Apprenez à programmer en Java

40 heures Difficile

Mis à jour le 05/08/2019

Comprenez les bases en Java

Bravo ! Vous avez réussi cet exercice !

Compétences évaluées

- Utiliser la syntaxe Java
- Installer les outils de développement Java

Question 1

Qu'est-ce qu'un IDE ?

- [x] Un environnement permettant de développer des programmes.
- [ ] Un outil permettant d'exécuter des programmes écrits en Java.
- [ ] Une instruction du langage Java.
- [ ] Un site web référençant les fonctionnalités d'un langage de programmation.

***IDE** est l'acronyme de **I**ntegrated **D**evelopment **E**nvironment. Il s'agit donc d'un environnement permettant de développer des programmes informatiques.*

Question 2

Qu'est-ce qu'un JRE ?

- Un outil permettant de développer des programmes.
- ✓ Un environnement permettant d'exécuter des programmes écrits en Java.
- Une instruction du langage Java.
- Un site web référençant les fonctionnalités d'un langage de programmation.

***JRE** est l'acronyme de **J**ava **R**untime **E**nvironment, soit, un environnement où vos programmes Java pourront s'exécuter. C'est donc grâce à ceci que vos programmes pourront être utilisés sur vos ordinateurs. Sans cet environnement, impossible de faire fonctionner un programme écrit en Java !*

Question 3

Pouvez-vous développer un programme Java sans IDE ?

- Non
- Oui, avec Word
- ✓ Oui, avec n'importe quel éditeur de fichier texte (Notepad++, bloc note Windows...)
- Oui, avec Netbeans

*Vous pouvez créer des programmes Java avec n'importe éditeur de fichier texte (Notepad++, bloc note Windows etc). Cependant, il vous faudra alors compiler vos codes sources manuellement, en installant le **JDK** qui n'est que le JRE avec les outils de compilation.*

Question 4

Qu'est-ce qu'une variable ?

- ✓ Un conteneur permettant de stocker des données (entier, caractère, booléen...)
- Une instruction du langage Java
- Une entité permettant de manipuler des nombres
- Une entité permettant de manipuler des caractères

Les variables sont des réceptacles permettant à vos programmes de mémoriser des données (entier, booléen, caractère...) afin de pouvoir les manipuler, les modifier, les utiliser...

Question 5

Quel est la valeur de la variable ci-dessous :

```
int entier = 0xFE;
```

- Rien, cette variable est mal initialisée.
- Rien, FE n'est pas un entier mais des caractères !

✓ ◉ 254

◯ 32

*La variable contient la valeur **254**. 0xFE correspond à la notation hexadécimale de l'entier 254.*

Question 6

Ce code est-il correct ?

```
int i = 10;
int j = 12
int k = 0;
k = (i*i)*(j*j)/j+i;
```

◯ Oui, rien ne manque.

◯ Non, les noms de variables doivent avoir au moins deux caractères.

◯ Non, la variable **k** doit impérativement être de type double.

✓ ◉ Non, une des variables est mal déclarée !

*La variable **j** est mal déclarée : il manque le point virgule final de la déclaration, sans lui, le programme ne compilera pas !*

Question 7

Que vaut la variable result :

```
double i = 10;
double j = 3;
int result = i / j;
```

◯ 3.333333333333333333...

◯ 3.4

✗ ◉ 3

✓ ◯ Rien, il y a une erreur dans le programme...

Le calcul étant basé sur des doubles, le compilateur refusera de stocker le résultat dans un entier car il y a un risque de perte de précision... Le programme ne compilera donc pas car il y a une erreur...

Question 8

Qu'est-ce qui ne va pas ici :

```
string str = 'ma chaîne de caractères';
```

- Les caractères accentués sont interdits dans le type String.
- Il s'agit de l'objet String et non string.
- ✗ Il faut initialiser la variable avec des double quotes ("") et non des single quotes ('').
- ✓ Il s'agit de l'objet String et non string et il faut initialiser la variable avec des double quotes ("") et non des single quotes ('').

