Examen 2011-2012 - Durée : 1h30

POO : Programmation Orientée Objet

Première année

Nancy-Université — Université Henri Poincaré

Nom :	Prénom :

▷ **Question 1.** (3 pt)

Indiquez pour chacune des affirmations si elle est correcte ou non :

Vrai	Faux	
☐	☒	Une variable locale est accessible depuis l'extérieur d'une méthode lorsqu'elle est déclarée `public`.
☐	☒	Seule une méthode marquée `static` peut modifier une variable de classe.
☐	☒	Une classe peut avoir deux méthodes dont les profils sont identiques hormis le type de retour.
☒	☐	Comme la méthode `random` de la classe `Math` peut être invoquée par `Math.random()`, il s'agit obligatoirement une méthode de classe.
☒	☐	Une sous-classe hérite de toutes les variables d'instance et de toutes les méthodes de sa super-classe.
☐	☒	Le *polymorphisme* permet la présence d'une méthode ayant le même nom avec des profils différents dans la classe mère et la classe fille.
☒	☐	Une variable d'instance marquée `protected` est accessible dans les sous-classes.
☒	☐	En tant que développeur, vous marquez une classe `final` pour interdire la création de sous-classe.
☒	☐	Si dans un programme Java vous lisez `X implements Y` alors `Y` est obligatoirement une interface.
☐	☒	Si dans un programme Java vous lisez `X extends Y` alors `Y` est obligatoirement une classe.
☐	☒	Lorsqu'une exception est levée au plus un bloc `catch` sera exécuté.
☒	☐	Dès qu'une exception est levée, l'exécution du bloc `try` englobant est arrêté.

▷ **Question 2.** (1 pt)

Considérez la classe suivante et indiquez **la/les réponses correcte(s)** si l'instruction proposée est insérée en ligne 24. S'il n'y a pas d'erreur, précisez l'affichage.

MachinBiduleTest.java

```
1  class Machin {
2    public void coincoin() {
3      System.out.println("Machin fait coincoin");
4    }
5    public static void bipbip() {
6      System.out.println("Machin fait bipbip");
7    }
8  }
9
10 class Bidule extends Machin {
11   public void coincoin() {
12     System.out.println("Bidule fait coincoin");
13   }
14   public static void bipbip() {
15     System.out.println("Bidule fait bipbip");
16   }
17 }
18
19 class Test {
20   public static void main(String[] args) {
21     Machin m = new Machin();
22     Machin b = new Bidule();
23
24     // instruction
25   }
26 }
```

	Erreur ?	Affichage si OK
`m.coincoin();`	☐	`Machin fait coincoin`
`b.coincoin();`	☐	`Bidule fait coicoin`
`((Bidule) b).coincoin();`	☐	`Bidule fait coincoin`
`m.bipbip();`	☐	`Machin fait bipbip`
`b.bipbip();`	☐	`Machin fait bipbip`
`((Bidule) b).bipbip();`	☐	`Bidule fait bipbip`

▷ **Question 3.** (1 pt)
Considérez les classes suivantes et indiquez **la réponse correcte** :

TestCounter.java
```
1  class Counter {
2    private int counter = 0;
3
4    public static void increment() {
5      counter++;
6    }
7    public static int getValue() {
8      return counter;
9    }
10 }
11
12 class TestCounter {
13   public static void main(String args[]) {
14     Counter c1 = new Counter();
15     c1.increment();
16     Counter c2 = new Counter();
17     c2.increment();
18
19     System.out.println(c2.getValue());
20   }
21 }
```

☐ Ce code ne compile pas (erreur ligne 2).
☒ Ce code ne compile pas (erreur ligne 5 et 8).
☐ Ce code ne compile pas (erreur lignes 15 et 17).
☐ Ce code compile et le programme affiche : `0`
☐ Ce code compile et le programme affiche : `1`
☐ Ce code compile et le programme affiche : `2`

▷ **Question 4.** (1 pt)
Indiquez les concepts mis en œuvre dans le sujet de TP portant sur les dipôles (Résistance, self, montage en série, montage en parallèle, etc.)

Vrai	Faux	
☒	☐	l'encapsulation.
☒	☐	l'héritage.
☒	☐	le polymorphisme.
☐	☒	la surchage de méthode.
☒	☐	la rédéfinition de méthode.
☒	☐	la délégation.

▷ **Question 5.** (1 pt)
Considérez la classe suivante et indiquez **la/les réponses correcte(s)** si l'instruction proposée est insérée en ligne 30. S'il n'y a pas d'erreur, précisez l'affichage.

LooksLike.java
```
1  class Pokemon {
2    protected long power;
3
4    public Pokemon(long p) {
5      this.power = p;
6    }
7    public boolean looksLike(Pokemon a) {
8      return (a.power == this.power);
9    }
10 }
11
12 class ElectricalPokemon extends Pokemon {
13   private String name;
14   public ElectricalPokemon(long p, String name) {
15     super(p);
16     this.name = name;
17   }
18   public boolean looksLike(ElectricalPokemon d) {
19     return (d.power == this.power
20           && this.name.equals(d.name));
21   }
22 }
23
24 class LooksLike {
25   public static void main(String args[]) {
26     boolean flag = false;
27     ElectricalPokemon pikachu = new ElectricalPokemon(1L, "Pikachu");
28     ElectricalPokemon cloneA = new ElectricalPokemon(1L, "Raichu");
29     Pokemon cloneB = new ElectricalPokemon(1L, "Fulguris");
30     // instruction a inserer
31     System.out.println(flag);
32   }
33 }
```

