Licence 2 MIEE **Première session 2013-2014**

GEN : Génie Logiciel

2 heures - Documents autorisés

Le barème est donné à titre indicatif.

Question 1 (8 points)**.** Vous indiquerez vos réponses sur votre copie sous la forme n° de question - lettre du choix correct. Par exemple, si pour la question `1` la réponse correcte est la réponse `A` vous indiquerez `1-A` sur votre copie d'examen. Certaines questions comportent plusieurs possibilités correctes, il faut les indiquer toutes pour que la réponse à la question soit considérée comme juste, par exemple : `2-CD`. Le barème pour cette question est le suivant :

- une réponse correcte à une question vaut 1 point ;
- une réponse incorrecte à une question vaut –0,5 points ;
- l'absence de réponse vaut 0 point.

1. Avec la définition `class A(x:Int){ val y=x }` comment obtenir un objet ?
   A- `val a=new A(10)` | B- `val a=A` | C- `val a=new A` | D- `val a=A(10)`

2. Avec la définition `object A{ val y=10 }` comment obtenir un objet ?
   A- `val a=new A(10)` | B- `val a=A` | C- `val a=new A` | D- `val a= new A{y=10}`

3. L'expression Scala `if (1==2) {println(19.0); 18} else {19; println(19.0)}` est de type : A-Unit | B-Double | C-AnyVal | D-Int | E- AnyRef

4. Le programme Scala suivant :

```
val m= Map(1 -> "un", 2 -> "deux")
m(3)="trois"
```

   A- Ne peut être compilé ;
   B- Lève une exception à l'exécution de la première ligne ;
   C- Lève une exception à l'exécution de la deuxième ligne ;
   D- Définit une Map à trois entrées.

5. Voici un programme Scala :

```
class A(x:Int) {
  private val c=x+1
  protected def f:Int=c
}
```

```
class B(x:Int) extends A(0){
  override def f:Int= super.f
}
val a= new B(10)
println(a.f)
```

   Le résultat affiché par ce programme est : A- `0` | B- `1` | C- `10` | D- `11`

6. On définit le trait suivant pour représenter les employés d'une entreprise.

```
trait Personne{
  val nom:String
  val prenom:String
}
```

On souhaite disposer d'un objet implantant le trait `Personne` et représentant le directeur de l'entreprise. Parmi les solutions suivantes, lesquelles sont correctes ?

A-
```
case class Directeur(nom:String,prenom:String)
val directeur= Directeur("Raoul","Macé")
```

B-
```
class Employé(n:String,p:String) extends Personne{
  val nom=n; val prenom=p
}
val directeur= new Employé("Raoul","Macé")
```

C-
```
case class Directeur(nom:String,prenom:String)
object Directeur extends Directeur("Raoul","Macé") with Personne
```

7. Le cycle de développement itératif :

   A- Met l'accent sur la satisfaction du client

   B- Impose un cahier des charges fixe

   C- Impose l'utilisation d'un gestionnaire de version

   D- Facilite l'organisation des équipes

8. Voici un trait muni d'un contrat :

```
trait NoteMidi{
    val num:Int
    require(num >= 21 && num<= 108)
    def transposer(n:Int):NoteMidi={
      require(this.num+n>=21 && this.num+n<=108)
      transposerImp(n:Int)
    }
    protected def transposerImp(n:Int):NoteMidi
}
```

Ce trait a été implémenté par A dans la classe `Note` et utilisé par B dans la fonction `transpList` :

A : implante `Note`	B : programme `transpList`

`class Note(n:Int) extends NoteMidi{`
`  val num=n`
`  override def transposerImp(n:Int)={`
`  new Note(this.num+n)`
`  }`
`}`
	`def transpList(l:List[NoteMidi],i:Int)=`
	`  if (l.forall((_).num >= 21) &&`
	`  l.forall((_).num <=108))`
	`  l.map((_).transposer(i))`
	`  else List()`

A- A a respecté le contrat

B- A n'a pas respecté le contrat

C- B a respecté le contrat

D- B n'a pas respecté le contrat

Question 2 (4 points)**.** On souhaite représenter des scènes graphiques en trois dimensions. Les scènes sont des associations de deux types d'objets : des boîtes rectangulaires (des parallépipèdes rectangles) et des sphères. Les boîtes sont définies par la position de leur coin inférieur gauche et par une hauteur, une largeur et une profondeur. Les sphères sont définies par la position de leur centre et un rayon. On définit les positions, boîtes, sphères et groupes d'objets par les classes Scala suivantes :

```
case class Position(x:Double,y:Double,z:Double)

sealed trait Scene3D
case class Boite(pos:Position,h:Double,l:Double,p:Double) extends Scene3D
case class Sphere(pos:Position,r:Double) extends Scene3D
case class Groupe(l:List[Scene3D]) extends Scene3D
```

1. Définir une fonction `translationH(s:Scene3D,d:Double):Scene3D` qui applique une translation horizontale [1] de valeur `d` de tous les objets de la scène 3D `s` et retourne la scène 3D obtenue.
2. Définir une fonction d'ordre supérieur `map(f: Scene3D => Scene3D,o:Scene3D):Scene3D` qui applique une transformation `f` à tous les objets d'une scène 3D et retourne la scène 3D obtenue.

