Prog1b : Programmation Structurée

Examen final du 12 décembre 2016 (1 h 30)

Seule une feuille recto-verso d'aide mémoire manuscrite est autorisée.

Cette épreuve ne porte que sur la seconde partie du cours de Prog1, consacrée à l'étude de la programmation structurée (orientée objets ou fonctionnelle) avec le langage Scala.

★ Exercice 1 : Compréhension de code (6 points).

▷ **Question 1 :** Dessiner le schéma mémoire résultant de l'exécution du code ci-dessous.

```
class ObjectA(u:Int, v:Int) {
   var a1 = u
   var a2 = v

   def action1(u:Int, v:Int) = {
       a1 = a1 + u
       a2 = a2 + v
   }
   def action2(u:Int, v:Int) =
       new ObjectA(a1+u, a2+v)
}
```

```
var n1 = 2
var n2 = 6
val a1 = new ObjectA(n1, n2)
val a2 = a1.action2(15, 9)
a1.action1(15, 9)
n1 = n1 + n2
n2 = n1 + n2
val a3 = new ObjectA(n1, n2)
a3.action1(9, 15)

// Schéma mémoire ici
```

▷ **Question 2 :** Dessinez le schéma mémoire après exécution du code suivant (`ObjectA` est inchangée).

```
class ObjectB(b1:ObjectA, b2:ObjectA) {
   def action1(u:Int, v:Int): Unit = {
     b1.action1(u,v)
     b2.action2(u,v)
   }
   def action2(u:Int, v:Int) =
     new ObjectB(b1.action2(u,v), b2.action2(u,v))

   def action3(u:Int, v:Int) = {
     b1.action1(u,v)
     b2.action2(u,v)
     new ObjectB(b1,b2)
   }
}
```

```
var A1 = new ObjectA(1,5)
var A2 = new ObjectA(6,7)
var B1 = new ObjectB(A1,A2)
B1.action1(20,10)
val B2 = B1.action2(10,5)

A1 = new ObjectA(3,4)
A2 = new ObjectA(2,5)
val B3 = new ObjectB(new ObjectA(3,4),
                     new ObjectA(2,5))
B1.action1(5,10)
val B4 = B2.action3(50,100)

// Schéma mémoire ici
```

▷ **Question 3 :** Indiquer l'affichage produit par l'exécution du code suivant (une seule réponse correcte).

```
class Plant {
  def watering() = println("Watering a plant")
}

class Cactus extends Plant {
  override def watering() = println("Watering a cactus")
}

val plant:Plant  = new Cactus()
val cactus:Plant = new Plant()
plant.watering()
cactus.watering()
```

Réponse 1 :
```
Watering a plant
Watering a plant
```

Réponse 2 :
```
Watering a plant
Watering a cactus
```

Réponse 3 :
```
Watering a cactus
Watering a plant
```

Réponse 4 : Erreur à la compilation

▷ **Question 4 :** Indiquez l'affichage du code suivant.

```
class Book(t:String) {
  val title = t
  def equals(b:Book): Boolean = (title == b.title)
}

val name1 = "The Hitchhikers Guide to the Galaxy"
val name2 = "Mostly Harmless"

```

```
val b1 = new Book(name1)
val b2 = new Book(name1)
val b3 = new Book(name2)

println(b1.equals(b2))
println(b1 == b2)
println(b1.equals(b3))
println(b1 == b3)
```

▷ **Question 5 :** Indiquez l'affichage du code suivant, ainsi que les types statiques et les types dynamiques des différentes valeurs définies.

```
class Fleur {
  def odeur() = println("Fleur")
}
class Rose extends Flower {
  override def odeur() = println("Rose")
}
```

