Answer Key for Exam A

Section 1. Questions fondamentales: références

Bonne réponse = 1pt; mauvaise réponse ou réponse incomplète =-1pt;

Reportez vos réponses dans les cases du tableau de la page des réponses pour cette section. Dans le cas où aucune réponse ne vous semble correcte, vous pouvez laisser la case du tableau vide.

Dans toutes les questions suivantes, on suppose que la définition de la classe ci-dessous est dans la portée et dans le même paquetage que les divers codes proposés,

Pour toutes les questions de cette section on considère la classe suivante:

```
class A implements Cloneable {
  int i = 0;
  A a;
  A(int j) {
    i = j;
  }
  boolean equals(A x) {
    return (i == x.i && a == x.a);
  }
  public A clone() {
    Object o = null;
    try {
      o = super.clone();
    }catch (CloneNotSupportedException e) {}
    return (A) o;
  }
}
```

1. Le code:
   `A u1=new A(4); A u2; u1.a= u2;`
   `System.out.println("u1.a.i"+u1.a.i);`
   Provoque:
   - [x] (a) Une erreur à la compilation ou à l'exécution
   - [ ] (b) L'affichage de 0

2. Le code:
   `A y1=new A(1); A y2 = new A(1);`
   `if (y1.equals(y2)) System.out.println("oui"); else System.out.println("non");`
   Provoque:
   - [ ] (a) Une erreur à la compilation ou à l'exécution
   - [x] (b) L'affichage de oui
   - [ ] (c) L'affichage de non

3. Le code:
   `A z1 = new A(2); A z2 = z1.clone();`
   `if (z2.equals(z1)) System.out.println("oui"); else System.out.println("non");`
   Provoque:
   - [ ] (a) Une erreur à la compilation ou à l'exécution
   - [x] (b) L'affichage de oui
   - [ ] (c) L'affichage de non

4. Le code:
   A x1=new A(1); A x2=x1; x2.a=x1; x2.i=2; System.out.println(x1.i+" "+x2.i);
   Provoque:

   (a) Une erreur à la compilation ou à l'exécution

   (b) L'affichage de 1 2

   [(c)] L'affichage de 2 2

5. Le code:
   A y1=new A(1); A y2 = new A(1);
   if (y1 == y2) System.out.println("oui"); else System.out.println("non");
   Provoque:

   (a) Une erreur à la compilation ou à l'exécution

   (b) L'affichage de oui

   [(c)] L'affichage de non

6. Le code:
   A t1 = new A(4); A t2 = t1.clone(); A t3 = new A(1); t1.a = t3; A t4 = t3.clone(); t2.a = t4;
   if (t1.equals(t2)) System.out.println("oui"); else System.out.println("non");
   Provoque:

   (a) Une erreur à la compilation ou à l'exécution

   (b) L'affichage de oui

   [(c)] L'affichage de non

Section 2. Questions fondamentales: héritage

Bonne réponse=1pt; mauvaise réponse ou réponse incomplète =-1pt;
Reportez vos réponses dans les cases du tableau de la page des réponses pour cette section. Dans le cas où aucune réponse ne vous semble correcte, vous pouvez laisser la case du tableau vide.

Dans toutes les questions suivantes, on suppose que les définitions de classes ci-dessous sont dans la portée et dans le même paquetage que les codes proposés.

```
interface HI {
	String tout();
}
class HA implements HI {
	String s1;
	String s2;
	HA(String s1, String s2) {
		this.s1 = "HA:s1=" + s1;
		this.s2 = "HA:s2=" + s2;
	}
	HA() {
		this("un", "deux");
	}
	void g() {
		System.out.println("HA:g " + tout());
	}
	public String tout() {
		return ("(" + s1 + " " + s2 + ")");
	}
}
class HB extends HA {
	String s1;
	String s3;
	HB(String s, String t, String u, String v) {
		super(s, t);
		this.s1 = "HB:s1=" + u;
		this.s2 = "HB:s2=" + t;
		this.s3 = "HB:s3=" + v;
	}
	HB() {
		this("un", "deux", "trois", "quatre");
	}
	String f() {
		return ("(" + s1 + " " + s3 + ")");
	}
	public String tout() {
		return ("(" + super.tout() + f() + ")");
	}
	void g() {
		System.out.println(tout());
	}
}
```

