IFT1020 - Session Été, Final

Mohamed Lokbani

IFT1020 - FINAL

Inscrivez tout de suite votre nom et code permanent.

Nom: _____| Prénom(s): _____|

Signature: _____| Code perm: _____|

Date : 31 juillet 2003

Durée: 3 heures (de 18h30 à 21h30) Local: 1140 du Pavillon André AISENSTADT.

Directives:

- Toute documentation permise.
- Calculatrice **non** permise.

- Répondre directement sur le questionnaire.
- Les réponses **doivent être clairement présentées.**

- L'examen compte 12 pages incluant celle-ci.

1._____/21

2._____/15

3._____/15

4._____/15

5._____/24

6._____/25

Total:_____/100

Question 1 (21 points)

1.1 A quoi sert l'option –classpath de la commande java permettant de lancer l'interpréteur java? De quelle nature peuvent être les éléments spécifiés à l'aide de cette option ?

Réponse

1.2 Soit une méthode de nom m1, méthode publique sans paramètre et sans type de retour. Le code de cette méthode est susceptible de lever deux types d'exceptions : des exceptions de classe E1 sous classe de la classe java.lang.Exception et des exceptions de classe R1 sous classe de la classe java.lang.RuntimeException. Donner la ou les signatures possibles pour cette méthode sachant que dans le corps de la méthode m1 aucun traitement des exceptions n'est effectué.

Réponse

1.3 Quel est l'intérêt de typer une référence avec une interface plutôt qu'avec son type concret comme par exemple dans la déclaration suivante ?

```
Collection x = new ArrayList();
```

Réponse

1.4 Si le "classpath" contient le seul répertoire "a/b", dans quel répertoire seront recherchées les classes du paquetage "udm.diro.ift1020.tps" ?

Réponse

1.5 Répondre par vrai ou faux aux questions suivantes et justifier votre réponse.

1.5.1 Le code d'une méthode d'instance peut utiliser toutes les variables d'instances de sa classe.

1.5.2 Le code d'une méthode d'instance peut utiliser toutes les variables static de sa classe.

1.5.3 Le code d'une méthode static peut utiliser toutes les variables d'instances de sa classe.

1.6 Quelle est la séquence affichée en sortie par le fragment de code suivant (qui compile et s'exécute correctement) et dites pourquoi.

```
Double a = new Double(Double.NaN);
Double b = new Double(Double.NaN);
if( Double.NaN == Double.NaN )
    System.out.println("True");
else
    System.out.println("False");
if( a.equals(b) )
    System.out.println("True");
else
    System.out.println("False");
```

Question 2 (15 points) Voici le code de 3 classes disposées dans 3 fichiers différents:

```
public class A {
  private int i;
  public int getI() {
    return i;
  }
}
```
A.java

```
public class B extends A {
  public int getI() {
    return super.getI() + 2;
  }
}
```
B.java

```
class Test {
   public static void main(String[] args) {

     // [Ligne à remplacer, voir question ci-dessus]

     System.out.print(a.getI());
     System.out.print(((A)a).getI());
     System.out.println(((B)a).getI());
   }
}
```
Test.java

2.1 On introduit à la place du commentaire, la première instruction de la méthode main du fichier Test.java, les deux instructions ci-dessus une à la fois. Pour ces deux cas de figures, cochez la bonne réponse et répondre à la question associée:

a) La classe Test ne compile pas. Indiquez la ligne qui provoquera l'erreur et dire pourquoi.

b) La classe Test compile mais provoque une erreur à l'exécution. Indiquez la ligne qui provoquera l'erreur et dire pourquoi.

c) La classe Test compile et affiche ..... Indiquez ce qui sera affiché et pourquoi.

2.1.1 `B a = new A();`

2.1.2 `A a = new B();`

2.2 Voici le nouveau code des 3 classes disposées dans 3 fichiers différents:

`public class A {`
`  private int i;`
`  public A(int i) {`
`  this.i = i;`
`  }`
`  public int getI() {`
`  return i;`
`  }`
`}`
	A.java

`public class B extends A {`
`  public int getI() {`
`  return super.getI() + 2;`
`  }`
`}`
	B.java

`class Test {`
`  public static void main(String[] args) {`
`  A a = new A(5);`
`  System.out.print(a.getI());`
`  System.out.print(((A)a).getI());`
`  System.out.println(((B)a).getI());`
`  }`
`}`
	Test.java

Cochez la bonne réponse et répondre à la question associée.

a) Une des classes ne compile pas. Indiquez la ligne qui provoquera l'erreur et dire pourquoi

b) Le code compile mais provoque une erreur à l'exécution. Indiquez la ligne qui provoquera l'erreur et dire pourquoi.

c) Le code compile et affiche ..... Indiquez ce qui sera affiché et pourquoi.