Question 3 (8 Points)**.** On souhaite programmer un système de gestion d'employés, bureaux et postes téléphoniques dans une entreprise. Les employés sont placés dans des bureaux. Chaque bureau peut contenir au plus un poste téléphonique. A chaque poste est associé un numéro de téléphone. Le système de gestion doit permettre de réaliser les opérations suivantes :

- définir/consulter le bureau dans lequel est placé un employé ;
- définir/consulter le bureau dans lequel est placé un poste téléphonique ;
- définir/consulter le numéro de téléphone associé à un poste ;
- consulter le numéro de téléphone associé à un employé. Le numéro est celui du poste présent dans le bureau de l'employé ;
- obtenir l'ensemble des employés installés dans un bureau.

1. Proposer des traits `PosteTel`, `Employé` et `Bureau` munis des fonctions que vous jugerez nécessaires.
2. Donner le diagramme montrant les associations existantes entre les trois traits précédents. Pour simplifier, on ne fera apparaître dans les traits du diagramme que les champs (nom et type), et les noms de fonctions (sans leur type).
3. Proposer une classe implantant chaque trait.
4. Compléter le trait `Employé` proposé dans la première question, avec des contrats permettant d'assurer les propriétés suivantes :
   (a) On ne peut placer, au maximum, que 4 employés par bureau ;
   (b) Quand on place un employé `e` dans un bureau `b`, si l'opération réussit, l'ensemble des employés associés au bureau `b` contient `e` ;
   (c) Le numéro de téléphone associé à un employé est nécessairement le numéro du poste situé dans le bureau de l'employé.

1. Sur la composante `x` des positions.

Licence 2 MIEE **Seconde session 2013-2014**

GEN : Génie Logiciel

2 heures - Documents autorisés

Le barème est donné à titre indicatif.

Question 1 (8 points)**.** Vous indiquerez vos réponses sur votre copie sous la forme n° de question - lettre du choix correct. Par exemple, si pour la question 1 la réponse correcte est la réponse A vous indiquerez `1-A` sur votre copie d'examen. Certaines questions comportent plusieurs possibilités correctes, il faut les indiquer toutes pour que la réponse à la question soit considérée comme juste, par exemple : `2-CD`. Le barème pour cette question est le suivant :

- une réponse correcte à une question vaut 1 point ;
- **une réponse incorrecte à une question vaut –0,5 points** ;
- l'absence de réponse vaut 0 point.

1. SVN est un outil qui permet :
   - A- De partager des fichiers Scala entre plusieurs développeurs ;
   - B- De partager des fichiers de n'importe quel type entre plusieurs développeurs ;
   - C- De partager des répertoires entre plusieurs développeurs ;
   - D- D'obtenir les modifications faites par un autre utilisateur sur un fichier ;
   - E- D'obtenir une ancienne version d'un fichier.
2. Quel sont les types possibles pour `12` : A-Unit | B-Double | C-AnyVal | D-Int | E- AnyRef
3. Que vaut l'expression Scala suivante :

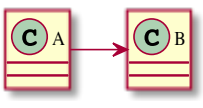
```
case class A(x:Int)
val x= A(10)
x match {
  case A(0) => 1
  case A(_) => 2
  case A(x) => x
}
```

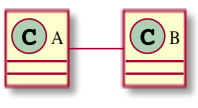
   A- `1` | B- `2` | C- `10` | D- `12`
4. Que retourne le programme suivant :

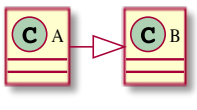
```
val t= Array("zero","un","deux")
t(1) = "one"
t= Array("trois","quatre","cinq")
println(t(1))
```

   A- `"un"` | B- `"one"` | C- `"quatre"` | D- une erreur
5. Quel est le diagramme de classe correspondant au code Scala suivant :

```
class A{
  var b: Set[B]=Set()
}
class B{ }
```

   A- 

   B- 

   C- 

6. Que retourne le programme suivant :

```
def f(x:Int):Int= {
  require(x<0)
  x+4
} ensuring (res => res >0)
println(f(-3))
```

A- une erreur car la précondition est violée | B- une erreur car la post-condition est violée | C- `1`

7. Dans le programme suivant, par quelle expression faut-il remplacer `???` pour que le programme affiche `List(3,4)` ?

```
val l2=List(1,2,3,4)
println(l2.filter(???))
```

A- `3` | B- `(_ >= 3)` | C- `(x >=3)` | D- `(x => x>=3)`

8. En Test Driven Development :
   A- on écrit des tests sans écrire le code
   B- on écrit des tests avant d'écrire le code
   C- on écrit des tests après avoir écrit le code

Question 2 (8 Points)**.** Représentation d'un système de fichiers. Pour les questions 1 à 3, on attend un unique diagramme de classes. Pour la question 4, on attend du code Scala.