1. Le code: HB b1= new HB(); System.out.println("b1.s1: "+b1.s1);
   (a) Provoque une erreur à la compilation ou à l'exécution
   (b) Affiche: b1.s1: HA:s1=un
   [(c)] Affiche: b1.s1: HB:s1=trois
2. Le code: HI i2=new HA(); ((HA) i2).g();
   (a) Provoque une erreur à la compilation
   (b) Provoque une erreur à l'exécution
   [(c)] Affiche HA:g (HA:s1=un HA:s2=deux)
3. Le code: HA a7=new HA(); a7.g();
   (a) Provoque une erreur à la compilation
   (b) Provoque une erreur à l'exécution
   (c) Affiche ((HA:s1=un HB:s2=deux)(HB:s1=trois HB:s3=quatre))
   [(d)] Affiche HA:g (HA:s1=un HA:s2=deux)
   (e) Affiche HA:g (HA:s1=un HB:s2=deux)
4. Pour cette question on rajoute dans la classe HB le code suivant:
   public void g(String st){System.out.print(st);g();}
   (a) Cet ajout provoque une erreur à la compilation
   [(b)] Cet ajout ne provoque pas d'erreur à la compilation et le code HA a8=new HB(); a8.g("bonjour"); provoque une erreur à la compilation
   (c) Cet ajout ne provoque pas d'erreur à la compilation et le code HA a8=new HB(); a8.g("bonjour"); affiche bonjour((HA:s1=un HB:s2=deux)(HB:s1=trois HB:s3=quatre)))
5. Le code: HB b5=new HB(); System.out.println("b5.s1: " + ((HA) b5).s1);
   (a) Provoque une erreur à la compilation
   (b) Provoque une erreur à l'exécution
   [(c)] Affiche: b5.s1: HA:s1=un
   (d) Affiche: b5.s1: HB:s1=trois
6. On suppose que l'on ajoute une classe X:

```
class X {
  public static void afficher(HI i) {
    System.out.println(i.tout());
  }
  public static HI creerI(String ch) {
    return new HI() {
      public String tout() {return ch; }
    };
  }
}
```

   (a) Cet ajout provoque une erreur à la compilation
   (b) Cet ajout ne provoque pas d'erreur à la compilation et le code X.afficher(X.creerI("bonjour")); provoque une erreur à la compilation
   [(c)] Cet ajout ne provoque pas d'erreur à la compilation et le code X.afficher(X.creerI("bonjour")); affichera: bonjour

7. Le code: HA a4 = new HA(); HB b4 = (HB) a4; System.out.println("b4.s2: " + b4.s2);

   (a) Provoque une erreur à la compilation

   [(b)] Provoque une erreur à l'exécution

   (c) Affiche: b4.s2: HA:s2=deux

   (d) Affiche: b4.s2: HB:s2=deux

8. Pour cette question on rajoute dans la classe HB le code suivant:
   public void g(String st){System.out.print(st);g();}

   (a) Cet ajout une erreur à la compilation

   (b) Cet ajout ne provoque pas d'erreur à la compilation et le code
   HB b8 = new HB(); b8.g("bonjour"); provoque une erreur à la compilation

   [(c)] Cet ajout ne provoque pas d'erreur à la compilation et le code
   HB b8 = new HB(); b8.g("bonjour"); affiche
   bonjour((HA:s1=un HB:s2=deux)(HB:s1=trois HB:s3=quatre)))

9. Le code: HA a3= new HB(); HB b2=a3; System.out.println("b2.s2: "+b2.s2);

   [(a)] Provoque une erreur à la compilation

   (b) Provoque une erreur à l'exécution

   (c) Affiche: b2.s2: HA:s2=deux

   (d) Affiche: b2.s2: HB:s2=deux

10. Le code: HA a6=new HB(); ((HA)a6).g();

    (a) Provoque une erreur à la compilation

    (b) Provoque une erreur à l'exécution

    [(c)] Affiche ((HA:s1=un HB:s2=deux)(HB:s1=trois HB:s3=quatre))

