UEF4.3. Programmation Orientée Objet

EXAMEN FINAL

02 heures-Documents interdits

Exercice 1 (6 pts) : Voici le code de 3 classes (dans 3 fichiers différents).

```
public class A {
 private int x = 2;
   public void m(int x) {
    this.x=this.x*x;
    System.out.print("A avec x="+this.x);
   }
}
public class B extends A {
private int x = 3;
public void m(int x) {
   super.m(x);
   this.x=this.x+x;
  System.out.println("B avec x="+x);
}}
```

```
public class Test {
public static void main(String[]args) {
 A a = new A();
 B b = (B) a;
 b.m(2);
}
}
```

Question 1 : Que provoque l'exécution du code de la classe Test ? Choisissez l'une de ces réponses ***en justifiant*** votre choix.

(1) Ce code ne compile pas.
(2) Il affiche "A avec x=4".
(3) Il affiche "A avec x=4 B avec x=2".
(4) Il y a une erreur à l'exécution.

Question 2 : Nous modifions l'entête de la classe A comme suit :

```
class A implements Serializable
```

Qu'affiche l'exécution de la classe Test avec le nouveau code suivant :

```
import java.io.*;
public class Test {
public static void main(String[] args) {
ObjectInputStream in=null;
ObjectOutputStream out=null;
try {
out = new ObjectOutputStream(
        new BufferedOutputStream
        new FileOutputStream(
        new File("test.dat"))));
 A a= new A();
 a.m(3);
 out.writeObject(a);
out.writeObject(new B());
out.close();
```

```
in = new ObjectInputStream(
        new BufferedInputStream(
        new FileInputStream(
        new File("test.dat"))));

((A)in.readObject()).m(2);
((B)in.readObject()).m(2);
in.close();
} catch (ClassNotFoundException e) {
        e.printStackTrace();
} catch (FileNotFoundException e) {
        e.printStackTrace();
} catch (IOException e) {
        e.printStackTrace();
}
}}
```

Question 3: Qu'affiche l'exécution de la classe Test en modifiant les codes des classes B et Test comme suit :

```
class B extends A {
 private int x = 3;
 public void m(int z) {
  final int y=2;
  class In{
   private int x =1;
   public void mI(int k) {
         this.x+=k*y;
         B.this.x+=this.x;
   }
  }
  In i= new In();
  i.mI(z);
  System.out.println("B avec x="+x);
 }
}
```

```
public class Test {
 public static void main(String[] args) {
  A a = new B();
  a.m(2);
 }
}
```

Exercice 2 (14 pts)

On veut réaliser un programme permettant la gestion de recettes et de menus dans le cadre d'un régime alimentaire. Une recette a un nom, une durée de préparation en minutes, un degré de difficulté (très facile, facile, difficile) et une liste d'ingrédients dont chacun est défini par un aliment et une quantité. Chaque aliment est caractérisé par son nom, une unité de mesure (gramme ou litre) et le nombre de calories pour 100 unités de mesure. Certaines recettes se composent, en plus de leurs ingrédients, d'une ou plusieurs recettes de base à préparer au préalable (exemple: pour réaliser une tarte aux fraises, il faut d'abord préparer une pâte sablée et une crème pâtissière). Grâce à ce programme, l'utilisateur peut ajouter une nouvelle recette ou supprimer une recette existante. Il peut aussi composer et sauvegarder des menus journaliers. Pour ajouter un menu, il doit indiquer le nombre de calories à ne pas dépasser et sélectionner un certain nombre de recettes. Le programme peut alors valider le menu ou signaler à l'utilisateur que le nombre maximal des calories indiqué est dépassé.

1. Proposez une conception orientée objet pour ce programme en traçant un diagramme des classes illustrant les relations entre elles (utilisation, héritage, implémentation d'interface), et en indiquant dans un tableau les attributs et les méthodes nécessaires (hormis les constructeurs, les getters et les setters).
2. Pour chacun des cas ci-dessous:
   - Donnez l'instruction java permettant de déclarer et instancier la collection la plus adéquate pour contenir les recettes.
   - Expliquez les conditions nécessaires pour le bon fonctionnement du programme et donnez le code java des méthodes nécessaires.

Cas 2.1. Il n'existe pas deux recettes ayant le même nom.

Cas 2.2. Il n'existe pas deux recettes ayant le même nom et les recettes sont triées de la moins calorique à la plus calorique.

Cas 2.3. *Les* recettes sont classées par degré de difficulté, et pour chaque degré de difficulté, *elles sont* triées suivant l'ordre décroissant de la durée de préparation.

3. Donnez le code source de la méthode *ajouterMenu* qui reçoit un menu et un nombre de calories en paramètres, vérifie si le menu est valide avant de l'ajouter, et alerte l'utilisateur s'il ne l'est pas.

4. Pour améliorer notre programme, on désire guider l'utilisateur dans la composition de ses menus : dans le cas où un menu dépasse le nombre de calories fixé, le programme propose pour chaque recette sélectionnée, des substituts moins caloriques pour chaque ingrédient ou recette de base (exemple: dans la tarte aux fraises, remplacer la crème pâtissière par du yaourt).
   - Expliquez (**sans** donner le code source) ce que vous devez modifier dans votre conception afin de satisfaire ce besoin.
   - Donnez le nouveau diagramme des classes.