1. Donnez une architecture logicielle (traits, classes, objets, association et héritage) pour un système de fichiers. Pour simplifier, on se contentera de représenter les fichiers et les répertoires. Un répertoire peut contenir un nombre arbitraire de sous-répertoires ou de fichiers. Un fichier contient seulement des données. Le système ne comporte qu'un seul utilisateur (qu'il n'est pas nécessaire de représenter).
2. Complétez votre architecture avec la notion de répertoire courant ;
3. Complétez votre architecture avec les opérations suivantes :
   - Consulter la liste des répertoires et fichiers situés dans un répertoire (`ls` d'UNIX) ;
   - Changer de répertoire courant : descendre dans un répertoire fils, remonter au répertoire père (`cd` d'UNIX) ;
   - Connaître le nom du répertoire courant (`pwd` d'UNIX).
4. Donnez le code Scala des traits, classes, objets et méthodes de votre architecture.

Question 3 (4 points)**.** On souhaite représenter des scènes graphiques en trois dimensions. Les scènes sont des associations de deux types d'objets : des boîtes rectangulaires (des parallépipèdes rectangles) et des sphères. Les boîtes sont définies par la position de leur coin inférieur gauche et par une hauteur, une largeur et une profondeur. Les sphères sont définies par la position de leur centre et un rayon. On définit les positions, boîtes, sphères et groupes d'objets par les classes Scala suivantes :

```
case class Position(x:Double,y:Double,z:Double)

sealed trait Scene3D
case class Boite(pos:Position,h:Double,l:Double,p:Double) extends Scene3D
case class Sphere(pos:Position,r:Double) extends Scene3D
case class Groupe(l:List[Scene3D]) extends Scene3D
```

1. Définir une fonction `translationH(s:Scene3D,d:Double):Scene3D` qui applique une translation horizontale [1] de valeur `d` de tous les objets de la scène 3D `s` et retourne la scène 3D obtenue.
2. Définir une fonction d'ordre supérieur `map(f: Scene3D => Scene3D,o:Scene3D):Scene3D` qui applique une transformation `f` à tous les objets d'une scène 3D et retourne la scène 3D obtenue.

1. Sur la composante `x` des positions.

Licence 2 MIEE **Première session 2014-2015**

GEN : Génie Logiciel

2 heures - Documents autorisés

Le barème est donné à titre indicatif.

Question 1 (7 points)**.** Vous indiquerez vos réponses sur votre copie sous la forme n° de question - lettre du choix correct. Par exemple, si pour la question `1` la réponse correcte est la réponse `A` vous indiquerez `1-A` sur votre copie d'examen. **Attention :** toutes les questions n'ont qu'une seule bonne réponse et certaines réponses doivent être **justifiées**. Pour chaque question, le barème est donné sous la forme d'un couple $(x, -y)$:

- une réponse correcte à cette question (avec justification si demandée) vaut x points ;
- **une réponse incorrecte (ou mal justifiée si elle est demandée) à cette question vaut $-y$ points** ;
- l'absence de réponse vaut 0 point.

1. $(1, -0.5)$ En développement dirigé par les tests (Test Driven Development) :

   a- On écrit des tests sans écrire le code

   b- On écrit le code avant d'écrire les tests

   c- On écrit des tests avant d'écrire le code

   d- On vérifie que chaque ligne du programme est testée

2. $(1, -0.5)$ Lequel de ces programmes construit exactement deux objets en mémoire ?

   A-
   ```
   class A{
     val y=10
   }
   val q= new A
   val r= new A
   ```

   B-
   ```
   object A{
     val y=10
   }
   val q= A
   val r= A
   ```

   C-
   ```
   class A(x:Int){
     val y=10
   }
   val q= new A(10)
   val r= q
   ```

3. $(1, -0.5)$ Le programme Scala suivant :

   ```
   val m= Map(1 -> "un", 2 -> "deux")
   m(3)="trois"
   ```

   A- Ne peut être compilé ;

   B- Lève une exception à l'exécution de la deuxième ligne ;

   C- Définit une table (Map) à trois entrées.

4. (1, −0.5) Voici un programme Scala :

```
class A(x:Int) {
  var c= x+1
}
```

```
val a= new A(10)
val b= a
b.c= b.c + 1
println(a.c)
```

Le résultat affiché par ce programme est : A- `12` | B- `11` | C- `10` | D- `9`

5. (2, −1) Voici un trait (avec contrat) représentant des intervalles d'entiers naturels :

```
trait Intervalle{
    var min:Int
    var max:Int
    def elargir(n:Int):Unit={
      require(n>0)
      elargirIMP(n:Int) ensuring (min<max)
    }
    protected def elargirIMP(n:Int):Unit
}
```

Ce trait a été implémenté par X dans la classe `InterIMP` et utilisé par Y dans la fonction `coller` :

X : implante `Intervalle`	Y : programme `coller`

`class InterIMP(x:Int) extends Intervalle{`
`  var min=x`
`  var max=x`
`  override def elargirIMP(n:Int)={`
`  max= max + n`
`  }`
`}`
	`def coller(i1:Intervalle,i2:Intervalle):Unit={`
	`  i1.elargir(i2.min-i1.max)`
	`}`

Qui a respecté le contrat ? **Justifiez** votre réponse en une phrase.