    (d) Affiche HA:g (HA:s1=un HA:s2=deux)

    (e) Affiche HA:g (HA:s1=un HB:s2=deux)

11. On suppose que l'on ajoute une classe X:

```
class X {
  public static void afficher(HI i) {
    System.out.println(i.tout());
  }
  public static HI creerI(String ch) {
    return new HI() {
      public String tout() {return ch; }
    };
  }
}
```

    (a) Cet ajout provoque une erreur à la compilation

    (b) Cet ajout ne provoque pas d'erreur à la compilation et le code X.afficher(X.creerI(new HB().tout())); provoque une erreur à la compilation

    [(c)] Cet ajout ne provoque pas d'erreur à la compilation et le code X.afficher(X.creerI(new HB().tout())); affichera: ((HA:s1=un HB:s2=deux)(HB:s1=trois HB:s3=quatre))

12. Le code: HA a3= new HB(); HB b3=(HB)a3; System.out.println("b3.s2: "+b3.s2);
    (a) Provoque une erreur à la compilation
    (b) Provoque une erreur à l'exécution
    (c) Affiche: b3.s2: HA:s2=deux
    [(d)] Affiche: b3.s2: HB:s2=deux

13. Le code: HA a5=new HB();a5.g();
    (a) Provoque une erreur à la compilation
    [(b)] Affiche ((HA:s1=un HB:s2=deux)(HB:s1=trois HB:s3=quatre))
    (c) Affiche HA:g (HA:s1=un HA:s2=deux)
    (d) Affiche HA:g (HA:s1=un HB:s2=deux)

14. Le code: HI i1= new HB(); System.out.println(i1.tout());
    (a) Provoque une erreur à la compilation ou à l'exécution
    (b) Affiche: (HA:s1=un HA:s2=deux)
    (c) Affiche: ((HA:s1=un HA:s2=deux)(HB:s1=trois HB:s3=quatre))
    [(d)] Affiche: ((HA:s1=un HB:s2=deux)(HB:s1=trois HB:s3=quatre))

15. Le code: HA a2= new HB(); System.out.println("a2.s3: "+a2.s3);
    [(a)] Provoque une erreur à la compilation
    (b) Provoque une erreur à l'exécution
    (c) Affiche: a2.s3: HA:s1=trois

16. Le code:HI i=new HA("x","y"); System.out.println(i.tout());
    (a) Provoque une erreur à la compilation ou à l'exécution
    [(b)] Affiche: (HA:s1=x HA:s2=y)
    (c) Affiche: (HA:s1=un HA:s2=deux)

17. Le code: HI i2=new HA(); i2.g();
    [(a)] Provoque une erreur à la compilation
    (b) Provoque une erreur à l'exécution
    (c) Affiche HA:g (HA:s1=un HA:s2=deux)

18. On suppose que l'on ajoute une classe X:

```
class X {
  public static void afficher(HI i) {
    System.out.println(i.tout());
  }
}
```

    (a) Cet ajout provoque une erreur à la compilation
    (b) Cet ajout ne provoque pas d'erreur à la compilation et le code X.afficher(new HA()); provoque une erreur à la compilation
    [(c)] Cet ajout ne provoque pas d'erreur à la compilation et le code X.afficher(new HA()); affichera (HA:s1=un HA:s2=deux)

19. Peut-on ajouter à la classe HB le méthode définie par:
    void g() {System.out.println("(" + s1 + " " + s3 + ")");}

    (a) oui

    [(b)] non

20. Le code: HA a1=new HB(); System.out.println("a1.s1: "+a1.s1);

    (a) Provoque une erreur à la compilation ou à l'exécution

    [(b)] Affiche: a1.s1: HA:s1=un

    (c) Affiche: a1.s1: HB:s1=trois

Section 3. Programmation

L'interface Iterator<E> contient les méthodes boolean hasNext() et E next() (les autres méthodes peuvent ne pas être implémentées). De même on rappelle que l'interface Iterable<T> contient une seule méthode Iterator<T> iterator() qui doit être implémentée.