*En Java, **TOUS** les objets commencent par une majuscule. String étant un objet, son type commence donc par une majuscule. De plus, ce type s'initialise avec des double quotes et non des single quotes.*

Question 9

Que devez-vous utiliser pour récupérer les saisies clavier ?

- La classe scanner
- La classe sanner
- La classe Sanner
- ✓ La classe Scanner

C'est la classe Scanner, présente dans le package java.util, qui vous permet de récupérer les saisies clavier. Attention, il s'agit là aussi d'un objet, donc son nom commence par une majuscule !

Question 10

Que faut-il absolument faire lorsque vous utilisez des objets présents dans des packages autres que java.lang ?

- Rien de particulier
- Il faut importer tout le package.
- ✓ Il faut importer la classe dont vous avez besoin.

*Il faut impérativement importer la classe dont vous avez besoin pour que votre programme puisse y avoir accès. Si vous ne vous rappelez plus où se trouve la classe souhaitée, utilisez le raccourci clavier **CTRL+SHIFT+O** qui permettent de réorganiser les instructions **import***

Question 11

Qu'est-ce qui ne va pas, ici ?

```
Scanner sc = new Scanner(System.in);
System.out.println("Veuillez saisir un entier : ");
double d = sc.nextInt();
System.out.println("Vous avez saisi le nombre : " + d);
```

- Rien. Tout va bien.
- L'objet Scanner est mal initialisé.
- ✓ Il y a une incohérence entre la variable d et le type de retour de l'objet Scanner.
- La méthode nextInt() n'existe pas...

Il y a un problème de cohérence entre la variable qui va stocker la saisie clavier et la méthode qui récupère cette saisie : une variable double est utilisée sur un retour de type entier.

Question 12

Quelle instruction peut-on utiliser avec une condition du type if ...else afin de rajouter des conditions ?

- then
- elseif
- ✓ else if
- then if
- if else

*Dans un bloc d'instruction de type **if ...else** , il est possible de rajouter des conditions avec l'instruction **else if**.*

Question 13

Que va retourner ce programme ?

```
int nbre = 999;
if (nbre < 1000 && nbre > 10000)
 System.out.println("c'est bon !");
else
 System.out.println("ce n'est pas bon");
```

- C'est bon !
- ✓ Ce n'est pas bon !

*Le programme retourne « Ce n'est pas bon ! » car la condition n'est **jamais** vraie : aucun nombre ne peut à la fois être plus petit que 1000 **ET** plus grand que 10000...*

Question 14

Quel sera le résultat de ce code :

```
int a = 10, b = 20;
int max = (a < b) ? ((b < 20) ? b * 2 : ((b > 20) ? b % 3 : b / 4) ) : ((a == 10) ? a / 2 : a % 3);
```

- [x] 5
- [ ] 10
- [ ] 20
- [ ] 30
- [ ] 40

Le résultat vaut 5. Pour bien comprendre cette condition ternaire absolument imbuvable, décortiquez-la ainsi :

```
int max;
if (a < b){
 if(b < 20)
  max = b*2;
 else{
  if(b > 20)
   max = b % 3;
  else
   max = b / 4;
 }
}
else{
 if (a == 10)
  max = a / 2;
 else
  max = a % 3;
}
```

Avec la valeur de a et b, nous arrivons à l'instruction max = b / 4;

Question 15

Quelles sont les différences entre la boucle while et la boucle do... while ?

- [x] La boucle do...while s'exécute au moins une fois et la condition du while prend un ';' à la fin.
- [ ] La boucle while s'exécute au moins une fois et la condition du while prend un ';' à la fin.
- [ ] La boucle do...while est plus rapide que la boucle while.
- [ ] La boucle while est plus rapide que la boucle do...while.
- [ ] Aucune

La boucle do..while a cette particularité unique de s'exécuter au moins une fois, ceci car la condition qui régit cette boucle se trouve en fin de boucle et non au début. De plus, la condition de cette boucle prend un « ***;*** *» final.*

Question 16

Quelle sera la valeur de la variable nbre après ces boucles ?

```
int i = 0, nbre = 0;
while(i <= 9)
 {
     for (int j = 0; j < 10; j++)
        nbre++;

     i++;
}
System.out.print(nbre);
```

- 10
- 20
- 50
- ✓ 100

*Nous avons ici deux boucles imbriquées l'une dans l'autre. La première doit faire 10 tours (<= 9 donne 10 tours de boucle) et la seconde, idem, 10 tours de boucle (< 10 donne 10 tours de boucle). Nous incrémentons notre variable à chaque tour de boucle, ce qui nous donne 10 tours * 10 tours soit 100 !*

Question 17

Quel indice d'un tableau permet de récupérer son premier élément ?

