



LU2IN002-2021oct  
Éléments de programmation  
par objets avec Java

Partiel du 16 novembre 2021 – Durée : 1 heure 30 minutes

Documents non autorisés. Appareils électroniques éteints et rangés.

Le barème sur 40 points est donné à titre indicatif.

**Remarque préliminaire :** la visibilité des variables et des méthodes à définir, ainsi que leurs aspects statique, ou final ne sera pas précisé. C'est à vous de décider de la meilleure solution à apporter. De même, penser à compléter les classes par l'écriture de la méthode `toString` qui leur correspond.

## Partie 1 Questions (15 points)

**Question 1.1 (2 points)** Soient les deux classes suivantes qui compilent sans erreur. Pour chacune des lignes 12, 13, 14 et 15, écrire le numéro de la ligne, puis indiquer l'affichage obtenu pour la ligne.

```

1 public class Quest {
2     public Quest() {}
3     public void methQ(int x) { System.out.println("int"); }
4     public void methQ(double y) { System.out.println("double"); }
5     public void methQ(int a, double b) { System.out.println("int,double"); }
6     public void methQ(double c, double d) { System.out.println("double,double"); }
7 }
8 public class Test {
9     public static void main(String [] args) {
10         double di=5;
11         Quest q=new Quest();
12         q.methQ(di);
13         q.methQ((int) 3.7);
14         q.methQ(8,9);
15         q.methQ((double) 1,2);
16     }
17 }
```

**Question 1.2 (3 points)** Tableau à deux dimensions avec nombre variable de colonnes.

**Q. 1.2a** Donner la déclaration et l'initialisation de la variable *tab* qui fait référence à un tableau d'entiers à 2 dimensions de 5 lignes. Le nombre de colonne de chaque ligne est choisi aléatoirement entre 1 et 10 compris.

**Q. 1.2b** Écrire une boucle sans indice qui, pour chaque ligne du tableau *tab*, affiche le nombre de colonnes qui lui correspond.

**Question 1.3** On considère un programme pour modéliser des bulles de savon. Soit la classe suivante qui compile sans erreur.

```

1 public class Bulle {
2     public static final double DIAMETRE_MAX=30;
3     private static int nbBulles=0;
4     public final int numero;
5     private double diametre ;
6
7     public Bulle(double diametre) {
8         nbBulles++;
9         numero=nbBulles;
10        this.diametre=diametre;
```

```

11     }
12     public Bulle() {
13         this(Math.random()*DIAMETRE_MAX);
14     }
15     public int getNbBulles() { return nbBulles; }
16 }

```

**Question 1.3.1 (2 points)** Pour chaque instruction des lignes 23 à 26 (4 instructions) ci-dessous, écrire le numéro de la ligne, puis indiquer si l'instruction compile (OK) ou pas (FAUX). Si l'instruction ne compile pas, justifier brièvement pourquoi (justification courte suffisante, exemples : privé, pas statique...).

```

20 public class Test1 {
21     public static void main(String [] args) {
22         Bulle b=new Bulle();
23         System.out.println(b.diametre);
24         System.out.println(b.DIAMETRE_MAX);
25         System.out.println(Bulle.nbBulles);
26         System.out.println(Bulle.getNbBulles());
27     }
28 }

```

**Question 1.3.2 (3 points)** Soit la classe suivante qui compile sans erreur.

```

50 public class Test2 {
51     public static void main(String [] args) {
52         Bulle [] tab=new Bulle[5];
53         Bulle b1=new Bulle();
54         Bulle b2=b1;
55         tab[0]=b1;
56         tab[1]=b2;
57         tab[2]=tab[0];
58         tab[3]=new Bulle();
59         // Diagramme mémoire
60         tab[3]=null;
61         // Le programme continue...
62     }
63 }

```

**Q. 1.3.2a** Dans la méthode `main`, au total, combien d'instances de la classe `Bulle` ont été créées ?

**Q. 1.3.2b** Dessiner le diagramme mémoire à la ligne 59 (sans indiquer les attributs ni les méthodes).

**Q. 1.3.2c** Quel est l'effet de l'instruction en ligne 60 sur l'état de la mémoire ?

**Question 1.4 (2 points)** Lorsqu'une bulle rencontre une autre bulle cela crée une troisième bulle qui a pour diamètre la somme des diamètres des deux bulles. Donner le code de la méthode `rencontrer` de la classe `Bulle` qui implémente ce comportement.

