

# Licence d'informatique 2000-2001

## Langage Java

Richard Grin – Françoise Baude

### Examen final - juin 2001

#### Durée de l'épreuve : 2 heures

L'usage des photocopiés du cours est autorisé. Tous les autres documents, sous quelque forme que ce soit (papier, électronique ou autre) sont interdits, y compris les énoncés et corrigés des exercices des TD.

Remarque importante : si vous trouvez des fautes de frappes dans les codes des classes, n'en tenez pas compte. Aucun des exercices ne repose sur ce genre d'erreur. Dans le doute, demandez à Richard Grin.

#### Exercice 1 (3 points)

Voici quelques portions de code. Pour chaque portion, écrivez

- “OK” si vous pensez que ce code ne provoquera pas d'erreur à la compilation ou à l'exécution; suivi d'une explication montrant que vous avez compris pourquoi.
- “Faux” si vous pensez que ce code provoquera une erreur. Dites alors si l'erreur aura lieu à la compilation ou à l'exécution. Proposez une brève correction (sous forme d'un bout de code si besoin) que vous commenterez.

```
Code 1 HashMap h = new HashMap();
      Employe e = new Employe("Toto");
      h.put(1235, e);
```

```
Code 2 Object o = new ArrayList();
```

```
Code 3 List l = new ArrayList();
```

```
Code 4 List l = new List();
```

```
Code 5 Object ol = new ArrayList();
      ol.add(new Employe("Toto"));
```

#### Exercice 2 (3 points)

Répondez en donnant le numéro de la bonne réponse, puis donnez une *très brève* explication de votre réponse. Distinguez bien les questions.

Voici le code de 3 classes (dans 3 fichiers différents).

```
public class A {
    public void m() {
        System.out.println("A");
    }
}
```

```
public class B extends A {
    public void m() {
```

```

        System.out.println("B");
    }
}

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

```

### Question 1

On veut lancer l'exécution de la classe Test.

- (1) Ce code ne compile pas.
- (2) Il affiche A.
- (3) Il affiche B.
- (4) Il y a une erreur à l'exécution qui lève une exception `java.lang.ClassCastException`.

### Question 2

Idem question 1, mais en remplaçant la méthode main par celle-ci :

```

A a = new B();
a.m();

```

### Question 3

Idem question 1, mais en remplaçant la méthode main par celle-ci :

```

B b = new B();
A a = (A)b;
a.m();

```

## Exercice 3 (5 points)

- (1) Voici le code d'une classe.

```

import java.util.*;
public class Test {
    public Test() {
        String[] str = {"a", "b", "c"};
        int lStr = str.length;
        HashSet set = new HashSet();
        int nbElem = 0;
        for (int k=0; k < lStr; k++) {
            set.add(str[k]);
        }
        Iterator it = set.iterator();
        while (it.hasNext()) {

```

```

        nbElem++;
    }
    System.out.println(nbElem);
}

public static void main(String[] args) {
    new Test();
}
}

```

On lance l'exécution de la classe Test. Expliquer pourquoi le programme boucle à l'infini sans rien afficher. Modifiez légèrement le code pour qu'il n'y ait plus ce problème de boucle qui ne se termine pas, et dites ce qui sera affiché.

- (2) Que peut contenir un fichier de suffixe .jar. Peut-on s'en servir "tel quel"? Donner un exemple d'utilisation.
- (3) Quel est le principe d'utilisation/de fonctionnement de l'outil javadoc?
- (4) Expliquer dans le code ci-dessous si l'instruction `nbLivres++` sera ou non toujours exécutée. Profitez-en pour comparer les 2 concepts de *Compile time Exception* et de *Runtime Exception*.

```

public void ajouter(Livre l) throws EtagerePleineException {
    try {
        livres[nbLivres] = l;
    }
    catch (ArrayIndexOutOfBoundsException e) {
        throw new EtagerePleineException(this);
    }
    nbLivres++;
}

```

## Exercice 4 (9 points)

Il s'agit de programmer un système qui permettra à l'enseignant en charge de préparer le jury de la licence, d'obtenir la moyenne générale de chaque étudiant, et ce à partir de toutes les notes, d'examen, de projets, de TP long, etc., exprimées sur 20.

Certaines notes sont saisies directement (on ne vous demande pas d'écrire l'interface graphique qui permettra de le faire dans la pratique). D'autres notes sont calculées à partir des notes saisies.