A- X et Y ont respecté le contrat

B- Seul Y a respecté le contrat, X ne le respecte pas

C- Seul X a respecté le contrat, Y ne le respecte pas

D- Ni X ni Y n'ont respecté le contrat

6. (1, −0.5) Que donne l'évaluation du code Scala suivant ? **Justifiez** votre réponse en une phrase.

```
trait A
class B extends A
class C extends A
List(new B, new B, new C)
```

A- L'évaluation échoue car le code est syntaxiquement incorrect

B- L'évaluation échoue car il n'est pas possible que deux classes héritent du même trait

C- L'évaluation échoue car il n'est pas possible de mettre des objets de classes différentes dans une même liste

D- L'évaluation réussit mais la liste contient uniquement des références `null`

E- L'évaluation réussit et la liste est de type `List[A]`

F- L'évaluation réussit et la liste est de type `List[Any]`

Question 2 (5 Points)**.** Soit le programme Scala suivant, définissant des couleurs et des éléments graphiques. Les éléments graphiques sont soit des boutons (d'une certaine couleur) soit des boîtes (conteneurs) contenant elles mêmes d'autres éléments graphiques.

```
sealed trait Couleur
case object Rouge extends Couleur
case object Bleu extends Couleur
case object Noir extends Couleur

sealed trait Element
case class Bouton(c:Couleur) extends Element
case class Boite(es: List[Element]) extends Element
```

1. Définissez une fonction `toutesCouleurs(e:Element):Set[Couleur]` qui rend l'ensemble des couleurs utilisées dans l'élément `e`.
2. Définissez une fonction `changerCouleur(e:Element,c:Couleur):Element` qui rend un élément graphique identique à `e` mais dans lequel tous les boutons sont de couleur `c`.

Question 3 (8 Points)**.** On souhaite programmer un outil de modélisation de bâtiments. On dispose du diagramme de classes de la figure 1. Donnez le code Scala, le plus simple possible permettant d'implanter ce diagramme. L'opération `ensOuverture` d'un mur donne l'ensemble des ouvertures (portes/fenêtres) de ce mur. L'opération `ensOuverture` d'une maison donne l'ensemble des ouvertures de tous les murs constituant la maison.

