POO-IG
Programmation Orientée Objet et Interfaces Graphiques
Examen de session 1

6 Janvier 2020

Durée : 2 heures
(Correction)

Documents autorisés : trois feuilles A4 recto-verso manuscrites ou imprimées. Portables/Ordinateurs/Tablettes interdits.

IMPORTANT : Dans les questions à choix multiple vous devez **entourer toutes les réponses correctes** (il peut y en avoir plusieurs). Si l'ensemble des réponses entourées ne coïncide pas avec l'ensemble des réponses correctes la question ne donnera aucun point.
Dans toutes les questions les étoiles donnent une indication approximative de la difficulté (plus d'étoiles = exercice plus difficile).

Question 1 (**)

```
public static XXXX boolean recherche (T[] t, T e) {
    int i =0;
    while(i < t.length && t[i].compareTo(e) < 0) i++;
    return (i < t.length && t[i].compareTo(e) == 0);
}
```

Par quoi doit-on remplacer XXXX dans l'en-tête de la méthode `recherche` ?

Réponses possibles :

- × `<T>`
- → `<T extends Comparable<T>>`
- × `<T super Comparable<T>>`
- × `<? extends Comarable<T>>`

< T extends Comparable< T■

Question 2 (***)

Soit le code suivant

```
interface Cast<S,T>{
  T cast(S s);
}
public class A {
  public static void main (String[] args) {
    Cast<Integer , Cast<String , Boolean>> multiArg =  XXXX;
    System.out.println(multiArg.cast(7).cast("hello"));
  }
}
```

Par quoi doit-on remplacer XXXX dans le code ci-dessus pour que le main affiche `true` ?

Réponses possibles :

→ `(Integer i) -> (String s) -> s.length() < i`

× `(Integer i, String s) -> s.length() < i`

× `(i,s) -> s.length() < i`

× `(Integer i) -> true`

Question 3 (*)

Soit le code suivant :

```
public class Exo{

    public static double f(int n) throws Exception{
        if(n == 0)
            throw new Exception("nombre nul");
        if (n < 0)
            throw new IllegalArgumentException("nombre negatif");
        return Math.sqrt(n);
    }

    public static void main(String[] args){
        try{
            System.out.print(f(-1));
        }
        catch(IllegalArgumentException e){
            System.out.print(" !!!! ");
        }
        catch(Exception e){
            System.out.print(" Oh! ");
        }
        finally{
            System.out.print(" Fini! ");
        }
    }
}
```

Qu'écrit le programme ci-dessus ?

Réponses possibles :

× !!!! Oh !

× Oh ! Fini !

→ !!!! Fini !

× !!!! Oh ! Fini !

Question 4 (**)

```
class A{
  static int a = 1;
  public static void f(){
    a++;
```

```
  }
  public String g(A obj){
    return "A";
  }
}
public class B extends A {
  public static void f(){
    a += 2;
  }
  public String g(B obj){
    return "B";
  }
  public static void main (String[] args) {
    B b = new B();
    ((A)b).f();
    String res = ((A)b).g(b);
  }
}
```

Que valent respectivement les variables `a` et `res` après l'exécution du programme ?

Réponses possibles :

- × 2 et B
- → 2 et A
- × 3 et A
- × 3 et B

Question 5 (**)

Soit le code suivant :

```
  class A{
    int i;
    {
      System.out.println(i);
      i = 1;
    }
    A(){
      System.out.println(i);
    }
  }
  class B extends A{
    {
      i=3;
    }

    B(int j){
      System.out.println(i);
      i=j;
    }
    B(){
      this(2);
      System.out.println(i);
    }

    public static void main(String[] args){
      B b = new B();
```

```
    }
  }
```

Qu'affiche le code ci-dessus à l'exécution ?

Réponses possibles :

- × 0 1 0 2
- × 0 0 1 2
- → 0 1 3 2
- × 3 2

Question 6 (**)

```
abstract class A { public abstract void f();}
XXXX { public void f() {}}
```

Par quoi peut-on remplacer XXXX dans le code ci-dessus pour que le programme compile sans erreur ?

Réponses possibles :

- → `abstract class B extends A`
- × `class B implements A`
- → `class B extends A`
- × `abstract class B implements A`

Question 7 (*)

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

Par quoi peut-on remplacer XXXX pour que le programme compile ?

Réponses possibles :

- × `new A.B()`
- × `A.new B()`
- × `new a.B()`
- → `a.new B()`

Question 8 (**)

On définit deux nouvelles exceptions

```
class E1 extends Exception {}
class E2 extends Exception {}
```

et on considère le code suivant :

```
void f () throws E1,E2 {throw new E1();}
void g () XXXX {try {f();} catch (E1 e) {}}
```

Par quoi peut-on remplacer XXXX pour que le programme compile ?

Réponses possibles :

- × On ne rajoute rien.
- × throws E1
- → throws E2
- → throws E1,E2

Question 9 (**)

```
class A {
  void f(A a) {System.out.println("A");}
}
interface I {
  default void f(I i) {System.out.println("I");}
}
public class B extends A implements I {
  void f(B b) {System.out.println("B");}
  public static void main (String[] args) {
    I i = new B();
    i.f(new B()); i.f(i);
  }
}
```

Qu'affiche la méthode `main` ?

Réponses possibles :

- × B B
- → I I
- × B A
- × I A

Question 10 (*)

Ecrire, en langage Java, deux classes A et B telles que :

- La classe B hérite de la classe A ;
- la classe A possède :

  - Deux attributs entiers x et y visibles par ses classes filles et les classes du même package
  - Un constructeur affectant une valeur à l'attribut x ;
- La classe B possède :
  - un attribut entier z visible uniquement dans la classe B ;
  - un constructeur affectant une valeur aux attributs x et z ;

Ecrire ici le code

```
class A {
 protected int x;
 protected int y;
 A(int a){x=a;}
}
class B extends A{
 private int z;
 B(int a,int b){
  super(a);
  z=b;
 }
}
```

Question 11 (**)

Pour la méthode `multiaccept` ci-dessous, trouver les bons types pour les paramètres.

```
public static void multiaccept(List<? /**/ Number> l, Consumer<? /**/ Number> c){
  for (Number x : l){
    c.accept(x);
  }
}
```

Rappel : L'interface `Consumer<T>` est une interface qui a une méthode abstraite `void accept(T t)`.

Quels sont les types de deux paramètres de la méthode `multiaccept`

Réponses possibles :

× `List<? extends Number> l, Consumer<? extends Number> c`

→ `List<? extends Number> l, Consumer<? super Number> c`

× `List<? super Number> l, Consumer<? extends Number> c`

× `List<? super Number> l, Consumer<? super Number> c`