On considère l'interface suivante:

```
interface PileFile<E> extends Iterable<E> {
  void mettre(E elem);
  E prendre();
  boolean estVide();
  boolean estPlein();
  Iterator<E> iterator();
}
```

Cette interface doit servir à la réalisation de piles et de files. On rappelle qu'une pile ou une file construite sur le type E est une structure de données contenant des objets de type E. Pour une pile l'opération, prendre retournera l'objet qui a été mis *le plus récemment* dans cette pile par une opération mettre, pour une file prendre retournera l'objet qui a été mis *le plus anciennement* dans cette file par une opération mettre. La méthode estVide teste si la structure de données ne contient aucun élément. De même dans le cas où la structure de données ne peut contenir qu'un nombre borné d'éléments, la méthode estPlein teste si de nouveaux éléments peuvent être ajoutés à la structure de données.

Ici, un itérateur sur une pile ou une file ne doit pas modifier le contenu de la pile ou de la file; la méthode next retournera successivement les références sur les objets de la structure de données dans l'ordre du plus ancien au plus récent pour une file et du plus récent au plus ancien pour une pile.

1. Donner une implémentation PileTab<E> d'une pile de PileFile<E> avec un tableau de taille fixe donnée par une variable d'instance taille qui sera initialisée par le constructeur avec une valeur par défaut égale à 100. En cas d'opérations impossibles (mettre sur une pile pleine ou prendre sur une pile vide) on pourra lancer des exceptions Plein ou Vide que l'on définira.
   On fera attention à ce que l'implémentation proposée soit efficace (les opérations mettre et prendre doivent se faire en temps constant).

2. Donner une implémentation FileListe<E> d'une file de PileFile<E> utilisant un chainage défini par une classe qui sera interne à FileListe<E>:

   ```
   class Item {
     E val;
     Item next;
     Item(E v, Item n) {
       val = v;
       next = n;
     }
   }
   ```

   On fera attention à ce que l'implémentation proposée soit efficace (les opérations mettre et prendre doivent se faire en temps constant).

3. Définir une méthode statique ayant deux arguments le premier un Iterator et le deuxième une PileFile qui ajoute à la PileFile tous les éléments de la structure de données implémentant l'Iterator.
   Utiliser cette méthode pour définir une méthode statique permettant de copier le contenu d'une file pour en faire une pile.

Section 4. Questions diverses

Dans toutes les questions suivantes, on suppose que les définitions de classes ci-dessous sont dans la portée des divers codes proposés, on suppose aussi que ces codes sont dans le même paquetage que ces définitions de classes.

Bonne réponse=1pt; mauvaise réponse ou réponse incomplète =-0,5pt; absence de réponse =0pt

1. En java:

   [(a)] le type d'une expression est toujours déterminé à la compilation

   (b) le type d'une expression peut changer au cours de l'exécution

2. ```
   class A1 {
     int a = 5;
     B1 b=new B1(10);
     void f() {System.out.println("A1 " + a); b.f();}
     class B1 {
       B1(int i){a=i;}
       void f() {System.out.println("B1 " + a); }
     }
   }
   ```

   Et le code `(new A1()).f();`

   (a) provoquent une erreur à la compilation ou à l'exécution

   (b) affichent A1 5 et B1 10

   [(c)] affichent A1 10 et B1 10

3. Le code:

   ```
   Integer m=new Integer(2);
   if (m==2) System.out.println("égal"); else System.out.println("non égal");
   ```

   [(a)] affichera égal

   (b) affichera non égal

4. ```
   class X implements Cloneable {
     public int i = 8; public int[] ti = new int[3];
     public Object clone() {X tmp = null;
       try {tmp = (X) super.clone();tmp.ti = ti.clone();}catch (Exception e) {}
       return tmp; }
   }
   ```

   et le code:

   ```
   X x1= new X(); x1.ti[0] = 1;X x3 = (X) x1.clone(); x3.i = 3;x3.ti[0] = 3;
   System.out.println(x1.ti[0] + " " + x1.i);
   ```