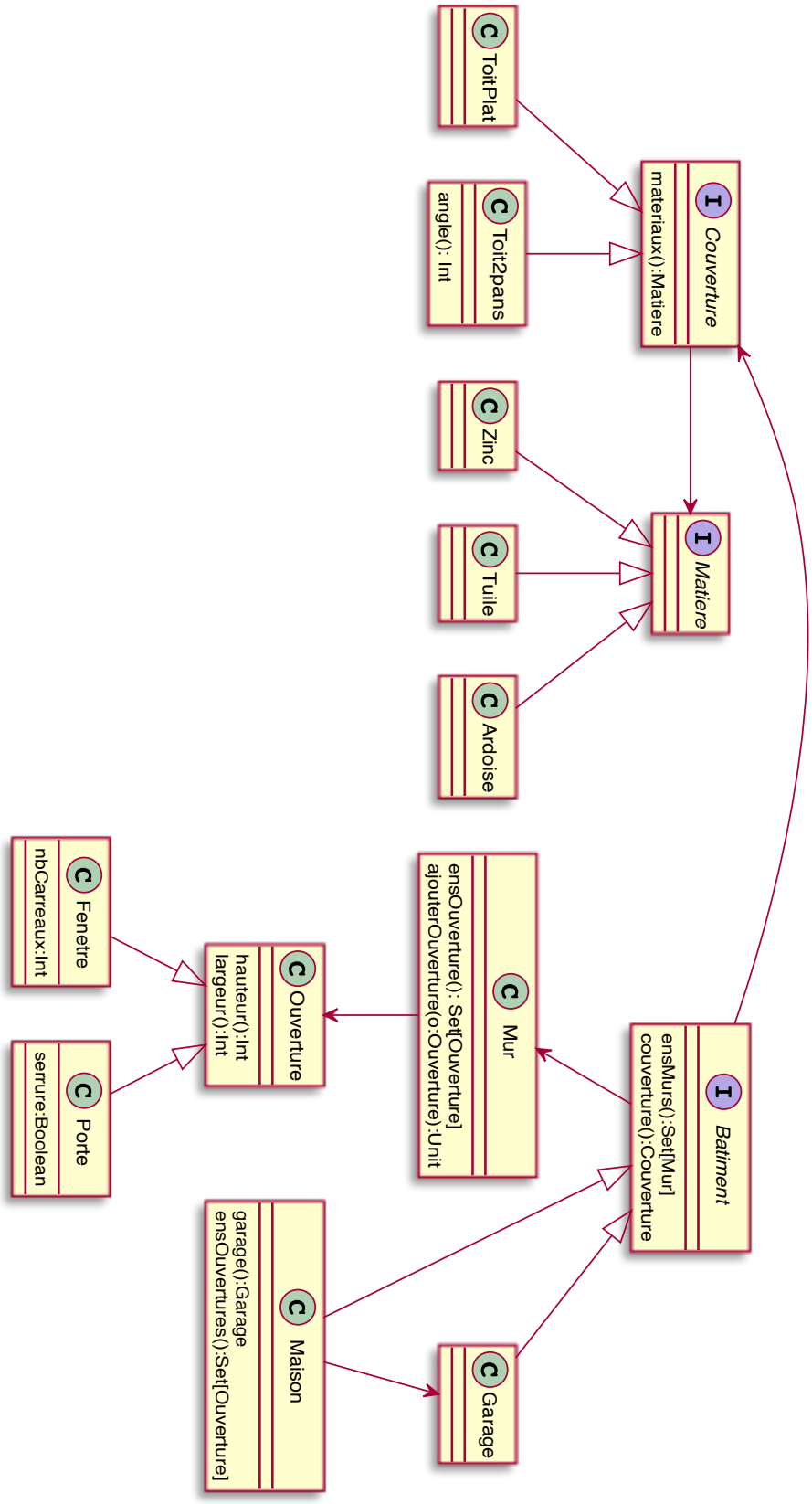


FIGURE 1 – Le diagramme de classe de l'outil de modélisation de bâtiments

Licence 2 MIEE **Seconde session 2014-2015**

GEN : Génie Logiciel

2 heures - Documents autorisés

Le barème est donné à titre indicatif.

Question 1 (6 points)**.** Vous indiquerez vos réponses sur votre copie sous la forme n° de question - lettre du choix correct. Par exemple, si pour la question `1` la réponse correcte est la réponse `A` vous indiquerez `1-A` sur votre copie d'examen. **Attention :** toutes les questions n'ont qu'une seule bonne réponse.

- une réponse correcte à une question vaut 1 point ;
- **une réponse incorrecte à une question vaut –0,5 points** ;
- l'absence de réponse vaut 0 point.

1. Le code suivant : `val l= List(1,List(2,3))`

   a- est autorisé par Scala et le type de `l` est `List`

   b- est autorisé par Scala et le type de `l` est `List[Int]`

   c- est autorisé par Scala et le type de `l` est `List[Any]`

   d- est refusé par Scala

2. Le code suivant est refusé par Scala :

```
val t= List(1,2,3)
t(2)= 4
```

   a- parce que les listes sont des objets immutables

   b- parce que le nom de variable `t` n'est pas autorisé pour référencer une liste

   c- parce que `t` est déclaré en `val` et on ne peut donc modifier `t`

   d- parce que `4` n'est pas du bon type

3. Le code suivant est refusé par Scala :

```
val l= List(1,2,3)
l.map "a"
```

   a- parce que le paramètre de `map` doit être une fonction

   b- parce `map` ne peut modifier la liste `l` qui est déclarée en `val`

   c- parce qu'on doit appeler la fonction `map` avec `()`

   d- parce que `"a"` est de type string et qu'il faut un paramètre de type `int`

4. Le code suivant est refusé par Scala :

```
val t= (1,2,3)
t match {
  case x => 1
  case (x,y,z) => 2
  case (x::r) => 3
}
```

a- car la definition `val t= (1,2,3)` est interdite

b- car le premier cas du match case ne correspond pas au type de `t`

c- car le deuxième cas du match case ne correspond pas au type de `t`

d- car le troisième cas du match case ne correspond pas au type de `t`

e- car on ne peut pas utiliser le même nom de variable (ici `x`) dans des cas différents

f- car le type du résultat n'est pas le bon

5. Le code suivant est refusé par Scala :

```
class A(x:Int) {
  var c= x
}
```

```
class B extends A(10)
  var c= x
}
val b= new B(12)
println(b.c)
```

a- car le champ `c` ne peut être défini de cette façon dans `B`

b- car le paramètre `x` ne peut être utilisé dans deux constructeurs différents

c- car la ligne `class B extends A(10)` est syntaxiquement incorrecte

d- car la création de l'objet associé à la variable `b` est impossible

6. Que donne l'évaluation du code Scala suivant ?

```
trait A
class B extends A
class C extends A
List(new B, new B, new C)
```

A- L'évaluation réussit mais la liste contient uniquement des références `null`

B- L'évaluation réussit et la liste est de type `List[A]`

C- L'évaluation réussit et la liste est de type `List[Any]`

D- L'évaluation échoue car le code est syntaxiquement incorrect

E- L'évaluation échoue car il n'est pas possible que deux classes héritent du même trait

F- L'évaluation échoue car il n'est pas possible de mettre des objets de classes différentes dans une même liste

---

Question 2 (7 Points)**.** On veut définir un contrat `TableauEntiersTrié` qui permet de modifier et d'accéder aux éléments d'un tableau d'entiers trié. Les indices seront compris entre 1 et la taille du tableau. On veut fournir les opérations suivantes :

— `taille:Int` qui donne la taille du tableau.

— `get(i:Int):Int` qui donnera l'entier à la position `i` du tableau. Dans le contrat, on s'assurera que la position `i` est bien comprise entre les bornes du tableau.