**Question 1.5 (3 points)** On souhaite écrire une classe `UsineABulles` pour créer des bulles de diamètre compris dans  $[1,10]$  de la façon suivante : la première bulle créée aura un diamètre de 1 cm, la deuxième de 2 cm, la troisième de 3 cm... la dixième de 10 cm, la onzième de 1 cm, la douzième de 2 cm... la vingtième de 10 cm, la vingt-et-unième de 1 cm... et ainsi de suite. Cette classe `UsineABulles` contient seulement trois membres : une variable `diametre` initialisée à 0, un constructeur, et une méthode `Bulle creerBulle()` qui permet de créer une bulle en suivant le protocole indiqué. On ne doit pas pouvoir créer d'instance d'`UsineABulles`. Écrire la classe `UsineABulles` et donner un exemple d'utilisation pour créer des bulles.

## Partie 2 Problème (27 points)

On souhaite écrire un programme pour gérer des véhicules. Un véhicule possède un nombre variable de roues (typiquement, 2 roues (vélo), 3 roues (tricycle), 4 roues (voiture), etc.).

Une roue est définie par son diamètre, en centimètres, dont la valeur est dans l'intervalle [50, 100].

**Question 2.1 (3.5 points)** Écrire la classe `Roue` qui possède un unique constructeur avec un argument donnant la valeur du diamètre de la roue. Afin de vérifier que le diamètre se trouve bien dans l'intervalle requis, la classe doit contenir deux constantes `DIAMETREMIN` et `DIAMETREMAX` (valant respectivement 50 et 100). Lors de la création d'une roue, si le diamètre donné est inférieur au diamètre minimum alors la roue est créée avec le diamètre minimum, de même si le diamètre donné est supérieur au diamètre maximum alors elle est créée avec le diamètre maximum.

**Question 2.2 (4.5 points)** Pour modéliser un véhicule avec un nombre de roues variable, on utilise pour attribut un tableau de roues qui doit être initialisé lors de la création du véhicule. Un deuxième attribut d'un véhicule est un numéro (entier) qui l'identifie de façon unique. Ce numéro est géré en utilisant un compteur de création de véhicule.

**Q. 2.2.1** Écrire la classe `Vehicule` qui possède un unique constructeur qui prend en argument un tableau de roues pour initialiser les roues du véhicules.

**Q. 2.2.2** Ajouter un accesseur pour connaître le nombre de véhicules créés depuis le lancement du programme, ainsi qu'une méthode qui rend le nombre de roues du véhicules.

**Question 2.3 (2 points)** Écrire une classe `Test` avec une méthode `main` réalisant les traitements suivants :

1. création d'un tableau de 2 roues de diamètre 60 cm ;
2. création d'un vélo (nom de variable : `velo1`) avec les 2 roues créées précédemment ;
3. création d'un autre vélo (nom de variable : `velo2`) avec 2 roues de 60 cm ;
4. affichage des 2 vélos créés et du nombre de véhicules créés depuis le lancement du programme ;

**Question 2.4 (4 points)** Dans la classe `Vehicule`, écrire la méthode `clone` pour cloner un véhicule. Ne pas oublier de donner les modifications à apporter à d'autres classes du programme si nécessaire afin que la solution soit conceptuellement correcte.

**Question 2.5 (4 points)** Deux roues sont dites compatibles si elles possèdent le même diamètre. Dans la classe `Roue`, écrire la méthode `compatible` qui prend en argument un Objet `obj` et qui rend le booléen vrai si cet objet correspond à une roue compatible avec la roue courante.

**Question 2.6 (7 points)** Un convoi est un ensemble de véhicules que l'on représente par la classe `Convoi` dont un des attributs est un tableau de véhicules. Dans notre programme, on veut garantir qu'il ne puisse exister qu'une seule instance de cette classe `Convoi` lors de l'exécution du programme.

En utilisant le mécanisme du singleton, écrire la classe `Convoi`. Dans cette classe, définir une constante `TAILLEMAX` (de valeur 4 ici) qui donne le nombre maximal de véhicules que peut contenir le convoi. Une méthode `ajoute` qui prend en argument un véhicule, permet d'ajouter un véhicule dans le convoi. Cette méthode rend le nombre de véhicules que contient le convoi à l'issue de cet ajout. Si le convoi est plein (la taille maximale de véhicules est atteinte), le nouveau véhicule n'est pas ajouté et la méthode rend la valeur `-1`. La classe doit fournir une méthode pour récupérer la référence du convoi créé et contenir un attribut qui donne le nombre de véhicules que contient le convoi.

**Question 2.7 (2.5 points)** Donner les instructions à ajouter dans la méthode `main()` de `Test` pour réaliser les traitements suivants :

1. création d'une variable de nom `c` qui contient la référence d'un convoi ;
2. ajouter les 2 vélos créés précédemment dans le convoi ;
3. afficher le convoi ;
4. tester la compatibilité d'une roue avec un convoi ;
5. tester la compatibilité d'une roue avec une autre roue de même diamètre.