Corrigé du contrôle terminal 1^{ere} session, Décembre 2014

Ce corrigé comporte 4 pages.

Documents non autorisés. Les téléphones portables, smartphones et ordinateurs doivent être éteints et rangés. Le barème est donné à titre indicatif.

1. Concepts (6 pts)

Exercice 1.1. Difficulté : ★☆☆☆☆, Durée : ★☆☆☆☆

Question 1. Donnez la signification de l'utilisation du mot-clé `final` dans le contexte de chacune des déclarations suivantes :

- déclaration d'une classe : toute dérivation (héritage) de la classe en question est interdite.
- déclaration d'un attribut de classe : c'est une constante de classe ; sa valeur ne peut modifiée après son initialisation à la déclaration.
- déclaration d'un attribut d'instance : c'est une constante d'instance ; sa valeur ne peut être modifiée après son initialisation au plus tard dans le constructeur.
- déclaration d'une méthode : toute redéfinition future de cette méthode (même signature) est interdite.
- déclaration d'un paramètre formel de méthode : toute modification de sa valeur dans le corps de la méthode est interdite.

Question 2. On dispose d'objets d'une classe `Client`, à insérer dans une collection selon leur priorité. Chaque client est principalement caractérisé par son identifiant (chaîne de caractères) unique, et son numéro de priorité (entier). Les objets clients doivent être triés à leur insertion ou suppression de la collection. Quelle modification faut-il apporter à la définition de `Client` et quelle implantation de collection faudrait-il utiliser pour satisfaire cette contrainte ?

Il faut rendre la classe `Client` comparable, en la faisant implanter l'interface `Comparable<Client>`. La comparaison se fera entre leurs priorités (deux à deux). Il faudrait ensuite utiliser une implantation type `TreeSet` (ou `TreeMap`, également accepté) de collection. Un exemple d'implantation de la classe `Client` est donné ci-après (non attendu dans la réponse).

```
1  package fileordonneeclients;
2
3  /**
4   * Client disposant d'une priorité permettant de
5   * comparer les instances entre elles.
6   * @author lom
7   */
8  public class Client implements Comparable<Client> {
9      private final String nom;
10     private final int priorite;
11
12
13     public Client(String name, int priority){
14         nom = name;
15         priorite = priority;
16     }
17
```

```
18      public String getName(){
19          return nom;
20      }
21
22      public int getPriority(){
23          return priorite;
24      }
25
26      public int compareTo(Client o) {
27          if (priorite < o.getPriority()) return -1;
28          if (priorite > o.getPriority()) return 1;
29          return 0;
30      }
31
32      @Override
33      public String toString(){
34          StringBuffer clientData = new StringBuffer("(");
35          clientData.append(nom + ", " + priorite + ")" );
36          return clientData.toString();
37      }
38  }
```

Exercice 1.2. Difficulté : ★★☆☆☆, Durée : ★★☆☆☆

Question 3. Que réalise l'instruction suivante : YYY y = new YYY();

1. **Déclare une référence de type YYY nommée y.**
2. **Instancie un nouvel objet de type YYY.**
3. **Invoque le constructeur par défaut de la classe YYY.**
4. **Affecte à la référence y l'adresse de l'objet nouvellement créé.**

Question 4. Que génère le code suivant ?

Listing 1 – Classe Exception

```
1   public class MainException {
2       public static void main(String args[]) {
3           // du code ...
4           try {
5               // encore du code ...
6               throw new MonExceptionZ();
7           } catch (MonExceptionX b) {
8               System.out.println("Levé MonExceptionX");
9           } catch (MonExceptionZ d) {
10              System.out.println("Levé MonExceptionZ");
11          }
12      }
13
14      static class MonExceptionX extends Exception {}
15      static class MonExceptionZ extends MonExceptionX  {}
16  }
```

1. Affiche à l'exécution : `Levé MonExceptionX`
2. Affiche à l'exécution : `Levé MonExceptionZ`
3. Erreur de compilation : `MonExceptionZ is not throwable`
4. **Erreur à la compilation : `MonExceptionZ class has already been caught`**

Question 5. Sélectionnez la bonne manière d'affecter un gestionnaire de disposition `GridLayout` à un composant Swing `cmp`.

1. cmp.setLayoutManager(new GridLayout(2, 3));
2. cmp.setGridLayout(2, 3);
3. **cmp.setLayout(new GridLayout(2, 3));**
4. cmp.getLayoutManagers().addLayoutManager(new GridLayout(2, 3));
5. cmp.getLayouts().addLayout(new GridLayout(2, 3));

Question 6. Reportez toutes les erreurs de syntaxe du bout de code suivant (lire le code en question dans l'énoncé).

Il y a 7 erreurs dans le code affiché :

1. Nom de la classe manquant.
2. Type de l'attribut `area` manquant.
3. Le deuxième attribut, dont le nom manque, est déclaré avec 2 types primitifs simultanément...
4. Erreur dans l'orthographe du modificateur d'accès `public` de la méthode f2.
5. L'opérateur de comparaison dans l'expression conditionnelle du `if` doit être == et non = (qui est l'opérateur d'affectation).
6. Le bloc de commentaire avant la méthode `setArea` n'est pas fermé.
7. La déclaration de la classe n'est pas fermée.

2. Analyse de code (4 pts)

Exercice 2.1. Difficulté : ★☆☆☆☆, Durée : ★☆☆☆☆

Question 7. Qu'affiche le programme ci-après ?

Listing 2 – Classes Super et Sub

```
1  public class Super {
2     void print () { System.out.println (this.getVal()); }
3     int getVal () { return 20; }
4  }
5
6  public class Sub extends Super {
7     int getVal () { return 10; } }
8
9     public class EE {
10       public static void main (String[] args) {
11          new Super().print();
12          new Sub().print(); }
13       }
14 }
```

1. 20
   10

Exercice 2.2. Difficulté : ★★½☆☆, Durée : ★★☆☆☆

Soit le programme (lire le programme dans l'énoncé).

Question 8. Qu'affichent les instructions suivantes ?

1. `Sonde s1 = new Sonde(); s1.doFirst(8, 4);`
   32 0 2 8 false
2. `Sonde s2 = new Sonde(); s2.doSecond(5.0, 8.0, 9.0);`
   Case 1
3. `Sonde s3 = new Sonde(); s3.doThird(3);`
   1 3 4

3. Système simplifié de réservation vols (8 pts)

Code source dans l'espace cours en ligne de CPO ligne (projet IntelliJ).

— Questions 9 e 10 : 1 point chacun
— Question 11 : 2 points
— Question 12 : 6 points