Séance 3: TESTS DU PROGRAMMEUR

Université de Paris

Objectifs:

- Tester des fonctions selon un contrat
- Utiliser les conditionnelles et les boucles
- Manipuler le type `int` des entiers
- Manipuler le type `String` des chaînes de caractères

L'une des grandes problématiques de la programmation informatique est le "bug", c'est-à-dire un programme qui ne termine pas ou qui ne renvoie pas le résultat attendu.

Le résultat attendu d'un programme est généralement spécifié sous la forme d'un *contrat* : il spécifie quelle est la forme des entrées que le programme attend et ce que le programme garantit alors sur ses sorties.

Il existe des techniques de preuve mathématique permettant de s'assurer qu'un programme respecte son contrat. Cependant, ces techniques ne sont pas au programme d'IP1.

Une autre méthode, moins sûre mais très répandue, permet de se convaincre de la correction du programme en faisant passer au programme un grand nombre de *tests*.

Un *test* est la donnée d'un couple formé d'une entrée I et d'une sortie attendue O respectant le contrat. Pour effectuer un test sur une fonction, on appelle cette fonction avec le paramètre I et on vérifie que la valeur de retour de la fonction est égale à la sortie attendue O définie par le test.

Important : les fonctions testées dans ce TP sont fournies mais il ne faut pas les modifier. Les tests sont en effet à programmer dans le `main`. En pratique, les fonctions ou les programmes testés ne sont pas toujours connus par le testeur. D'ailleurs, à la fin de ce TP, vous verrez comment en testant une fonction on peut arriver à comprendre ce qu'elle calcule.

Exercice 1 (Valeur Absolue, $\star$)
Le fichier `AbsoluteValue.java` *propose deux fonctions* `myAbs` *et* `newAbs` *qui prennent en paramètre un entier et retournent sa valeur absolue. Le but de cet exercice est de les tester. Les tests seront à placer dans la fonction* `main` *de ce fichier.*

1. *À l'aide de la procédure* `System.out.println`*, afficher dans le terminal la valeur de retour de la fonction* `myAbs` *évaluée sur l'entier* 0.
2. *On souhaite vérifier que la fonction* `myAbs` *vérifie le contrat suivant :*

   Contrat:

   $$\begin{array}{lcl} x = \text{-}10 & \rightarrow & 10 \\ x = 0 & \rightarrow & 0 \\ x = 1 & \rightarrow & 1 \end{array}$$

   Recopiez et modifiez la suite d'instructions :

```
System.out.println("Entrée : " + String.valueOf(0));
System.out.println("Sortie myAbs : " + String.valueOf(myAbs(0)));
```

pour afficher le résultat de chaque test de manière lisible :

```
Entrée : -10
Sortie myAbs : 10
Entrée : 0
Sortie myAbs : 0
Entrée : 1
Sortie myAbs : 1
```

3. *A l'aide d'une boucle, répétez ces tests pour toutes les entrées entre -10 et 10 inclus.*
4. *Testez les entiers entre -10 et 0 d'une part, et entre 1 et 10 d'autre part, à l'aide de deux boucles, en précisant désormais dans l'affichage si chaque test est positif par rapport au contrat suivant*

   Contrat:
   Pour tout entier `i>=0` *passé en paramètre, la valeur de retour de la valeur absolue est* `i`.
   Pour tout entier `i<0` *passé en paramètre, la valeur de retour de la valeur absolue est* `-i`.

   par exemple, si myAbs renvoyait -1 pour l'entrée 1 (mais ce n'est pas le cas), on afficherait

```
...
Entrée : -1
Sortie myAbs : 1 (valide)
Entrée : 0
Sortie myAbs : 0 (valide)
Entrée : 1
Sortie myAbs : -1 (erreur)
...
```

   On rappelle que la fonction `System.out.print`*, à la différence de la fonction* `System.out.println`*, ne passe pas à la ligne à la fin de l'affichage.*
5. *Modifiez votre code pour tester tous les entiers entre -100 et 99 inclus à l'aide d'une seule et même boucle, et si possible d'une seule conditionnelle.*
6. *Testez si la fonction* `newAbs` *vérifie le contrat suivant sur les paramètres compris entre* -10 *et* 100*, en affichant* `"ok"` *si le test est passé et un message d'erreur indiquant le paramètre qui ne passe pas sinon. Pour se faire, aidez-vous d'une boucle et d'une conditionnelle.*