	Erreur	Ok	Affichage
`flag = pikachu.looksLike(cloneA);`	☐	☒	false
`flag = cloneA.looksLike(pikachu);`	☐	☒	false
`flag = pikachu.looksLike(cloneB);`	☐	☒	true
`flag = cloneB.looksLike(pikachu);`	☐	☒	true

▷ **Question 6.** (1 pt)
Considérez l'extrait de code Java suivant :

```
1  int lowerLimit;
2  ...
3  try {
4    System.out.println("Entering the try block.");
5    if (lowerLimit < 100)
6      throw new Exception("Lower limit violation.");
7
8    System.out.println("Exiting the try block.");
9  } catch (Exception e) {
10         System.out.println("Exception: " + e.getMessage());
11 }
12
13 System.out.println("After the catch block.");
```

Quel est l'affichage si la valeur de la variable `lowerLimit` est 50 ?

Entering the try block.
Exception : Lower limit violation.
After the catch block.

Quel est l'affichage si la valeur de la variable `lowerLimit` est 150 ?

Entering the try block.
Exiting the try block.
After the catch block.

▷ **Question 7.** (1,5 pt)
Considérez l'extrait de code Java suivant :

Constructors.java

```
1  class Animal {
2    Animal() {
3      System.out.println("cons de Animal");
4    }
5  }
6
7  class Bovide extends Animal {
8    Bovide() {
9      System.out.println("cons de Bovide");
10   }
11
12   Bovide(int x) {
13     this();
14     System.out.println("autre cons de Bovide");
15   }
16 }
17
18 class Vache extends Bovide {
19   Vache() {
20     super(3);
21     System.out.println("cons de Vache");
22   }
23
24   public static void main(String[] arg) {
25     new Vache();
26   }
27 }
```

Déterminez l'affichage de la méthode principale `void main(String args[])`. :

cons de Animal
cons de Bovide
autre cons de Bovide
cons de Vache

Même question en supposant que l'on supprime l'instruction `this()` dans la classe `Bovide` :

cons de Animal
autre cons de Bovide
cons de Vache

Même question en supposant que l'on supprime également l'instruction `super(3)` de la classe `Vache` :

```
cons de Animal
cons de Bovide
cons de Vache
```

▷ **Question 8.** (1,5 pt)
Expliquez les notions de *couplage* et de *cohésion*. Vous pourrez illustrer votre réponse par un schéma.

todo.

▷ **Question 9.** (1 pt)
Donnez un exemple illustrant le concept de *liaison dynamique*.

todo.

▷ **Question 10.** Schéma mémoire (3 pt)

Considérez les classes suivantes et réalisez des schémas de la mémoire (états de la pile et du tas) lors de l'exécution de la méthode `main()` de la classe principale `MainLibrary` aux points identifiés dans le code source (indiqués par le marqueur `// POINT_? memory schema at this point`).

Vous préciserez également l'affichage obtenu sur la sortie standard à chaque étape.

MainLibrary.java

```
1  class MainLibrary {
2    public static void main(String[] args) {
3      Library l = new Library();
4
5      Book b = new Book("H2G2 vol.1");
6      Author a = new Author("D. Adams");
7      b.addAuthor(a);
8      l.register(b);
9
10     Book b2 = new Book("H2G2 vol.2");
11     Author a2 = a.duplicate();
12     b2.addAuthor(a2);
13     a2.setName("Doug Adams");
14     l.register(b2);
15
16     // POINT_1 memory schema at this point
17     System.out.println(l.toString());
18
19     Book b3 = new Book("H2G2 vol.3");
20     Author a3 = a2;
21     b3.addAuthor(a3);
22     a3.setName("Douglas Adams");
23     l.register(b3);
24
25     // POINT_2 memory schema at this point
26     System.out.println(l.toString());
27   }
28 }
```

Library.java

```
1  import java.util.List;
2  import java.util.ArrayList;
3
4  class Library {
5    private List<Book> books;
6
7    public Library() {
8      this.books = new ArrayList<Book>();
9    }
10
11   public void register(Book b) {
12     this.books.add(b);
13   }
14
15   public String toString() {
16     String res = "";
17     for (Book b : this.books)
18       res += b.toString()+"\n";
19     return res;
20   }
21 }
```

Book.java

```
1  import java.util.List;
2  import java.util.ArrayList;
3
4  class Book {
5    private List<Author> authors;
6    private String title;
7
8    public Book(String title) {
9      this.title = title;
10     this.authors = new ArrayList<Author>();
11   }
12
13   public void addAuthor(Author a) {
14     this.authors.add(a);
15   }
16
17   public String toString() {
18     String res = this.title + ", ";
19     for (Author a : this.authors)
20       res += a.toString()+" ";
21     return res;
22   }
23 }
```

Author.java

```
1  class Author {
2    private String name;
3
4    public Author(String n) {
5      this.name = n;
6    }
7
8    public void setName(String n) {
9      this.name = n;
10   }
11
12   public String toString() {
13     return this.name;
14   }
15
16   public Author duplicate() {
17     return new Author(this.name);
18   }
19 }
```

Réponse

```
// POINT 1
H2G2 vol.1, D. Adams
H2G2 vol.2, Doug Adams

// POINT 2
H2G2 vol.1, D. Adams
H2G2 vol.2, Douglas Adams
H2G2 vol.3, Douglas Adams
```

Fin réponse