- ✓ 0
- 1
- 2
- x

Le premier élément d'un tableau est stocké à l'indice 0 !

Question 18

Qu'est-ce qui ne va pas ici :

```
int tableau{} = ['1','2','3','3','3'];
```

- Rien du tout
- ✓ Les crochets sont à utiliser sur la variable et les accolades dans l'initialisation du tableau.
- Des caractères sont mis à la place des entiers !

Les symboles utilisés dans cette instruction ne sont pas au bon endroit. La bonne syntaxe est :

```
int tableau[] = {'1','2','3','3','3'};
```

Vous pouvez mettre des variables de type caractère dans un entier, vous aurez seulement la représentation entière de votre caractère au lieu du caractère lui-même. Donc, ici, vous auriez eu un tableau contenant les valeurs 49, 50, 51, 51, 51 en lieu et place des caractères 1, 2, 3, 3, 3.

Question 19

Cette déclaration est-elle correcte ?

```
int entier [] [] = {{1,2,3,4,5}{1,2,3,4,5}};
```

- Oui
- Non, la variable est mal déclarée : il faut utiliser des double.
- Non, les deux tableaux ont la même taille : c'est interdit !
- ✓ Non, il manque une virgule entre l'initialisation des deux tableaux !

Il manque une virgule entre les deux tableaux : ceci est une erreur de syntaxe qui empêchera votre programme de compiler !

Question 20

Que va afficher ce programme :

```
String tab[][] = {{"toto", "titi", "tutu"}, {"tata", "tete", "tyty"}};

for(String str[] : tab)
{
        for(String str2 : str)
        {
        System.out.println("La valeur est  = " + str2);
        }

}
```

- Rien, il y a une erreur dans le code.
- Seulement le premier tableau
- Seulement le deuxième tableau
- ✓ L'intégralité du tableau bidimensionnel

Il s'agit ici d'une boucle for façon Java 7... Ce code fonctionne parfaitement et affichera l'intégralité du tableau bidimensionnel.

Question 21

Comment se construit une méthode ?

- ✓ (●) Avec une portée, un type de retour, un nom, des paramètres (ou non) et un corps
- ( ) Avec une portée, un nom, des paramètres (ou non) et un corps
- ( ) Avec une portée, un type de retour, des paramètres (ou non) et un corps
- ( ) Avec une portée, un type de retour, des paramètres (ou non)

Une méthode se construit avec au moins une portée, un type de retour, un nom et un corps. Elle peut ne pas avoir de paramètres.

```
public static double calcul(double d){
   return d * 3;
}
```

public : portée.

static : on y reviendra plus tard dans le cours...

double : type de retour.

calcul : le nom de la méthode.

(double d) : les paramètres, mais il peut ne rien y avoir, dans ce cas les parenthèses seront vides.

*{return d*3;} : le corps de la méthode.*

Question 22

Que faut-il faire pour surcharger une méthode ?

- ✓ (●) Modifier le nombre ou le type de ses paramètres.
- ( ) Modifier le type de retour de la nouvelle méthode.
- ( ) Modifier le nom de la méthode.

Pour surcharger une méthode existante, il faut bien entendu que la nouvelle méthode conserve le nom de la méthode initiale, mais elle doit aussi modifier le type ou le nombre de ces paramètres. Modifier le type de retour ne surcharge pas la méthode !

‹ ÉCRIVEZ DES MÉTHODES DE CLASSE

CRÉEZ VOTRE PREMIÈRE CLASSE ›

Le professeur

Cyrille Herby

Spécialiste en développement Java et curieux insatiable d'informatique et de programmation web. Actuellement auditeur en sécurité.

Découvrez aussi ce cours en...

Livre

PDF