Cours 420-202-RE Traitement de données orienté objet Hiver 2020 Cégep Limoilou Département d'Informatique	**Tp1 (4 %)** **Classes, exceptions, JavaDoc et tests unitaires**

OBJECTIFS :

- Définir et utiliser des classes.
- Commenter des classes avec de la JavaDoc
- Tester unitairement des classes avec JUnit.
- Utiliser les exceptions.

DURÉE DU LABORATOIRE :

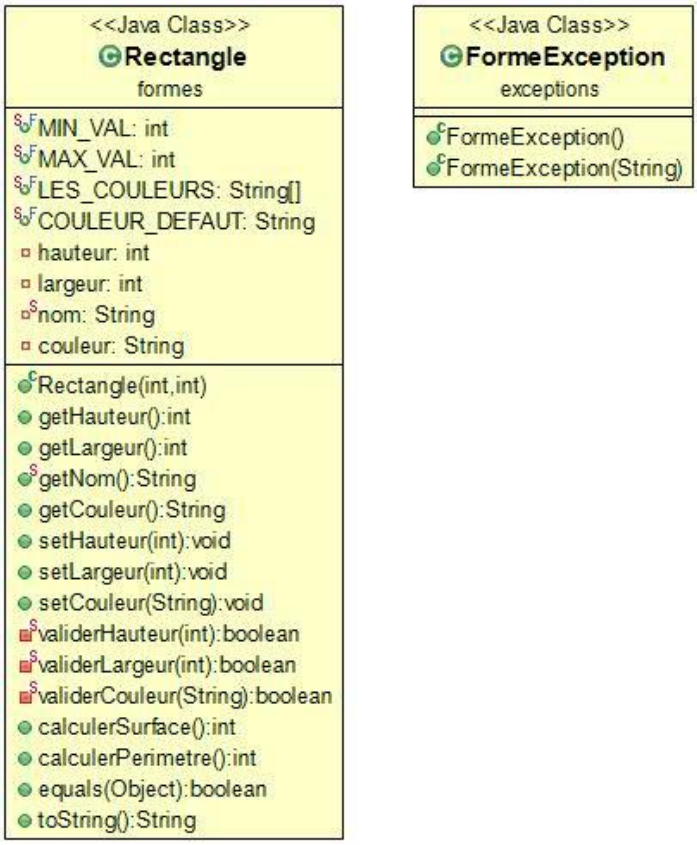
- **2 semaines** sont consacrées à ce travail. Le travail est fait individuellement.

ACTIVITÉS À RÉALISER :

Dans un projet du nom de **« votre nom – Tp1 »**, créez deux classes pour représenter deux formes géométriques « **Cercle »** et « **Rectangle »** selon les consignes suivantes.

1. Vous allez créer une classe **Rectangle**, dans un package **formes** et utiliser une classe d'exception personnalisée, dans le package **exceptions,** afin de construire un rectangle valide. Vous allez également tester unitairement votre classe à l'aide de JUnit.

   Voici le diagramme de classe **Rectangle** suivi d'explications pour certaines méthodes :

