de déclarer des objets de type FormeGeometrique on obtient une erreur qui est logique car c’est une classe abstraite donc on ne peut pas créer d’objet.

8./

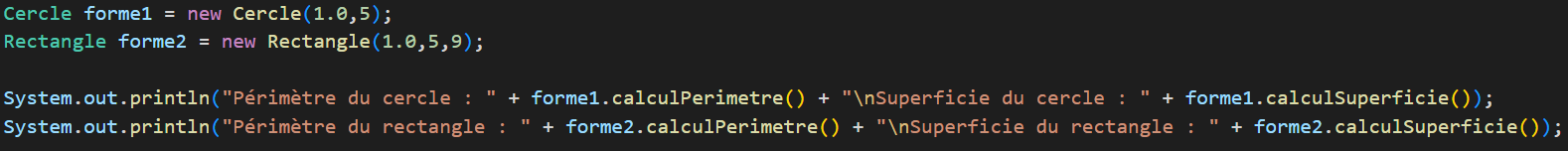


Figure [7] classe main

Une image contenant texte, Police, capture d’écran

Description générée automatiquement

Figure [8] Résultat après exécution du main

On voir bien qu’après exécution du main on obtient bien les bonnes valeurs pour les périmètre et pour les superficies.

9./

La compilation se passe bien mais normalement il faudrait redéfinir la fonction toString dans chaque classe. C’est donc ce qu’on va faire par la suite.

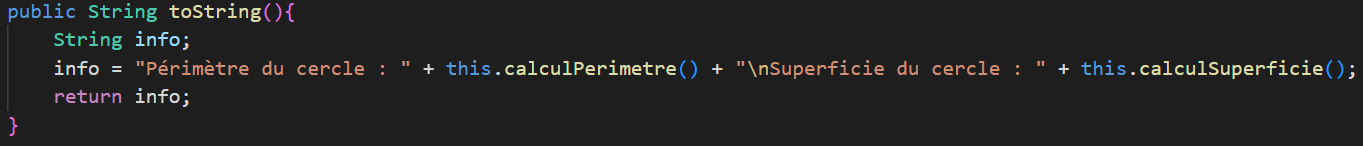


Figure [9] méthode de la classe Cercle

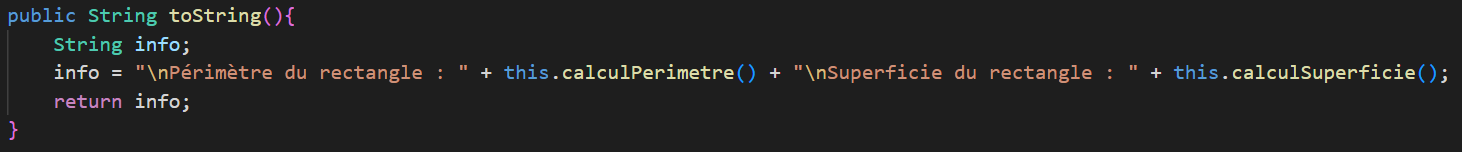


Figure [10] méthode de la classe Rectangle

On met juste bien en forme le texte avec les bonnes informations et le bon texte à renvoyer. Donc ensuite dans le main pourra simplement appeler la méthode toString et on aura toutes les information dans le bon formalisme.

Exercice 2 :

1./

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Figure [11] classe TableauFormeGeometrique

On pourrait mettre les variables en private cela reviendrait à la même chose.

2./

Une image contenant texte, capture d’écran, Police, ligne

Description générée automatiquement

Figure [12] Constructeur de la classe TableauFormeGeometrique

On initialise un tableau vide de longueur nbFormeGeometrique.

3./

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Figure [13] méthode pour ajouter une forme géométrique

Pour ajouter une forme on parcourt tout le tableau et si l’élément est vide alors on met la forme à cet emplacement et on arrête la boucle pour ne pas remplir le tableau avec le même élément.

4./

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Figure [14] Main

Une image contenant texte, capture d’écran, Police, logiciel

Description générée automatiquement

Figure [15] Méthode de la classe TableauFormeGeometrique

Pour la fonction toString on appelle simplement les fonctions toString sur chaque éléments du tableau, cela nous permet de réutiliser le code précédent.