   (a) provoquent une erreur à la compilation ou à l'exécution

   (b) affichent 3 3

   (c) affichent 3 8

   (d) affichent 1 3

   [(e)] affichent 1 8

5. class E1 extends Exception{}
   class E2 extends E1{}
   class ZZ {void f() throws E2{}}
   class TT extends ZZ{void f(){}}

   [(a)] Ces classes peuvent être compilées

   (b) Ces classes ne peuvent pas être compilées

6. En supposant que a et b sont des variables correctement déclarées "a=b+1" est:

   [(a)] une expression

   (b) une instruction

7. Le code:
   Number t[]=new Integer[10];t[0]=1;

   (a) provoque une erreur à la compilation

   (b) provoque une erreur à l'exécution

   [(c)] ne provoque ni erreur à l'exécution ni erreur à la compilation

8. 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) provoquent une erreur à la compilation

   (b) affichent 1 3

   [(c)] provoquent une erreur à l'exécution

9. Soit la classe : class A {public int i; }. Le code suivant A a=new A();Object o=a;((A)o).i=10;:

   [(a)] est correct (il peut être compilé et exécuté)

   (b) n'est pas correct (il ne peut pas être compilé et exécuté)

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

    et le code: (new A()).g();

    [(a)] provoquent une erreur à la compilation

    (b) affichent i=0 j=10

    (c) provoquent une erreur à l'exécution

11.
```
class A2 {
  int a = 5;
  B2 b=new B2(10);
  void f() {System.out.println("A2 " + a); b.f();}
  class B2 {
    int a;
    B2(int i){a=i;}
    void f() {System.out.println("B2 " + this.a); System.out.println("B2 " + A2.this.a); }
  }
}
```

Et le code (new A2()).f();

(a) provoquent une erreur à la compilation ou à l'exécution

(b) affichent A2 5 B2 10 et B2 10

(c) affichent A2 5 B2 5 et B2 5

[(d)] affichent A2 5 B2 10 et B2 5

12. Avec la méthode: "static void perm(Integer i,Integer j){Integer tmp; tmp=j; j=i; i=tmp;}" le morceau de code:
int i1=10,i2=5; perm(i1,i2); System.out.println(i1);

[(a)] affichera 10

(b) affichera 5

(c) provoquera une erreur à la compilation ou à l'exécution

13. Le code:
List<Number> ln=new ArrayList<Number>();
ln.add(10); ln.add(2.3);

(a) provoque une erreur à la compilation

(b) provoque une erreur à l'exécution

[(c)] ne provoque ni erreur à l'exécution ni erreur à la compilation

14. class A{static void f(){System.out.println("A.f");}}
class B extends A{void f(){System.out.println("B.f");}} et le code A a=new B(); a.f();

[(a)] provoquent une erreur à la compilation

(b) affichent A.f

(c) affichent B.f

15. Avec la méthode: " static Integer f(int i){Integer n=i; return n;}" le morceau de code:
"Integer n1=2; if (2==f(n1))System.out.println("égal"); else System.out.println("non égal"); "

[(a)] affichera égal

(b) affichera non égal

(c) provoquera une erreur à la compilation ou à l'exécution

16. public static void permuter (String s1, String s2, int x1, int x2){
    String tmp1=s1; s1=s2; s2=tmp1; int tmp2=x1; x1=x2; x2=tmp2;
    }

    Considérons: String a="bon"; String b="jour"; int c=3; int d =4; permuter(a,b,c,d);
    Quelles seront les valeurs de a,b,c,d après l'exécution de ce code?

    [(a)] "bon", "jour", 3, 4
    (b) "jour", "bon", 3, 4
    (c) "bon", "jour", 4, 3
    (d) "jour", "bon", 4, 3

17. Le code:
    List<Integer> li1=new ArrayList<Integer>();
    List<? super Integer> lb=li1;
    lb.add(4); System.out.println(lb.get(0));

    (a) provoque une erreur à la compilation
    (b) provoque une erreur à l'exécution
    [(c)] affiche 4

18. Le code:
    List<Number> ln=new ArrayList<Number>();
    ln.add(10); ln.add((Double)2.3);

    (a) provoque une erreur à la compilation
    (b) provoque une erreur à l'exécution
    [(c)] ne provoque ni erreur à l'exécution ni erreur à la compilation