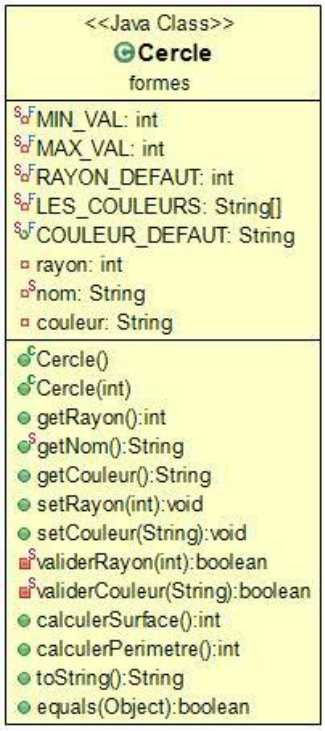


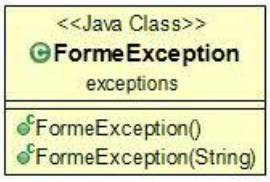
Les attributs : • **Attention !** Le nom est « Rectangle » pour **tous** les objets Rectangle. • La hauteur et la largeur sont entre 1 et 30 (bornes incluses) • La couleur est l'une des couleurs suivantes : rouge, vert, bleu, jaune, noir, orange. Ces valeurs (en minuscules) sont dans un vecteur constant. "rouge" est la couleur par défaut.
Constructeur : Si les paramètres sont valides, on les assigne ainsi que nom et couleur (rouge). Si une des valeurs est **invalide** on lève une exception de type **FormeException,** qui est une classe du package **exceptions**.
setCouleur : Exceptionnellement, si la couleur est invalide, on affecte la couleur « **rouge** ». Si la couleur reçue contient des espaces (avant ou après) ou est en majuscules elle doit être considérée comme valide. Exemple : Si on reçoit " BLEU " l'attribut devient "bleu".
validerCouleur : Si la couleur reçue contient des espaces (avant ou après) ou est en majuscules elle doit être considérée comme valide. Exemple : Si on reçoit " BLEU " le paramètre devient "bleu". **Optimiser votre recherche**.
toString : retourne l'information de l'objet sous cette forme : nom couleur hauteur, largeur (dans l'ordre) : **ex : Rectangle rouge 3, 4**
equals : retourne vrai si deux rectangles sont égaux. 2 rectangles sont égaux, si leur **surface** et leur **couleur** sont égales.

2. Vous devez tester unitairement chaque méthode publique de votre classe Rectangle dans la classe **RectangleTest** dans le package **tests** à l'aide de JUnit.

3. Documentez **obligatoirement** vos classes avec de la JavaDoc (voir notes de cours).

4. Vous allez construire une seconde classe **Cercle**, dans le package formes, complète (testée avec JUnit et documentée). Les tests de la classe **Cercle** se font dans la classe **CercleTest** du package **tests**. Voici le diagramme de classe Cercle suivi d'explications pour certaines méthodes :



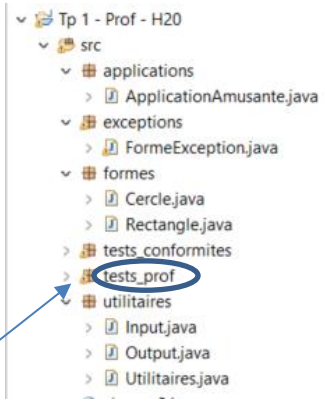


Les attributs : • Le nom est « Cercle » pour **tous** les objets Cercle. • Le rayon est entre 1 et 30 (bornes incluses) • La couleur est l'une des couleurs suivantes : rouge, vert, bleu, jaune, noir, orange. Ces valeurs (en minuscules) sont dans un vecteur constant. "vert" est la couleur par défaut.
Constructeurs : Le rayon par défaut est 10. Si le rayon est valide, on assigne sa valeur ainsi que nom et couleur (vert). S'il est **invalide** on lève une exception de type **FormeException,** qui est une classe du package **exceptions**.
setCouleur : Exceptionnellement, si la couleur est invalide, on affecte la couleur « **vert** ». Si la couleur reçue contient des espaces (avant ou après) ou est en majuscules elle doit être considérée comme valide. Exemple : Si on reçoit " BLEU " l'attribut devient "bleu".
validerCouleur : Si la couleur reçue contient des espaces (avant ou après) ou est en majuscules elle doit être considérée comme valide. Exemple : Si on reçoit " BLEU " le paramètre devient "bleu". **Optimiser votre recherche**.
toString : retourne l'information de l'objet sous cette forme :nom couleur rayon **ex : Cercle orange 20**
equals : retourne vrai si deux cercles sont égaux. 2 cercles sont égaux si leur **surface** et leur **couleur** sont égales.
Pour la surface et le périmètre du cercle, on retourne la partie entière du résultat.

5. Dans le dossier du TP1 sur Léa, vous trouverez un dossier src. Il contient dans les packages « applications » et « utilitaires » du code fonctionnel qui utilise vos classes « Rectangle » et « Cercle ». Copiez ces packages dans votre projet. L'application « ApplicationAmusante » devrait être fonctionnelle sans modifications.

ATTENTION :

- Utilisation de constantes.
- Diviser votre application en packages selon la demande.
- Utilisation de modificateurs d'accès appropriés (public, private).
- Une méthode = une action (sinon on divise la méthode en plusieurs, on factorise).
- Utilisation de noms significatifs pour « auto documenter ».
- **<u>Écrire une méthode et la tester immédiatement.</u>**
- Le nombre de tests réussis témoignent de l'avancement.
- Au lieu de tests_prof, vous aurez **tests** contenant 2 classes : CercleTest.java et RectangleTest.java



À REMETTRE :

- Au début du laboratoire, voir date sur Léa.
- Code source (50 %) respectant les consignes, utilisant efficacement les **packages**, la programmation orientée objet et les exceptions.
- **Tout votre code documenté avec génération de la JavaDoc. (15 %)**
- Les tests unitaires (JUnit) pour les classes développées (Rectangle et Cercle). (30 %)
- **<u>Projet</u>** complet compressé (.zip) dans un fichier du nom « votre nom – Tp1.zip » et déposez le sur Léa.
- Respect des consignes (5 %)