— `set(i:Int,v:Int):Unit` qui changera la valeur de l'entier associé à l'indice `i` du tableau et lui donnera la valeur `v`. Dans le contrat, on s'assurera que la position `i` est bien comprise entre les bornes du tableau et que le tableau reste trié.

Donnez le code du trait `TableauEntiersTrié` augmenté par les contrats correspondants (pre-condition et/ou post-condition pour les fonctions). On n'attend pas l'implémentation effective des opérations `taille`, `get` et `set`.

Question 3 (7 Points)**.** On souhaite définir un outil appliquant des modifications à un document texte en fonction d'un numéro de ligne. On représentera un document comme une liste de chaînes de caractères. On suppose que les lignes sont numérotées à partir de 1. Le premier élément de la liste correspond à la première ligne du document et ainsi de suite. Les opérations considérées sont l'ajout d'une ligne avant/après un numéro de ligne et la suppression d'une ligne de numéro donné. On définit les types de modification par les case classes suivantes :

```
sealed trait Modification
case class AjoutAvant(ligne:Int,valeur:String) extends Modification
case class AjoutApres(ligne:Int,valeur:String) extends Modification
case class Suppression(ligne:Int) extends Modification
```

Donnez le code Scala définissant la classe `Document` des documents textes représentés par des listes de chaînes de caractères. Complétez le code de cette classe par les fonctions :

1. `exportation:String` qui exporte la liste de chaînes de caractères sous la forme d'une seule chaîne où chaque ligne est séparée de la suivante par le caractère '`\n`' ;
2. `applique(l:List[Modification]):Unit` qui applique la liste de modifications `l` au document.

Licence 2 MIEE **Première session 2015-2016**

GEN : Génie Logiciel

2 heures - Documents autorisés

Le sujet comporte 4 pages. Le barème est donné à titre indicatif.

Question 1 (7 points)**.** Vous indiquerez vos réponses sur votre copie sous la forme n° de question - lettre du choix correct. Par exemple, si pour la question `1` la réponse correcte est la réponse `A` vous indiquerez `1-A` sur votre copie d'examen. **Attention :** toutes les questions n'ont qu'une seule bonne réponse et certaines réponses doivent être **justifiées**. Pour chaque question, le barème est donné sous la forme d'un couple $(x, -y)$:

- une réponse correcte à cette question (avec justification si elle est demandée) vaut x points ;
- **une réponse incorrecte à cette question (ou mal justifiée si une justification est demandée) vaut $-y$ points** ;
- l'absence de réponse vaut 0 point.

1. $(1, -0.5)$ En développement dirigé par les tests (Test Driven Development) :

   A- On écrit des tests sans écrire le code

   B- On écrit le code avant d'écrire les tests

   C- On écrit des tests avant d'écrire le code

   D- On vérifie que chaque ligne du programme est testée

2. $(1, -0.5)$ Lequel de ces programmes construit exactement deux objets en mémoire ?

   A-
   ```
   class A{
     val y=10
   }
   val q= new A
   var r= q
   r = new A
   ```

   B-
   ```
   class A(x:Int){
     val y=10
   }
   val q= new A(10)
   val r= q
   ```

   C-
   ```
   object A{
     val y=10
   }
   val q= A
   val r= A
   ```

3. $(1, -0.5)$ Le programme Scala suivant :

   ```
   val t= Array(1,2,3,4)
   t(3)=4
   ```

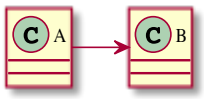
   A- Ne peut être compilé car le tableau `t` est défini en val, et il ne peut donc pas être modifié ;

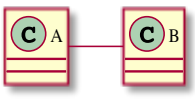
   B- Ne peut être compilé car on ne peut pas associer un tableau à un identificateur par `val` ;

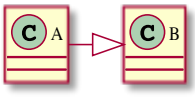
   C- Place bien la valeur `4` dans le tableau `t`.

4. (1, −0.5) Quel est le diagramme de classe correspondant au code Scala suivant :

```
class A{
  var b: Set[B]=Set()
}
class B{ }
```

A- 

B- 

C- 

5. (2, −1) Voici un trait (avec contrat) représentant les entiers naturels :

```
trait Naturel{
    var i:Int
    def soustraire(j:Int):Unit={
      require(j>0)
      soustraireIMP
    } ensuring (i>=0)
    protected def soustraireIMP:Unit
}
```

Ce trait a été implémenté par X dans la classe `NatIMP` et utilisé par Y dans la fonction `diviser` :

X : implante `Naturel`	Y : programme `diviser`

`class NatIMP(x:Int) extends Naturel{`
`  var i= if (x>=0) x else 0`
`  override def soustraireIMP(j:Int)={`
`  i= i-j`
`  }`
`}`
	`def diviser(x:Naturel,j:Int):Naturel={`
	`  if (x.i>j) diviser(x.soustraire(j),j)`
	`  else x`
	`}`

Qui a respecté le contrat ? **Justifiez** votre réponse en une phrase.