19. class E1 extends Exception{}
    class E2 extends E1{}
    et l'appel de la fonction f définie ci-dessous

    ```
    void f(){
      try{throw new E1(); }
      catch (E2 e){System.out.println("E2");}
      finally {System.out.println("Fin");}
    }
    ```

    (a) affichent E2
    (b) affichent Fin
    (c) affichent E2 et Fin
    [(d)] provoquent une erreur à la compilation

20. Le code:
    List<Integer> li=new ArrayList<Integer>();
    li.add(3); List<?> lqm1= li;
    System.out.println(lqm1.get(0));

    (a) provoque une erreur à la compilation
    (b) provoque une erreur à l'exécution
    [(c)] affiche 3

21. class XX {void f(){}; }
class YY extends XX{void f() throws Exception{throw new Exception();}}

    (a) Ces classes peuvent être compilées

    [(b)] Ces classes ne peuvent pas être compilées

22. Le code:
List<Number> ln1=new ArrayList<Number>();
List<? super Integer> lnb=ln1;
lnb.add(5); System.out.println(lnb.get(0));

    (a) provoque une erreur à la compilation

    (b) provoque une erreur à l'exécution

    [(c)] affiche 5

23. public static Number somme1(List<Number> L){
  double tmp=0.0;
  for(Number n:L)tmp = tmp+ n.doubleValue();
return tmp;
}

    Et le code:
List<Number> lnum=new ArrayList<Number>();
for(int i=0;i<10;i++)lnum.add(i);
System.out.println(somme1(lnum));

    (a) provoquent une erreur à la compilation

    (b) provoquent une erreur à l'exécution

    [(c)] affichent 45.0

24. Le code:
List<Number> ln=new ArrayList<Integer>();ln.add(10) ;

    [(a)] provoque une erreur à la compilation

    (b) provoque une erreur à l'exécution

    (c) ne provoque ni erreur à l'exécution ni erreur à la compilation

25. public static Number somme1(List<Number> L){
  double tmp=0.0;
  for(Number n:L)tmp = tmp+ n.doubleValue();
  return tmp;
}

    Et le code:
List<Integer> lint=new ArrayList<Integer>();
for(int i=0;i<10;i++)lint.add(i);
System.out.println(somme1(lint));

    [(a)] provoquent une erreur à la compilation

    (b) provoquent une erreur à l'exécution

    (c) affichent 45.0

26. En java:

    (a) toute instruction a un type

    [(b)] toute expression a une valeur

27. Le code:
List<?> lqm=new ArrayList<Integer>(); lqm.add(2);

- [x] (a) provoque une erreur à la compilation
- [ ] (b) provoque une erreur à l'exécution
- [ ] (c) ne provoque ni erreur à l'exécution ni erreur à la compilation

28. Après les déclarations int i,j=0,k,l; le morceau de code "l=1+(k=i=j+2)" est:

- [ ] (a) est une instruction qui modifie les variables i, k et l
- [x] (b) est une expression de type int, qui vaut 3
- [ ] (c) n'est syntaxiquement pas correct

29. Avec la classe: "class B{int i;B(int i){this.i=i;}}" et la méthode:
" static void permute(B a,B b){B tmp=a;a=b;b=tmp; }"
le morceau de code:
B a=new B(3); B b=new B(0); permute(a,b);System.out.println(a.i);

- [x] (a) affichera 3
- [ ] (b) affichera 0
- [ ] (c) provoquera une erreur à la compilation ou à l'exécution

30. Le code:
Number t[]=new Integer[10];t[0]=3.2;

- [ ] (a) provoque une erreur à la compilation
- [x] (b) provoque une erreur à l'exécution
- [ ] (c) ne provoque ni erreur à l'exécution ni erreur à la compilation

31. En supposant que f est une méthode ayant un int en paramètre "f(1);" est:

- [ ] (a) une expression
- [x] (b) une instruction

32. Soit la classe : class A {public int i; }. Le code suivant A a=new A();Object o=a;o.i=10;:

- [ ] (a) est correct (il peut être compilé et exécuté)
- [x] (b) n'est pas correct (il ne peut pas être compilé et exécuté)