UNIVERSITÉ
CAEN
NORMANDIE

U.F.R. Sciences

Université de Caen Normandie
Département mathématiques-informatique
2017–2018, première session
20 décembre 2017
Bruno Zanuttini

L2 informatique et L2 mathématiques

Contrôle terminal

Unité M.MIM3A1 : Introduction à la programmation orientée objet

2 h — Tous documents autorisés

Chaque candidat doit, au début de l'épreuve, porter son nom dans le coin de la copie qu'il cachera par collage après avoir été pointé. Il devra en outre porter son numéro de place sur chacune des copies, intercalaires, ou pièces annexées.

1 Notions fondamentales (10 points)

Dans tous les exemples de code donnés, l'élision `[...]` représente des portions de code non pertinentes pour la question correspondante.

Question 1 (2 points) *Quelles sont les noms représentant des classes, et ceux représentant des instances, dans les lignes de code Java suivantes :*

```
[...]
X y = new X ();
z = y.f(t);
U v = null;
```

Question 2 (2 points) *On considère les définitions de classes Java suivantes :*

```
public class A {
   public void f () {System.out.println("Je suis A");}
   public void g () {f(); f();}
}

public class B extends A {
   @Override
   public void f () {System.out.println("Je suis B");}
}
```

Que vont afficher les instructions suivantes ? Justifier brièvement.

1. `A a = new A (); a.g();`
2. `B b = new B (); b.g();`
3. `A c = new B (); c.g();`
4. `A a = new A (); B b = new B (); a = b; a.g();`

Question 3 (2 points) *Dans le code Java suivant, le développeur a cherché à éviter la redondance de code entre ses classes* `Point` *et* `PointAvecTexte`*, en ajoutant la relation d'héritage. Quelle erreur de conception a-t-il commise ? Justifier brièvement.*

```
public class Point {
    protected int x;
    protected int y;
    [...]
    @Override
    public String toString () {
        return "(" + this.x + "," + this.y + ")";
    }
}

public class PointAvecTexte extends Point {
    protected int x;
    protected int y;
    protected String texte;
    [...]
    @Override
    public String toString () {
        return "(" + this.x + "," + this.y + ":" + this.texte + ")";
    }
}
```

Question 4 (2 points) *On suppose que l'on a une interface* `I`*, une classe abstraite* `A` *et trois classes (non abstraites)* $\texttt{C}_1$*,* $\texttt{C}_2$ *et* $\texttt{C}_3$*, telles que toutes les classes possèdent un constructeur sans argument, et*

- `A` *implémente* `I`*,*
- $\texttt{C}_1$ *et* $\texttt{C}_2$ *héritent de* `A`*,*
- $\texttt{C}_3$ *implémente* `I`*,*

à l'exclusion de toute autre relation entre ces classes. Dire, pour chacune des instructions suivantes, si elle a du sens ou non (dans ce dernier cas, justifier brièvement) :

1. `I i = new I ();`
2. `A a = new A ();`
3. $\texttt{C}_1$ $\texttt{c}_1$ `= new` $\texttt{C}_1$ `();`
4. $\texttt{C}_3$ $\texttt{c}_3$ `= new` $\texttt{C}_3$ `();`
5. `I i = new A ();`
6. `I i = new` $\texttt{C}_1$ `();`
7. `I i = new` $\texttt{C}_3$ `();`
8. `A a = new` $\texttt{C}_1$ `();`

9. `A a = new C3 ();`

10. `C1 c1 = new C2 ();`

11. `C1 c1 = new C3 ();`

Question 5 (2 points) *On considère la classe Java suivante :*

```
public class C {
    private static int attribut = 0;
    public void f ([...]) {
        C.attribut = C.attribut + 1;
        [...]
    }
    public static int g () {
        return C.attribut;
    }
}
```

Comment interpréter la valeur retournée par la méthode `g` *lorsqu'elle est appelée ?*

2 Conception (10 points)

On souhaite réaliser un *package* Java permettant de représenter des formes géométriques régulières en deux et trois dimensions, et de calculer leur surface et leur volume, respectivement. On rappelle que la surface d'un disque de rayon r est πr^2, que celle d'un rectangle de côtés c_1, c_2 est $c_1 \times c_2$, que le volume d'un cylindre de hauteur h et ayant pour base un disque de rayon r est $\pi r^2 \times h$, que celle d'un parallélépipède de base $c_1 \times c_2$ et de hauteur h est $c_1 \times c_2 \times h$, et que celle d'une boule de rayon r est $\frac{4}{3}\pi r^3$.

Pour toutes les questions qui suivent, on pourra répondre avec du code Java ou avec du pseudo-code, au choix. Pour les questions 6 et 7, seul le squelette des classes et interfaces est demandé (attributs, et méthodes avec type des arguments et de retour), pas le corps des méthodes.

Question 6 (2 points) *Sachant qu'on ne s'intéresse qu'au calcul des surfaces et volumes, proposer une interface* `Figure2D` *permettant de représenter une figure régulière quelconque en deux dimensions, ainsi qu'une interface* `Figure3D`*. Pour cela, lister les méthodes des interfaces, avec pour chacune ses arguments et son type de retour. Proposer une classe* `Disque` *et une classe* `Rectangle` *implémentant l'une et/ou l'autre de ces interfaces, en listant de même leurs attributs et méthodes.*

Question 7 (4 points) *Compléter la conception précédente avec des classes* `Pallélépipède`*,* `Cylindre` *et* `Boule`*. On veillera à éviter au maximum la duplication de code (même logique répétée plusieurs fois), en introduisant si besoin des classes et interfaces additionnelles. Pour chaque classe ou interface proposée, donner les attributs, les méthodes déclarées, définies et redéfinies en expliquant leurs arguments, leurs valeurs de retour et leur fonctionnement, et*

donner les liens entre les différentes classes et interfaces (implémentation, héritage, utilisation d'instances de l'une comme attributs d'une autre, etc.).

Question 8 (2 points) *Parmi les types proposés, donner le type à utiliser pour une liste contenant*

1. *des figures en trois dimensions quelconques,*
2. *des parallélépipèdes et des cylindres,*
3. *des cylindres et des boules,*
4. *des cylindres et des disques.*

Dans chacun des ces quatre cas, répondre en donnant le type `T` *le plus spécifique possible, parmi ceux proposés précédemment, tel que le type Java* `List<T>` *permette de contenir les objets cités.*

Question 9 (2 points) *Que prendrait en argument une méthode permettant de calculer le volume total occupé par une liste de figures quelconques en trois dimensions (parmi celles considérées) ? Quel algorithme utiliseriez-vous pour définir cette méthode ?*