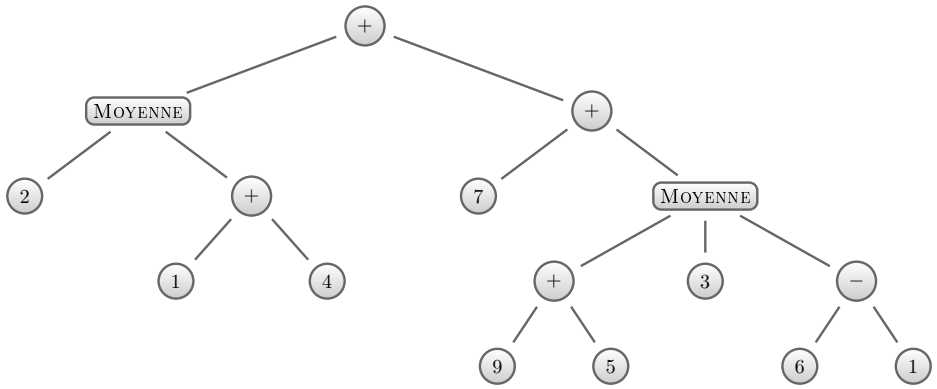
```
val f1:Flower = new Rose()
val f2:Flower = new Flower()
val f3:Rose   = new Rose()
f1.odeur()
f2.odeur()
f3.odeur()
```

★ **Exercice 2 : Écriture de code** (6 points).

On veut implémenter une solution permettant de manipuler des expressions arithmétiques. L'objectif de cet exemple-jouet est de vous permettre de démontrer votre aptitude à organiser du code de façon POO. La syntaxe exacte du code écrit est importante, mais moins que la forme générale.

Les expressions sont constituées d'opérandes constantes, d'opérateurs binaires ($+$, $-$, $\times$ et $/$) ainsi que d'opérateurs n-aires (MAXIMUM, MOYENNE, SOMME). Une expression peut être représentée sous la forme d'un arbre abstrait dont la racine est l'opérateur le moins prioritaire, les nœuds internes sont des opérateurs, et les feuilles sont les opérandes. Par exemple, l'arbre ci-dessous représente l'expression :

$$\text{MOYENNE}(2, 1+4) + (7 + \text{MOYENNE}(9+5, 3, 6-1))$$



▷ **Question 1 :** Proposer une hiérarchie de classes permettant de modéliser le problème. Identifier les classes qui seront abstraites.

▷ **Question 2 :** Dessiner le schéma mémoire de l'objet correspondant à l'expression donnée ci-dessus.

▷ **Question 3 :** Pour calculer la valeur d'une expression, que doit-on ajouter comme primitives (attributs et méthodes) et dans quelles classes ? Dessiner le graphe d'héritage avec les attributs et les méthodes (uniquement leur profil).

On souhaite maintenant étendre nos expressions en autorisant la définition et l'utilisation de variables. Une même expression pourra ainsi être réduite en fonction des valeurs de ses variables. Par exemple, on définit une variable $x = 3$ et on souhaite évaluer l'expression $max(x+2, 7, 9)$. Il faut donc être capable de lier le nom de la variable x avec la valeur 3. Puis, il faut pouvoir remplacer x par sa valeur dans l'expression. L'expression ci-dessus se réduit ainsi à la valeur 9.

▷ **Question 4 :** Définir une nouvelle classe `Variable` qui doit permettre de manipuler des expressions particulières que sont les variables ainsi que la classe `Contexte` pouvant contenir différentes définitions de variables liées.

▷ **Question 5 :** On souhaite pouvoir réduire une expression dans un contexte laissant certaines variables libres. La réduction de $min(x+y, y, x, 9)$ dans le contexte contenant $x = 3$ donne $min(3+y, y, 3)$. On ne peut pas terminer le calcul car y est encore libre, mais on a remplacé x par sa valeur et déjà calculé le minimum entre cette valeur et 9. Indiquez le profil des méthodes à définir dans chaque classe, et donnez l'intuition de leur fonctionnement (sans forcément écrire de code).

★ **Exercice 3 : Questions de synthèse** (8 points).

▷ **Question 1 :** Présentez comment les notions d'héritage, d'interfaces et de polymorphisme permettent de factoriser du code au sein d'une application en vous appuyant sur des exemples.

▷ **Question 2 :** Discutez les notions de *co-variance*, *non-variance*, *contra-variance* et *sous-typage*.

▷ **Question 3 :** Rappelez le problème classique lié à l'héritage multiple, et discutez la solution proposée par Scala par rapport à d'autres langages que vous pouvez connaître.

▷ **Question 4 :** Quelles notions POO du cours n'étaient pas reprises par les différentes fourmis à implémenter dans le projet *Ants vs. SomeBees* ? Quelles créatures pourriez-vous imaginer pour mettre ces notions à l'œuvre ?