POLYTECH ANGERS

TP 4A - Génie Logiciel
Tests unitaires et preuve de programmes

Nicolas Delanoue

Objectif : Se familiariser avec le vocabulaire du test logiciel.

1 Preuve de programmes

Exercice 1

Lire le programme suivant :

```
def maximum(liste):
    maxi = liste[0]
    for i in liste:
        if i >= maxi:
            maxi = i
    return maxi
```

1. Écrire la spécification de ce programme.
2. Est-il possible que l'exécution de ce programme crée une erreur ? Corriger-le.
3. Prouver que ce nouveau programme vérifie la spécification.

Exercice 2 (Algorithme récursif pgcd)

1. Lire le programme suivant :

```
#!/usr/bin/python
def pgcd(a,b):
    """pgcd(a,b): calcul du 'Plus Grand Commun Diviseur'
    entre les 2 nombres entiers a et b"""
    if b==0:
        return a
    else:
        r=a%b
        return pgcd(b,r)

# Exemple d'utilisation:
print pgcd(70,42) # =>  affiche 14
```

2. Décrire la trace de l'exécution `pgcd(70,42)`.
3. Ecrire la spécification de ce programme.
4. Montrer la correction de ce programme.

Exercice 3 (Division par 0)

1. Le programme suivant contient une erreur :

```
#!/usr/bin/python
def inverse(x):
  if x=!0:
    x = 1
  y = 1/x
  return y
```

2. Donnez le graphe de contrôle de ce programme.
3. Donnez l'ensemble des chemins $C0$, c'est à dire qui passent par tous les noeuds.
4. Proposez une valeur x pour chaque chemin de la question précédente.
5. Est-ce que la division potentielle par 0 est détectée ?
6. Donnez l'ensemble des chemins $C1$, c'est à dire vérifiant "toutes les décisions".
7. Proposez une valeur x pour chaque chemin de la question précédente.
8. Est-ce que la division potentielle par 0 est détectée ?

2 Tests de programmes

Exercice 4 (Test dans le `main`)

Cet exercice est à réaliser avec l'environnement de développement Eclipse.

1. Créez une classe `Entier` qui admet pour seul attribut `valeur` qui est de type `int`. Cette classe devra contenir les méthodes suivantes :
   — un constructeur qui a pour entrée un `int`.
   — une méthode nommée `triple()` renvoyant un `int` dont la valeur est le triple de `valeur`.
2. Créez une classe `main` avec une méthode `main` effectuant les opérations suivantes :
   (a) créez un objet `t` de la classe `Entier` qui a pour valeur 7.
   (b) comparez `t.triple()` à 21, et affichez un message de dépendant du résultat de cette comparaison
   (c) exécutez le `main()`.
3. Modifiez la méthode `triple` de sorte que le résulat retourné ne soit pas correct.
4. Exécutez le `main()` et afin de vérifier que le test échoue.

Exercice 5 (Tests dans une classe dédiée)

La situation de l'exercice précédent ne nous permet pas de découpler les tests du développement principal de notre application. Cet exercice a pour but de corriger partiellement ce défaut. On suppose qu'on dispose toujours de la classe `Entier` de l'exercice précédent.

1. Créez un nouveau projet et insérez votre classe `Entier`.
2. Créez une classe `TestEntier` qui contient une méthode `boolean test1()`.
3. Complétez le contenu de la méthode `test1()` de la façon suivante :
   (a) initialisation d'un `Entier` nommé `t` avec comme valeur 7.
   (b) comparaison de `t.triple()` et de 21.

4. Faite un appel de cette nouvelle méthode depuis votre `main()`.
5. Cette solution est-elle préférable à la précédente ?

Exercice 6 (Prise en main du framework Junit 4)
Le framework Junit permet d'automatiser les taches de test unitaire en Java.

1. Créez un nouveau projet et insérez votre classe `Entier`.
2. Dans l'explorateur de paquets, faites un clic droit sur la classe `Entier`.
3. Dans le menu contextuel, cliquez sur `New - JUnit Test Case`. Dans ce panneau :
   — Sélectionnez le bouton radio `New JUnit 4 test`.
   — Nommez la classe `EntierTest`.
   — Cochez les cases `setUp()` et `tearDown()`.
   — Dans le champ `Class under test`, saisissez votre classe `Entier`.
   — Enfin cliquez sur `Finish`.
4. Une fois cette nouvelle classe créée, ajoutez un attribut nommé `t` de classe `EntierTest`.
5. Complétez la méthode annotée avec `@Before` en initialisant votre objet `t` avec la valeur entière de votre choix.
6. Complétez la méthode annotée avec `@Test` nommée `testTriple()` avec une instruction du type :

```
assertEquals(valeur_attendue, t.triple());
```

7. Exécutez la classe `TestEntier`. Vos tests réussissent-ils ?
8. Quel est le rôle de l'instruction `assertEquals` ?
9. En existe-il d'autre dans la même famille ?