Programmation objet : concepts avancés
Examen 2017-2018

Durée: 2 heures

Aucun document est autorisé

Le soin apporté à la rédaction et à la présentation ainsi que la rigueur seront pris en compte dans la notation.

1 Scala

```
abstract class A {
  def m1 = println("A.m1")
  def m2 : Unit
}
class B extends A {
  override def m1 = println("B.m1")
  override def m2 = println("B.m2")
}
class C extends A {
  override def m2 = { println("C.m2") }
}
trait T1 extends A {
  abstract override def m2 = {
    super.m2; println("T1.m2")
  }
}
trait T2 extends A {
  override def m2 = {
    super.m1; println("T2.m2")
  }
}
```

```
class D {
  def op : Int => Int = (n => n + 1)
}
class E {
  def op : Int => Int = (n => n - 1)
}
class M[A <: {def op : Int => Int}] {
  def g(a : A) = a.op(1) + a.op(1)
}
class N[A <: D] {
  def g(a : A) = a.op(10)
}
class H(var x : Int = 1 ) {
  def hIncX(y : H) {
    x = x + y.x; println(x)
  }
}
class F(var x : Int = 10) {
  def fIncX(y : H) {
    x = x + y.x; println(x)
  }
}
```

Figure 1 – Exercice 1

Question 1 *Pour chacunes des déclarations suivantes évaluées dans le contexte du programme dans la Figure 1, indiquez si elle est acceptée par le compilateur* Scala *et ce que produit son évaluation. Si elle est rejetée, expliquez brièvement pourquoi.*

1. `(new C with T1).m2`
2. `(new C with T2).m2`
3. `(new C with T2 with T1).m2`
4. `(new T2).m2`
5. `(new B with T2).m2`
6. `(new A with T1).m1`

□

Question 2 *Même énoncé que la question précédente.*

1. `(new M[D]).g(new D)`
2. `(new M[E]).g(new D)`
3. `(new N[D]).g(new D)`
4. `(new N[D]).g(new E)`

□

```
val f = new F
val h = new H
```

Question 3 *Même question en ajoutant les objets `f` et `h` donnés ci-dessus.*

1. `f.fIncX(f)`
2. `f.fIncX(h)`
3. `h.hIncX(f)`
4. `h.hIncX(h)`

□

```
object auto {
  implicit def fromFtoH(f:F) : H = {
    val r = new H; r.x = f.x; r
  }
  implicit def fromHtoF(h:H) : F = {
    val r = new F; r.x = h.x; r
  }
}
```

Question 4 *Même question en ajoutant l'objet `auto`.*

1. `f.fIncX(f)`
2. `f.fIncX(h)`
3. `h.hIncX(f)`
4. `h.hIncX(h)`

□

2 POO

Question 5 *Problème de la classe de base fragile.*

1. *Expliquez la* différence *entre la redéfinition et la surcharge de méthode. Donnez un exemple en Scala pour chacun de ces mécanismes.*
2. *Expliquez le mécanisme des* **vues** *en Scala. Quelle est la différence entre le sous-typage et les vues ?*

□

Question 6 *Problème de la classe de base fragile.*

1. *Expliquez brièvement* **deux** *instances du problème de la classe de base fragile, et donnez des implantations simples de ces exemples en Scala.*
2. *Expliquez brièvement le problème de l'extensibilité fonctionnelle un des mécanismes vue en cours permettant résoudre ce problème.*

□

3 Pattern Visiteur

```
trait VisitableItem {
  abstract class Item {
    def accept (v: ItemVisitor)
  }

  type ItemVisitor <: AbstractItemVisitor
  trait AbstractItemVisitor {
    self: ItemVisitor =>
    def visit (s : Item) = s.accept (this)
  }
}
```

FIGURE 2 – Pattern Visiteur

Question 7 *On considère le pattern visiteur vu en cours et donné dans la Figure 2.*

1. *Expliquez brièvement quel problème résout ce pattern.*
2. *Détaillez les éléments principaux de ce pattern, son fonctionnement et comment il aide a résoudre ce problème. Explicitez et expliquez si les types sont virtuels ou concrets, quelles contraintes doivent être satisfaites par chaque type et pourquoi.*
3. *Expliquez quelles sont les pas a suivre pour concrétiser de ce pattern. Quelles classes doivent être définies ?*

□

4 Projet

Question 8 *On travaille sur le projet de PooCAv : eVALUATOR*

1. *Donnez une vue globale des interfaces et classes fondamentales de votre projet de PooCAv. Seule la structure générale de l'architecture est demandée. Il n'est pas demandé de faire figurer chaque fonction. Il est recommandé d'utiliser des diagrammes façon UML.*
2. *Expliquez en quoi cette architecture aide a l'extensibilité de votre implémention. (Soyez concret.)*

□