Attention, certaines notes peuvent être calculées à partir d'autres notes déjà calculées. On pourrait ainsi utiliser la moyenne du cours de concurrence pour calculer la moyenne du module L3. Ce cas n'est pas illustré par la méthode main pour ne pas vous donner un code trop long à lire.

S'il manque une note à un étudiant pour faire le calcul d'une moyenne, on considère qu'il a eu une absence justifiée et on ignore cette note dans le calcul de la moyenne (voir méthode main pour Bibi).

Voici ci-dessous un exemple consistant à calculer les notes du cours de concurrence pour deux étudiants Toto et Bibi.

```
import java.util.*;
```

```

public class TestNotes {

    public static void main(String[] args){
        // Description des notes ; si une note est calculée à partir d'autres
        // notes (note composée), on indique la façon de la calculer
        DescriptionNote exCc = new DescriptionNoteSimple("Examen concurrence");
        DescriptionNote projCc = new DescriptionNoteSimple("Projet concurrence");
        DescriptionNoteComposee cc = new
        DescriptionNoteComposee("Concurrence");
        // La moyenne de concurrence est calculée à partir de la note
        // du projet (coeff. 4) et de la note d'examen (coeff. 6).
        cc.addNote(exCc, 6);
        cc.addNote(projCc, 4);

        // Attribution des notes aux étudiants
        Etudiant e1 = new Etudiant("Toto");
        e1.addNote(exCc, 10);
        e1.addNote(projCc, 12);

        // On n'entre pas toutes les notes pour Bibi qui a été absent à
        // l'examen de concurrence
        Etudiant e2 = new Etudiant("Bibi");
        e2.addNote(projCc, 13);

        // Affichage de quelques notes et moyennes
        // D'abord une note simple
        // Affichera Moyenne de Toto en Examen concurrence : 10.0
        System.out.println("Moyenne de " + e1.getNom()
            + " en " + exCc.getLibelle()
            + " : "+ e1.getMoyenne(exCc));

        // Et ensuite des moyennes calculées
        // Affichera Moyenne de Toto en Concurrence : 10.8
        System.out.println("Moyenne de " + e1.getNom()
            + " en " + cc.getLibelle()
            + " : "+ e1.getMoyenne(cc));

        // Affichera Moyenne de Bibi en Concurrence : 13.0
        System.out.println("Moyenne de " + e2.getNom()
            + " en " + cc.getLibelle()
            + " : " + e2.getMoyenne(cc));
    }
}

```

On vous fournit la classe Etudiant :

```
import java.util.*;
```

```
/**
 * Un étudiant avec un nom et des notes.
```

```

*/
public class Etudiant {
    private String nom;
    // Contient les notes de l'étudiant
    private Map notes = new HashMap();

    public Etudiant(String nom) {
        this.nom = nom;
    }

    public String getNom() {
        return nom;
    }

    /**
     * Attribue une note à l'étudiant.
     * @param descriptionNote indique de quelle note il s'agit.
     * @param note est la valeur de la note.
     */
    public void addNote(DescriptionNote descriptionNote, double note) {
        notes.put(descriptionNote, new Double(note));
    }

    /**
     * Retourne une moyenne ou une note.
     * @return la note ou la moyenne.
     * retourne -1 si l'étudiant n'a pas cette note ou si aucune
     * note n'est disponible pour calculer la moyenne.
     */
    public double getMoyenne(DescriptionNote descriptionNote) {
        return descriptionNote.getValeur(this);
    }

    /**
     * Retourne une note enregistrée avec addNote.
     * @return la note.
     * retourne -1 si l'étudiant n'a pas cette note dans la HashMap notes.
     */
    public double getNote(DescriptionNote descriptionNote) {
        Double note = (Double)notes.get(descriptionNote);
        if (note != null) {
            return note.doubleValue();
        }
        else {
            return -1;
        }
    }
} // Etudiant

```

### **Question 1**

Écrire toutes les classes nécessaires au fonctionnement de la méthode main en utilisant le polymorphisme le plus possible.

Toutes vos classes devront utiliser les classes fournies. Vous devrez donc respecter les noms des méthodes données dans les classes fournies.

Vous vous attacherez plus particulièrement à écrire les méthodes `getValeur` (voir classe `Etudiant`) qui permettent de fournir la valeur d'une note (calculée ou non) à partir de la définition de la note et des notes attribuées à l'étudiant.

### **Question 2**

Rajouter ce qu'il faut afin de pouvoir sauvegarder dans un fichier par sérialisation la description d'une note composée (pas les notes elles-mêmes mais seulement la description).

Ajouter du code dans la méthode main pour sauvegarder la description de "Concurrence".