Cours 420-202-RE Traitement de données orienté objet Hiver 2021 Cégep Limoilou Département d'Informatique	Tp 2 (6%) Héritage et énumération

OBJECTIFS

- Définir et utiliser des classes liées par le principe de l'héritage.
- Commenter des classes avec de la JavaDoc
- Tester unitairement des classes
- Connaître et utiliser les exceptions.

À REMETTRE à la date indiquée sur Léa (Vous avez 2 semaines)

- Code source respectant les consignes, utilisant efficacement les **packages**, la programmation orientée objet, l'héritage et les exceptions.
- Tout votre code documenté avec la JavaDoc générée.
- Les tests unitaires (JUnit) pour toutes les classes développées.
- Projet complet compressé (.zip) dans un fichier du nom « votre nom – Tp2.zip » et déposé sur LÉA.

ACTIVITÉS À RÉALISER

Comme vous avez pu le constater, vos classes Rectangle et Cercle ont des comportements et des propriétés en commun. Vous allez commencer par répertorier les propriétés et les méthodes qui sont partagés par Rectangle et Cercle.

IMPORTANT : Avant de commencer votre travail vous allez créer une copie du projet du « Tp 1 » dans un nouveau projet « Tp 2 ». Il faut terminer le « Tp 1 » avant, s'il ne l'est pas !

À partir du nouveau projet « Tp 2 » :

1. Vous allez construire, dans le package formes, une classe **abstraite** « Forme » qui contient les propriétés et les comportements communs aux formes géométriques. Voici ce que vos formes doivent offrir, il vous faut placer les éléments dans les bonnes classes.
   - Chaque forme a une couleur choisie dans une énumération. Vous allez transformer votre liste de couleur (rouge, vert, bleu, jaune, noir, orange) en énumération. Cette énumération possède un attribut et quelques méthodes. Voir le diagramme de classe fourni ainsi que les exemples Java et les notes de cours fournis. La couleur par défaut d'une forme est rouge.
   - Chaque forme a un nom qui la représente (Cercle, Triangle, Rectangle). Les autres paramètres obligatoires seront spécifiques à chaque forme (rayon, longueur des côtés, etc.).
   - Chaque forme a la possibilité de se **comparer** en mettant en œuvre l'interface « Comparable ». Les formes se comparent par leur **nom** et par leur **couleur**. Cela nous permet de trier les formes selon leur forme (nom) et si le nom de la forme est identique, la couleur est prise en considération. Penser à utiliser les méthodes de comparaison d'une « String ». (Ex. cerclebleu est inférieur à cerclevert)

     Très important : Prendre connaissance de la documentation Java au sujet de la méthode d'interface Comparable, « compareTo » : http://docs.oracle.com/javase/8/docs/api/java/lang/Comparable.html
   - Comme présenté, un constructeur affecte les valeurs aux attributs en prenant soin de valider les paramètres avant de construire l'objet. La couleur par défaut de toutes les formes est "rouge" et le nom est celui qui lui est associé parmi (Rectangle, Cercle ou Triangle). Si le ou les paramètres obligatoires (dimension de la forme) sont invalides, l'objet ne doit pas être construit et une exception "FormeException" est levée.

- Lorsque l'on affecte une couleur à une forme (setCouleur), si cette couleur n'est pas valide (« null » par exemple), la forme conserve sa couleur. Il **ne faut plus** lui affecter la couleur par défaut.
- Il y a les accesseurs, mutateurs et méthodes de validation nécessaires pour gérer la forme.
- Il y a une méthode **calculerSurface** qui calcule la surface de la forme.
- Il y a une méthode **calculerPerimetre** qui calcule le périmètre de la forme.
- Il y a une méthode **toString**() pour faire afficher les attributs sous forme d'une chaîne de caractères. Par exemple :

  `Cercle bleu 5` (où 5 est le rayon)

  `Rectangle rouge 3, 4` (où 3 est la hauteur et 4 est la largeur)

  `Triangle jaune scalène 5, 7, 10` (où 5 est coteA, 7 est coteB et 10 est coteC)
- Il faut redéfinir la méthode **boolean equals(Object obj)** qui permet de vérifier l'égalité entre deux formes. Deux formes sont réputées égales si le **nom**, la **couleur** et la **surface** sont égaux.

2. Vous allez modifier vos classes **Rectangle** et **Cercle** pour qu'elles héritent de la classe **Forme**. La classe Cercle n'a plus de constructeur par défaut. (voir diagramme de classes)

3. Vous allez adapter vos tests unitaires JUnit pour que vos classes de tests testent l'ensemble des méthodes publiques des classes **Forme**, **Rectangle** et **Cercle**.

4. Mettre à jour et adapter la JavaDoc dans vos classes **Cercle**, **Rectangle** et **Forme**.

5. **ATTENTION**, pour alléger votre tâche, profitez efficacement du menu Refactor.

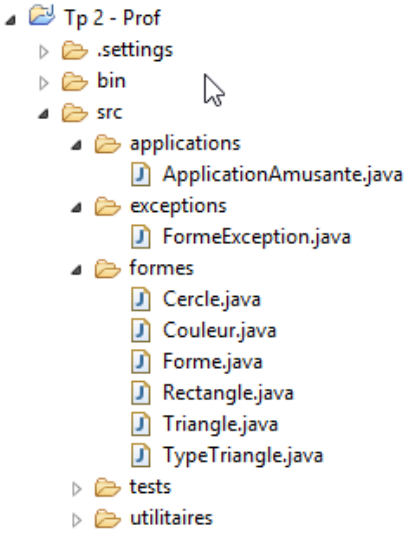
6. Vous allez construire une classe **Triangle** complète (testée avec JUnit et documentée) qui hérite de **Forme** et qui comporte les caractéristiques suivantes :

   - Un attribut *coteA*, un entier entre 1 et 30 cm.
   - Un attribut *coteB*, un entier entre 1 et 30 cm.
   - Un attribut *coteC*, un entier entre 1 et 30 cm.
   - Lors de la construction, si un des côtés est invalide, une exception est levée avec un message approprié, et si ce n'est pas un triangle (voir méthode estTriangle), une exception est aussi levée avec un message approprié. (Donc 2 exceptions différentes)
   - Pour calculer la **surface** (en int) d'un triangle une petite recherche s'impose… à vous de trouver.
   - Le document « Angles et Triangles.pdf » est en référence dans le dossier du travail pour vous permettre de construire certains algorithmes.
   - Une méthode qui permet de définir le type de triangle (rectangle, isocèle, scalène, équilatéral). Le type de triangle sera une énumération (comportant un attribut et quelques méthodes). Cette méthode devrait se servir des méthodes privées getNbrCoteEgaux() et estRectangle().
   - Une méthode qui permet de valider si nous sommes en présence d'un vrai triangle (selon la longueur de ses côtés).
   - Pour la méthode retournant les côtés triés vous pouvez chercher dans la classe Arrays. Cette méthode pourrait être utilisée dans la méthode estRectangle().
   - N'oubliez pas de faire la JavaDoc et les tests JUnit sur la nouvelle classe Triangle.

7. Ajustez vos tests en fonction des ajouts et des ajustements faits et exécutez vos JUnit pour vous assurer que tous les tests passent. Votre couverture de tests doit être suffisante, plus de 85%.

PACKAGES et DIAGRAMME DE CLASSES

Packages :



Critères de correction :

Respect des consignes	**/ 3**
JavaDoc	**/ 15**
Votre code	**/ 41**
Vos tests unitaires	**/ 20**
Tests du profs (conformité et fonctionnement)	**/ 36**
Total	**/ 115**

Diagramme de classes :