CMO
Classes et objets

Steven Costiou
Inria

June 9, 2021

1 Gestion d'exceptions

Dans cet exercice, nous allons scanner des entrées utilisateur sur System.in pour créer des formes géométriques. Nous allons:

- Utiliser le flux d'entrée standard pour récupérer des commandes venant des utilisatrices,
- gérer les entrées non reconnues via des exceptions.

Pour récupérer une entrée, vous pouvez utiliser le code suivant :

```
Scanner commandScanner = new Scanner(System.in);
String input = commandScanner.nextLine();
```

Exercice 1. Nous souhaitons manipuler les entrées utilisateurs au travers d'un objet dédié. Créez une classe CommandLineInterface qui:

1. Contient une variable scanner qui référence le flux d'entrée standard,
2. contient une méthode scanCommand() qui lit une ligne sur le flux d'entrée et retourne sa valeur sous forme de chaine de caractères.

Créez une petite classe de test contenant une méthode *stCode* et créez un objet CommandLineInterface afin de le tester empiriquement.

Exercice 2. Nous allons maintenant créer des formes géométriques à partir d'entrées venant des utilistrices. Dans cet exercice, vous prendrez soin de gérer les cas spécifiques (mauvaise type en entrée, forme géométrique inconnue...). Dans tous les cas, une exception due à une mauvaise entrée utilisateur ne doit **jamais** provoquer l'arrêt ou le dysfonctionnement de votre programme.

1. Programmez une classe GeoFormCreator qui prend en paramètre une chaine de caractères correspondant au nom d'une forme géométrique,
2. en fonction du nom de la forme géométrique, utilisez une nouvelle interface de commande pour demander à l'utilisatrice les informations nécessaires pour la création de la forme géométrique,
3. instanciez la forme géométrique demandée et affichez là dans une fenêtre.

Vous testerez empiriquement votre code.

Note : Comme les données entrées par l'utilisateur sont des chaines de caractères encodant des entiers ou des doubles ("10.0" par exemple), il faut les transformer en valeurs de type entier ou double (10.0). Pour cela utiliser la méthode static Double.parseDouble(String s) de la classe Double, wrapper de double.

Exercice 3. Modifiez votre programme pour demander à l'utilisateur un nombre arbitraire de formes géométriques. Mettez à jour votre gestion d'exception si nécessaire.

2 Booléens : exercice d'implémentation orienté objet

En guise d'exercice, nous allons réimplémenter une classe pour représenter les booléens true et false.

Notre classe principale sera CMO.Boolean, qui aura deux sous classes: CMO.True et CMO.False. CMO.Boolean ne possède que des méthodes abstraites, c'est-à-dire devant être redéfinies par ses sous classes :

1. value() : renvoie la valeur du booléen (true ou false).
2. not() : renvoie l'inverse de la valeur du booléen.
3. and(CMO.Boolean bool) : opérateur booléan "*and*".
4. or(CMO.Boolean bool) : opérateur booléan "*or*".
5. xor(CMO.Boolean bool) : opérateur booléan "*xor*".
6. toString() : imprime "*CMO.True*" ou "*CMO.False*" sur la sortie standard.

Écrivez une classe de test dans laquelle vous instanciez des booléens et vous vous assurez que les méthodes implémentées renvoie un résultat correct.