A- X et Y ont respecté le contrat

B- Seul Y a respecté le contrat, X ne le respecte pas

C- Seul X a respecté le contrat, Y ne le respecte pas

D- Ni X ni Y n'ont respecté le contrat

6. (1, −0.5) Que donne l'évaluation du code Scala suivant ? **Justifiez** votre réponse en une phrase.

```
trait A
class B extends A
class C extends A
Set(new B, new B, new C)
```

A- L'évaluation réussit et donne un ensemble de type `Set[Any]`

B- L'évaluation réussit et donne un ensemble de type `Set[A]`

C- L'évaluation réussit mais l'ensemble obtenu contient uniquement des références `null`

D- L'évaluation échoue car le code est syntaxiquement incorrect

E- L'évaluation échoue car il n'est pas possible que deux classes héritent du même trait

F- L'évaluation échoue car il n'est pas possible de mettre des objets de classes différentes dans un même ensemble

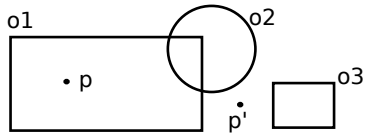
---

Question 2 (5 Points)**.** On souhaite représenter des scènes graphiques en deux dimensions à l'aide du code Scala suivant. Les scènes sont constituées de groupes de deux types d'objets : des rectangles et des cercles. Les rectangles sont définis par la position de leur coin inférieur gauche, par une hauteur et une largeur. Les cercles sont définis par la position de leur centre et leur rayon. On suppose que le repère est classique : l'axe des x orienté vers la droite et l'axe des y orienté vers le haut.

```
case class Position(x:Double,y:Double)

sealed trait Scene2D
case class Rectangle(pos:Position,h:Double,l:Double) extends Scene2D
case class Cercle(pos:Position,r:Double) extends Scene2D
case class Groupe(l:List[Scene2D]) extends Scene2D
```

Définir une fonction `estDans(p:Position,s:Scene2D):Boolean` qui détermine si un point `p` se trouve dans l'aire occupée par une `Scene2D`. Par exemple, dans la figure ci-dessous, soit $s1$ la `Scene2D` composée des objets $o1$, $o2$ et $o3$, *i.e.* $s1 = Groupe(List(o1, o2, o3))$. L'appel `estDans(`$p$`,`$s1$`)` rendra vrai et l'appel `estDans(`p'`,`$s1$`)` rendra faux.



Question 3 (8 Points)**.** On souhaite programmer un serveur de courriers électroniques (Email). Un serveur de courrier (Serveur) comporte un ensemble de boîtes aux lettres. Sur le serveur, chaque personne (Personne) dispose d'une boîte de courrier entrant (CourrierEntrant) et d'une boîte de courrier sortant (CourrierSortant). Les boîtes de courrier entrant et sortant implantent le trait BoiteMail qui offre une opération d'ajout d'un Email et de consultation de la liste de *tous* les Emails de la boîte. Le diagramme de classe du serveur de mails est donné FIGURE 1. On rappelle que dans un diagramme de classe ne figurent que les champs/méthodes publiques. Vos implantations peuvent contenir d'autres champs/méthodes `private` ou `protected` si vous jugez cela nécessaire.

1. Donnez le code Scala pour Email et Personne
2. Donnez le code Scala pour BoiteMail
3. Donnez le code Scala de CourrierSortant et CourrierEntrant. Lors d'un ajout de Email, une boîte CourrierEntrant d'une personne p enregistre un Email si p est destinataire du Email. Une boîte CourrierSortant enregistre un Email si p est expéditeur du Email. Une boîte CourrierEntrant permet également de connaitre la liste des Email non encore lus. Un mail ajouté dans cette boîte est considéré comme non lu. L'opération marquerCommeLu permet de passer un mail de l'état non lu à l'état lu.
4. Donnez le code Scala de Serveur. L'opération envoiMail déclenche l'envoi d'un Email : il est ajouté à la boîte CourrierSortant de son expéditeur et à la boîte CourrierEntrant de son destinataire.
5. Complétez le trait BoiteMail avec un contrat garantissant que l'opération `ajoute(m:Email)` ajoute bien le mail à la boîte si l'expéditeur ou le destinataire est le propriétaire de la boîte.

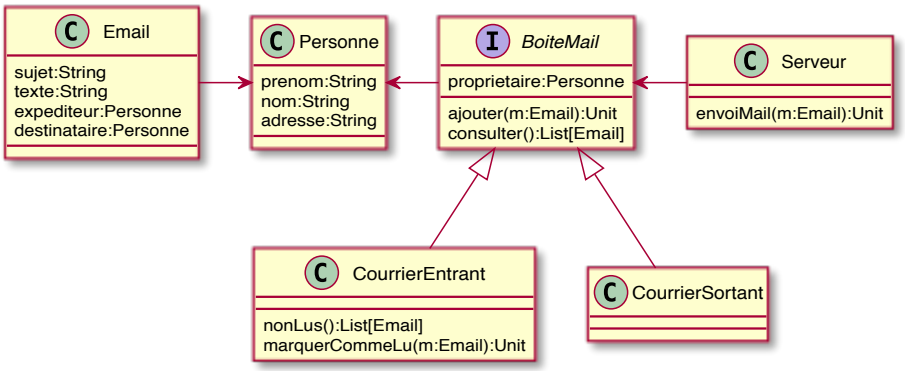


FIGURE 1 – Le diagramme de classe du serveur de mail

Licence 2 MIEE **Seconde session 2015-2016**

GEN : Génie Logiciel

2 heures - Documents autorisés

Le barème est donné à titre indicatif.

