

Examen de Programmation

(janvier 2010)

durée trois heures; tous les documents sont interdits

2 pages

Le qcm est sur des feuilles à part. Le barème est donné à titre indicatif.

1. Le type abstrait de données *Pile* est défini par les opérations *pop* (qui dépile l'élément au sommet de la pile), *push* (qui empile un élément), et *estVide* (qui teste si la pile est vide).
Définir une interface java paramétrée par un type T pour une pile de T. (1pt).

2. On considère la classe *Item* paramétrée par un type :

```
class Item<T> {  
    private T Val;  
    private Item Suiv;  
    public Item(T t) {  
        Val = t;  
        Suiv = null;  
    }  
    public T getVal() {  
        return Val;  
    }  
    public Item getSuiv() {  
        return Suiv;  
    }  
    public void setVal(T t) {  
        Val = t;  
    }  
    public void setSuiv(Item n) {  
        Suiv = n;  
    }  
}
```

Définir une implémentation *PileItem* (paramétrée par un type T) qui implémente l'interface *Pile* précédente en utilisant la classe *Item*. (2pts)

Pour cette question et la suivante, l'implémentation devra bien sûr assurer les propriétés usuelles des opérations *pop*, *push* et *estVide*, et on ne se préoccupera pas des cas où ces opérations ne sont pas définies.

3. On veut définir une implémentation *PileTab* (paramétrée par un type T) qui implémente l'interface *Pile* précédente à partir d'un tableau de taille fixe.

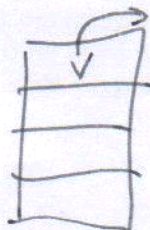
Est-il possible de représenter la pile par un tableau de T? Si ce n'est pas le cas, comment peut-on contourner le problème et avec quel(s) inconvénient(s)? (1pt)

Définir une implémentation *PileTab* qui contiendra un constructeur ayant un paramètre entier indiquant la taille maximale de la pile (et du tableau utilisé pour sa représentation). (1pt)

4. Si *p* est déclaré comme *Pile<Integer>* (et instancié par un objet d'une classe qui implémente *Pile*), *p.push("un")* est-il possible (=ne provoque pas d'erreur à la compilation)? Même question pour *p* déclaré comme *PileTab<Integer>* et *PileItem<Integer>*. (1pt)

Si `p` est déclaré comme `Pile` (et instancié par un objet d'une classe qui implémente `Pile`), `p.push("un")` est-il possible (=ne provoque pas d'erreur à la compilation)? Même question pour `p` déclaré comme `PileTab` et `PileItem`. (1pt)

5. On s'intéresse au traitement des opérations non définies (`pop` sur une pile vide et `push` sur une pile pleine (pour `PileTab`)), pour cela on définira deux exceptions `PileVideException` et `PilePleineException`. Modifier les classes précédentes de façon à ce qu'en cas d'opérations non définies les exceptions `PilePleineException` ou `PileVideException` soient lancées. En maintenant la hiérarchie des classes est-il possible de ne pas modifier l'interface `Pile`? Les modifications faites ont-elles des conséquences sur les programmes déjà écrits qui utilisent `Pile`, `PileItem` ou `PileTab`? (2pts)
6. Définir une méthode statique `cmp` ayant comme argument deux piles et qui vérifie que ces deux piles sont composées des mêmes éléments. (0,5pt)
En quoi cette question peut être considérée comme ambiguë? (0,5pt)
La méthode que vous avez écrite modifie-t-elle les piles qui lui sont passées en argument? Si c'est le cas proposez des solutions pour contourner le problème. (1pt)
7. Définir une extension de `PileTab` qu'on appellera `PileTabUn` pour laquelle :
 - (a) le clonage est possible et crée une copie de la pile. On demande de procéder de façon à ce que les invariants classiques des piles s'appliquent aussi bien à la copie qu'au clone. En particulier, pour le type de données pile, pour une pile `p`, un `p.pop()` qui suit un `p.push(v)` (sans autres opérations sur la pile `p` entre ce `push` et ce `pop`) le `pop` doit retourner un `v` et, par exemple, dans votre implémentation, si `c` est un clone de `a`, alors pour un morceau de code comme `c.push(y)`; `a.push(x)`; `c.pop()`; `a.pop()`; alors `c.pop()` retourne `y` et `a.pop()` retourne `x`. (1pt)
 - (b) il est possible pour un objet `p` de la classe `PileTabUn` d'utiliser une construction de la forme `for(Object o : p){ ... }`. (On rappelle qu'il faut pour cela que la classe implémente l'interface `Iterable` qui déclare la méthode `iterator()` retournant un `Iterator` définissant les méthodes `hasNext`, `next`, et `remove`) (2pts)
8. Définir une classe `PileTabBis` extension de `PileTabUn` telle qu'en cas d'un `push` sur une pile pleine au lieu de lancer `PilePleineException`, la taille du tableau de données pour représenter la pile soit augmentée. Pour cela, on redéfinira (entre autres choses) `push` de façon à appeler `push` de `PileTabUn` et à attraper l'exception `PilePleineException` et, dans ce cas, augmenter la taille du tableau. (2pts)
9. On va dans ce qui suit utiliser les piles définies précédemment de façon concurrente.
 - (a) Définir une extension `PileThread` de `Thread` contenant une pile `p`, et telle qu'un objet de cette classe empile successivement tous les entiers de 0 à 99. Cette classe contiendra un constructeur ayant comme argument une `Pile` qui sera affectée à `p`. (1pt)
 - (b) Définir un code qui crée et lance deux threads objets de `PileThread` partageant une pile `p` initialement vide. (0,5pt)
 - (c) Écrire un programme réalisant la même chose en utilisant l'interface `Runnable` sans créer explicitement d'extension de `Thread`. (1pt)
 - (d) Si on considère `p` après la terminaison de ces deux threads, est-on sûr que la pile contiendra 200 éléments? Si ce n'est pas le cas comment modifier l'implémentation de pile utilisée pour le garantir? (2pt)
 - (e) On définit maintenant une méthode `pushpop` qui réalise un `push` suivi d'un `pop`. Si plusieurs threads travaillent sur la même pile, est-on sûr que la valeur retournée par le `pop` soit celle qui était en argument du `push`? Comment le garantir? (3pts)