Réponse

Question 3 (15 points) Une classe `A` définit une méthode `m()` et redéfinit la méthode `toString()`. Le profil de la méthode `m()` est : "`public A m()`".

Voici une portion de code d'une autre classe `B` :

```
java.util.List liste = new java.util.ArrayList();
// les lignes qui suivent ont été omises ;
// elles remplissent la liste avec des instances de A
. . .
String s;
for (int i = 0; i < liste.size(); i++) {
    System.out.println(liste.get(i).m());
    s += liste.get(i).toString();
}
```

classe B

3.1 Indiquez les erreurs éventuelles du code ci-dessus. Pour chaque erreur, dites si le code provoquera une erreur à la compilation ou à l'exécution, et expliquez le problème.

3.2 Corrigez les erreurs et, pour chaque ligne à l'intérieur de la boucle, précisez quelles méthodes seront appelées (y compris les appels implicites) en donnant le nom de la méthode et la classe dans laquelle elle est définie.

Remarque importante: ne corrigez que le code qui empêche le programme de s'exécuter ; ne touchez pas au reste du code, même si vous pensez que le résultat n'est pas celui qui vous semble le meilleur.

Question 4 (24 points) Écrire deux programmes (donc contenant aussi la méthode main) permettant de couper un gros fichier en plusieurs parties puis de le reconstituer.
Le programme coupe reçoit le nom du fichier à couper et la taille en Ko (1024) des fichiers à créer en paramètres. Par exemple, on écrira :

```
java coupe test.txt 1400
```

Pour obtenir des fichiers de 1400 Ko qui pourront être copiés sur une disquette. On suppose que le fichier âssé en argument (ici test.txt) se trouve dans le même répertoire que coupe.class. Les fichiers obtenus seront nommés :

test.txt.segment000
test.txt.segment001
test.txt.segment002
test.txt.segment003
etc. …

Pour reconstituer le fichier de départ, on écrira :

```
java recolle test.txt
```

Les fichiers obtenus précédemment doivent se trouver dans le même répertoire que recolle.class.

4.1. coupe.java

4.2. recolle.java

Question 5 (25 points) Le but de cet exercice est d'implanter les méthodes de l'interface Iterator décrite dans l'interface Collection pour un cas bien particulier. On désire donc implanter Iterator pour une classe dont toutes les instances partagent un vecteur aléatoire. L'itérateur doit permettre une « promenade » sur les valeurs du vecteur aléatoire.

- Soit la classe MonIterateur qui implémente l'interface Iterator. La classe MonIterateur est décrite comme suit :

Attributs

- v est un tableau du type double.
- size est la taille du tableau v.
- position est un indice dans le tableau v, permettant de donner la position courante.

Méthodes

- MonIterator(int length) : Ce constructeur permet d'initialiser le tableau v dont la taille est passé en argument (length), avec des valeurs aléatoires. Il se charge aussi d'initialiser la variable position à 0.
- boolean hasNext() : Cette méthode permet de vérifier s'il y a un élément qui suit.
- Object hasNext() : Cette méthode permet de pointer l'élément suivant.
- void remove() : Cette méthode permet de retirer l'élément courant.

La méthode remove de l'interface Iterator si elle est invoquée sans être implantée, c'est l'exception UnsupportedOperationException qui sera levée. Si dans le cas où elle est implantée et elle est invoquée avant l'appel à la méthode next pour l'itération en cours l'exception IllegalStateException sera levée. À signaler que les deux exceptions sont classées dans la catégorie unchecked de java.lang.

- La classe TestMonIterateur permet de tester les différentes méthodes de la classe MonIterateur. L'appel suivant :

java TestMonIterateur 10

Cet appel permet de tester ces méthodes pour un tableau de taille 10.

Écrire les classes MonIterateur et TestMonIterateir et leurs méthodes respectives. Toutes les méthodes doivent être définies y compris la méthode remove.