   Contrat:
   Si la fonction prend en paramètre un entier, alors elle renvoie un entier positif.
7. *À l'aide d'une boucle et d'une conditionnelle, comparer les fonctions* `Math.abs` *(valeur absolue de Java) et* `newAbs` *sur les entiers compris entre* -200 *et* 199 *inclus, en affichant* `"ok"` *si les deux fonctions retournent la même valeur et* `"ko"` *sinon. À partir de quelle valeur les deux fonctions ne coïncident plus ?*

□

Exercice 2 (Miroir, ⋆⋆)
Le fichier `Mirror.Java` *contient deux fonctions* `reverse` *et* `myReverse` *permettant d'inverser une chaîne de caractères. Le but de cet exercice est de tester leur validité. Vous écrirez vos tests dans la fonction* `main` *de ce fichier.*

1. *À l'aide de la procédure* `System.out.println`*, afficher dans un terminal la valeur de retour des fonctions* `reverse` *et* `myReverse`*. Regarder si elles vérifient le contrat suivant :*

   Contrat:
   À chaque paramètre `s` *est associé la valeur de retour attendue d'une fonction encodant l'inversion d'une chaîne de caractères :*

s = *"Miroir"*	→	*"rioriM"*
s = *"a"*	→	*"a"*
s = *""*	→	*""*
s = *"avec des espaces"*	→	*"secapse sed ceva"*
s = *"elle"*	→	*"elle"*

2. *Vérifier le contrat suivant sur trois chaînes de caractères de votre choix.*

   Contrat:
   Inverser deux fois une chaîne de caractères revient à ne rien faire.
   `s.equals(reverse(reverse(s)))` *et*
   `s.equals(myReverse(myReverse(s)))` *doivent être vrais.*

   Utiliser une conditionnelle afin d'afficher "ok" si le test est vérifié et sinon, d'afficher la chaîne de caractères `s` *passée en paramètre.*

3. *Pour tester les fonctions sur plus d'exemples, on va utiliser le dictionnaire de la langue française. La fonction* `wordFromDict` *prend en paramètre un entier* i *compris entre* 0 *et* 336530 *et renvoie la chaîne de caractères correspondant au* $i^{ème}$ *mot du dictionnaire. À l'aide d'une boucle sur une dizaine d'entiers compris entre* 0 *et* 336530 *et d'une conditionnelle, afficher les valeurs de retour des fonctions* `reverse` *et* `myReverse` *sur quelques exemples.*

4. *Tester les fonctions* `reverse` *et* `myReverse` *sur des paramètres aléatoires en utilisant la fonction* `wordFromDict(i)` *et la fonction* `randRange(a, b)`*, qui renvoie aléatoirement un entier* `n` *compris entre les paramètres* `a` *et* `b` *:* `a <= n < b`.

5. *Plutôt que de vérifier à la main que la fonction renvoie le bon résultat, utiliser une boucle pour vérifier si les deux fonctions vérifient le contrat suivant sur un paramètre quelconque.*

   Contrat:
   Si sur le paramètre `s` *la fonction inverse retourne la chaîne* `r`*, alors les deux chaînes ont même longueur* ℓ *et pour* $0 \leq i < \ell$*,*
   `s.charAt(i)==r.charAt(`ℓ`-i-1)` *est vrai.*

6. *À l'aide de la question précédente et de boucles, tester la fonction* `myReverse` *sur dix mots du dictionnaire choisis aléatoirement.*

□

Exercice 3 (Mystère, ⋆⋆⋆)
Le fichier `Mystery.java` *contient 7 fonctions nommées* `mystere0`, `mystere1`,..., `mystere6` *qui prennent en paramètre soit un entier, soit une chaîne de caractères et renvoient, soit un entier, soit une chaîne de caractères. À l'aide de tests, déterminer leur comportement décrit sous forme d'un contrat le plus précis possible, puis vérifier votre hypothèse.* □