6. On considère le code suivant:

```
class A {private void g() {System.out.println("A:f()"); }}
class B extends A {void g() {System.out.println("B:g()"); }}
class C extends B {void g() {System.out.println("C:g()"); }}
```

et la séquence de code `A a = new C(); ((C)a).g();` quelle est la bonne réponse:

- (a) Le code provoque une erreur à la compilation ou à l'exécution
- (b) Il affiche C:g()
- (c) Il affiche A:f()
- (d) Il affiche B:g()

*

7. La classe paramétrée: `class X <T>{public T val; X(T x){val=x;}}` avec la méthode:
`public static void somme(X<? extends Number> x){System.out.println(x.val.doubleValue());}`
et le code: `X<Integer> x= new X<Integer>(2); somme(x);`

- (a) provoque une erreur à la compilation ou à l'exécution
- (b) affiche i=2.0

8. L'interface: `interface I{int i;void f();}` avec le morceau de code:
`(new I){int i=0; public void f(){System.out.println("bravo "+ i);}}.f();`

- (a) provoque une erreur à la compilation
- (b) affiche bravo 0

9. Le code:

```
class A{
    int i=10; void f(){System.out.print(" i="+i);}
}
class B extends A{
    static int i=0; void f(){System.out.print("i="+i);super.f(); }
}
```

avec le morceau de code: `A a=new B();a.f();`

- (a) provoque une erreur à l'exécution
- (b) provoque une erreur à la compilation
- (c) affiche i=10 i=10
- (d) affiche i=0 i=10

10. Les classes:

```
class D {public int val; public D(int i){val=i;}}
class E extends D{public E(int i){super(i);}}
```

avec le code :

```
D[] t1= new E[4]; t1[0]=new D(1); t1[1]=new E(3); System.out.println(t1[0].val+" "+t1[1].val);
```

- (a) provoque une erreur à la compilation
- (b) affiche i 3
- (c) provoquent une erreur à l'exécution

*

11. La classe paramétrée: `class X <T>{public T val; X(T x){val=x;}}` avec la méthode:
`public static void somme(X<Number> x){System.out.println(x.val.doubleValue());}`
et le code: `X<Integer> x= new X<Integer>(2); somme(x);`

- (a) provoque une erreur à la compilation ou à l'exécution
- (b) affiche i=2.0

12. L'interface: `interface I{static int i=0; void f();}` avec le morceau de code:
`(new I){int i=2; public void f(){System.out.println("bravo"+ i);}}.f();`

- (a) provoque une erreur à la compilation
- (b) affiche bravo 2
- (c) affiche bravo 0

13. (Trouver la ou les bonnes réponses) On considère la classe:
 class X {static int i=0; int j=2; /*...*/}

- (a) On peut ajouter à cette classe: void f(){i=i+j;}
- (b) On peut ajouter à cette classe: static void g(){i=j+i;}
- (c) On peut ajouter à cette classe: static void h(){i=X.j+i;}
- (d) On peut ajouter à cette classe: static void i(){this.i=this.i+this.j;}

14. On considère les classes:

```
class D {public int val; public D(int i){val=i;}}
class E extends D {public E(int i){super(i);}}
```

le code :

```
D[] t1= new D[4]; t1[0]=new D(1); t1[1]=new E(3); System.out.println(t1[0].val+" "+t1[1].val);
```

- (a) provoque une erreur à la compilation
- (b) affiche 1 3
- (c) provoquent une erreur à l'exécution