Question 1 (5 points)**.** Vous indiquerez vos réponses sur votre copie sous la forme n° de question - lettre du choix correct. Par exemple, si pour la question `1` la réponse correcte est la réponse `A` vous indiquerez `1-A` sur votre copie d'examen. **Attention :** toutes les questions n'ont qu'une seule bonne réponse.

— une réponse correcte à une question vaut 1 point ;

— **une réponse incorrecte à une question vaut –0,5 points** ;

— l'absence de réponse vaut 0 point.

1. Avec la définition `class A(x:Int){ val y=x }` comment obtenir un objet ?
   A- `val a=new A(10)` | B- `val a=A` | C- `val a=new A` | D- `val a=A(10)`

2. Avec la définition `object A{ val y=10 }` comment obtenir un objet ?
   A- `val a=new A(10)` | B- `val a=A` | C- `val a=new A` | D- `val a= new A{y=10}`

3. Le code suivant est refusé par Scala :

```
val l= List(1,2,3)
l.map "a"
```

   A- parce que le paramètre de `map` doit être une fonction

   B- parce `map` ne peut modifier la liste `l` qui est déclarée en `val`

   C- parce qu'on doit appeler la fonction `map` avec `()`

   D- parce que `"a"` est de type string et qu'il faut un paramètre de type `int`

4. Voici un programme Scala :

```
class A(x:Int) {
  var c= x+1
}
```

```
val a= new A(10)
val b= a
b.c= b.c + 1
println(a.c)
```

   Le résultat affiché par ce programme est : A- `12` | B- `11` | C- `10` | D- `9`

5. Le programme Scala suivant :

```
val m= Map(1 -> "un", 2 -> "deux")
m(3)="trois"
```

   A- ne peut être compilé car la table `m` est déclarée en `val` et ne peut être modifiée ;

   B- ne peut être compilé car la table `m` est un objet immutable et ne peut être modifiée ;

   C- lève une exception à l'exécution de la première ligne ;

   D- lève une exception à l'exécution de la deuxième ligne ;

   E- définit une Map à trois entrées.

Question 2 (7 Points)**.** On veut définir un contrat `TableauEntiersTrié` qui permet de modifier et d'accéder aux éléments d'un tableau d'entiers trié de taille n. Les indices seront compris entre 0 et $n-1$. On veut fournir les opérations suivantes :

- `taille:Int` qui donne la taille du tableau.
- `get(i:Int):Int` qui donnera l'entier à la position `i` du tableau. Dans le contrat, on s'assurera que la position `i` est bien comprise entre les bornes du tableau.
- `set(i:Int,v:Int):Unit` qui changera la valeur de l'entier associé à l'indice `i` du tableau et lui donnera la valeur `v`. Dans le contrat, on s'assurera que la position `i` est bien comprise entre les bornes du tableau et que le tableau reste trié.

Donnez le code du trait `TableauEntiersTrié` augmenté par les contrats correspondants (pre-condition et/ou post-condition pour les fonctions). On n'attend pas l'implémentation effective des opérations `taille`, `get`, `set`, ni celle de l'opération qui vérifie que le tableau est trié.

Question 3 (8 Points)**.** On souhaite programmer un système de gestion d'employés, bureaux et postes téléphoniques dans une entreprise. Les employés sont placés dans des bureaux. Chaque bureau peut contenir au plus un poste téléphonique. A chaque poste est associé un numéro de téléphone. Le système de gestion doit permettre de réaliser les opérations suivantes :

- définir/consulter le bureau dans lequel est placé un employé ;
- définir/consulter le bureau dans lequel est placé un poste téléphonique ;
- définir/consulter le numéro de téléphone associé à un poste ;
- consulter le numéro de téléphone associé à un employé. Le numéro est celui du poste présent dans le bureau de l'employé ;
- obtenir l'ensemble des employés installés dans un bureau.

1. Proposer des traits `PosteTel`, `Employé` et `Bureau` munis des fonctions que vous jugerez nécessaires.
2. Donner le diagramme montrant les associations existantes entre les trois traits précédents. Pour simplifier, on ne fera apparaître dans les traits du diagramme que les champs (nom et type), et les noms de fonctions (sans leur type).
3. Proposer une classe implantant chaque trait.
4. Compléter le trait `Employé` proposé dans la première question, avec des contrats permettant d'assurer les propriétés suivantes :
   (a) On ne peut placer, au maximum, que 4 employés par bureau ;
   (b) Quand on place un employé `e` dans un bureau `b`, si l'opération réussit, l'ensemble des employés associés au bureau `b` contient `e` ;
   (c) Le numéro de téléphone associé à un employé est nécessairement le numéro du poste situé dans le bureau de l'employé.