15. Le code:

```
class A {
  static int i=0; void f(){System.out.print(" i="+i);}
}
class B extends A {
  static int i=5; static void f() {System.out.println(" i="+i);}
}
```

avec le morceau de code: A a=new B();a.f();

- (a) provoque une erreur à l'exécution
- (b) provoque une erreur à la compilation
- (c) affiche i=0
- (d) affiche i=5

16. L'interface: interface I {void f();} avec le morceau de code:
 (new I){public void f(){System.out.println("bravo");}}.f();

- (a) provoque une erreur à la compilation
- (b) affiche bravo

17. Pour la classe D définie comme suit:

```
class D {
  public int x;
  public D() {x=3; };
  public D( int a){this(); x=x+a;};
  public D( int a, int b){this(b); x= x-a; }
}
```

$$\begin{array}{r}
 x = 3 \\
 \hline
 a = 6 \\
 \hline
 a = 9 \\
 b = 6 \\
 \hline
 x = x + 6 = 9 \\
 x = x - 5 = 4
 \end{array}$$

qu'affichera le code suivant?

```
D a=new D(5,6);
System.out.println(a.x);
```

- (a) 1
- (b) 2
- (c) 3
- (d) 4

18. On considère le code suivant:

```
class A {void f() {System.out.println("A:f()"); }}
class B extends A {void g() {System.out.println("B:g()"); }}
class C extends B {void g() {System.out.println("C:g()"); }}
```

et la séquence de code A a = new C(); a.g(); quelle est la bonne réponse:

- (a) Le code provoque une erreur à la compilation
- (b) Le code provoque un erreur à l'exécution
- (c) Il affiche C:g()
- (d) Il affiche A:f()
- (e) Il affiche B:g()

19. On considère le code:

```
class A1 {void f(){System.out.println(i);}int i=0;}
class B1 extends A1{public void f(){System.out.println(i);}int i=1;}
class B2 extends A1{public void f(){System.out.println(i);}int i=2;}
```

Le code suivant: "A1 a1= new B1(); B2 b2=(B2)a1;b2.f();" "

- (a) provoque une erreur à la compilation
- (b) provoque une erreur à l'exécution
- (c) affiche 1
- (d) affiche 2
- (e) affiche 0

20. La classe paramétrée: class X <T>{public T val; X(T x){val=x;}}
avec la méthode: public static void somme(X<Number> x){System.out.println(x.val.doubleValue());}
et le code: X<Number> x= new X<Number>(new Integer(2)); somme(x);

- (a) provoque une erreur à la compilation ou à l'exécution
- (b) affiche i=2.0

21. Soit:

```
class T{
    public int i=1;int j=2;
}
```

```
class U extends T{
    public int i=10;
}
```

On considère les instructions suivantes: T x=new U();
System.out.println("x.i="+x.i+" ((T)x).i="+((T)x).i+" ((U)x).i="+((U)x).i);
Trouver la ou les bonnes réponses:

- (a) ces instructions ne passeront pas à la compilation
- (b) ces instructions provoqueront une erreur à l'exécution
- (c) ces instructions provoqueront l'affichage: x.i=1 ((T)x).i=1 ((U)x).i)=10
- (d) ces instructions provoqueront l'affichage: x.i=10 ((T)x).i=1 ((U)x).i)=10
- (e) ces instructions provoqueront l'affichage: x.i=10 ((T)x).i=10 ((U)x).i)=10

22. On considère le code suivant:

```
class A {void g() {System.out.println("A:g()"); }}
class B extends A {void g() {System.out.println("B:g()"); }}
class C extends B {void g() {System.out.println("C:g()"); }}
```

et la séquence de code A a = new B(); ((A)a).g(); quelle est la bonne réponse:

- (a) Le code provoque une erreur à la compilation ou à l'exécution
- (b) Il affiche A:g()
- (c) Il affiche B:g()

23. Le code:

```
class A{
    static int i=0;int j=10;
    static void g(){System.out.println(j);}
}
```

avec le morceau de code : (new A()).g();

- (a) provoque une erreur à la compilation
- (b) affiche 10
- (c) provoque une erreur à l'exécution

24. La classe paramétrée: class X <T>{public T val; X(T x){val=x;}}
avec la méthode: public static void somme(X<?> x){System.out.println(x.val.doubleValue());}
et le morceau de code: X<Number> x= new X<Number>(2); somme(x);

- (a) provoque une erreur à la compilation ou à l'exécution
- (b) affiche i=2.0

25. On considère le code:

```
interface I{void f();}
class A implements I{public void f(){System.out.println("A");}}
class B implements I{public void f(){System.out.println("B");}}
```

Le code suivant: "I[] tabl=new A[2]; tabl[0]= new A();tabl[1]= new B(); tabl[0].f();" 7

- (a) provoque une erreur à la compilation
- (b) provoque une erreur à l'exécution
- (c) affiche A