SUPNUM — Programmation Orientée Objet

Quiz Java

Bases du langage · Classes et objets

QCM · Vrai / Faux · Analyse de code · Identification d'erreurs · Programmation

QUESTIONS 1 À 14

Partie A — Bases de Java

Paradigme orienté objet, plateforme, syntaxe et premiers programmes

Question 1 — Paradigme orienté objet

QCM

Quelle affirmation décrit correctement la différence entre l'approche procédurale et l'approche orientée objet ?

A) L'approche procédurale se demande « De quoi doit être composé mon programme ? »

B) L'approche orientée objet se demande « Que doit faire mon programme ? »

C) L'approche procédurale se centre sur les actions ; l'OO se centre sur les entités (objets)

D) Les deux approches sont strictement équivalentes en Java

Justifiez brièvement votre choix en une phrase.

Question 2 — Portabilité de Java

Vrai / Faux

Indiquez si chaque affirmation est vraie ou fausse. Justifiez.

1) Un programme Java compilé peut s'exécuter sous Windows, Mac et Linux grâce à la portabilité.

2) Java a été initialement développé par Oracle, puis racheté par Sun Microsystems.

3) Avec Java, on peut réaliser des applications console, fenêtrées, web et mobiles.

4) Un programme Java est uniquement compilé ; il n'y a pas d'étape d'interprétation.

Question 3 — Plateforme Java

QCM

La plateforme Java se compose principalement de :

A) Uniquement du compilateur javac

B) La Java Virtual Machine (JVM) et l'API Java

C) Uniquement de l'IDE (Eclipse ou NetBeans)

D) Le système d'exploitation et le processeur uniquement

Question 4 — Identificateurs Java

QCM

Parmi les identificateurs suivants, lesquels sont valides en Java ? (Plusieurs réponses possibles — indiquez toutes les bonnes réponses.)

A) _total

B) 2emeValeur

C) $montant

D) class

E) vitesseMax

F) une-variable

Rappel : un identificateur commence par une lettre Unicode, $, _ ou µ ; il ne peut pas être un mot-clé.

Question 5 — Règles de nommage

Erreurs

Pour chaque déclaration, indiquez si elle respecte les conventions de nommage Java. Si non, proposez une correction.

```
1) class compteBancaire { ... }
2) package Com.Banque.Util;
3) public static final int TAUX_INTERET = 5;
4) void CalculerSolde() { ... }
5) class Compte_Bancaire { ... }
```

Conventions : Classe en PascalCase ; packages en minuscules ; constantes en MAJUSCULES_AVEC_UNDERSCORE.

Question 6 — Types primitifs

QCM

Quelle affirmation est correcte concernant les types primitifs en Java ?

A) boolean peut prendre les valeurs 0 et 1

B) char occupe 1 octet

C) int occupe 4 octets et long occupe 8 octets

D) float occupe 8 octets et double occupe 4 octets

E) Il n'existe que des types objets ; les primitifs n'existent pas en Java

Question 7 — Wrappers (enveloppeurs)

Vrai / Faux

Indiquez Vrai ou Faux pour chaque affirmation.

1) Les wrappers permettent d'« envelopper » une primitive dans un objet.

2) Integer est le wrapper de int ; Boolean est le wrapper de boolean.

3) Les wrappers n'héritent pas de la classe Object.

4) Les wrappers offrent des méthodes utilitaires (comparaison, min, max, etc.).

Question 8 — Structures de contrôle

Prédiction

Que produit l'exécution du programme suivant ?

```
public class TestBoucle {
    public static void main(String[] args) {
        int s = 0;
        for (int i = 1; i <= 5; i++) {
            if (i % 2 == 0) {
                s = s + i;
            }
        }
        System.out.println(s);
    }
}
```

A) 15

B) 6

C) 9

D) 0

E) Erreur de compilation

Question 9 — Outils de développement

QCM

Associez correctement chaque outil à son rôle (choisissez la correspondance exacte).

A) JDK = exécution seule ; JRE = développement ; IDE = machine virtuelle

B) JRE = exécution (contient la JVM) ; JDK = développement (javac, jar, javadoc) ; IDE = environnement intégré

C) IDE = compilateur seul ; JDK = éditeur de texte ; JRE = documentation

D) JVM = IDE ; JRE = package java.util ; JDK = mot-clé Java

Question 10 — Premier programme

Erreurs

Le fichier Bienvenue.java contient le code ci-dessous. Identifiez les problèmes (compilation / exécution / conventions) et proposez une version corrigée.

```
class bienvenue
{
    public static void Main(String args[])
    {
        System.out.println("Bienvenue dans le monde Java ");
    }
}
```

On compile avec javac Bienvenue.java puis on exécute avec java Bienvenue.

Question 11 — Scanner et affichage

Prédiction

On exécute le programme Addition. L'utilisateur saisit successivement 12 puis 8. Quelle est la sortie exacte affichée ?

```
import java.util.Scanner;
public class Addition {
    public static void main(String args[]) {
        Scanner input = new Scanner(System.in);
        int n1, n2, somme;
        System.out.print(" Donner le premier entier: ");
        n1 = input.nextInt();
        System.out.print(" Donner un deuxième entier: ");
        n2 = input.nextInt();
        somme = n1 + n2;
        System.out.printf("Somme est  %d\n", somme);
    }
}
```

A) Somme est 20

B) Somme est 128

C) 20

D) Erreur : Scanner n'existe pas dans java.util

Question 12 — Lecture et moyenne

Programmation

Écrire une classe publique Moyenne qui :

1) Importe java.util.Scanner
2) Lit trois entiers au clavier
3) Calcule leur moyenne sous forme de double
4) Affiche le résultat avec System.out.println

Respectez les conventions de nommage et la signature classique de main.

Question 13 — Classe, attributs et méthodes

QCM

Quelle affirmation est exacte concernant les classes en Java ?

A) Les attributs représentent les actions ; les méthodes représentent les données

B) Les attributs décrivent les données propres à une classe ; les méthodes décrivent les actions

C) Une classe Java ne peut contenir que des méthodes, jamais d'attributs

D) Attributs et méthodes sont des synonymes de « mot-clé »

Question 14 — do-while et condition

Prédiction

Que produit l'exécution du programme suivant ?

```
public class TestDo {
    public static void main(String[] args) {
        int n = 1;
        do {
            System.out.print(n + " ");
            n++;
        } while (n < 4);
        System.out.println("fin=" + n);
    }
}
```

A) 1 2 3 fin=4

B) 1 2 3 4 fin=4

C) fin=1

D) Erreur de compilation (do-while n'existe pas en Java)

QUESTIONS 15 À 32

Partie B — Classes et objets

Encapsulation, références, packages, méthodes, constructeurs, static et final

Question 15 — Classe et objet

QCM

Parmi les affirmations suivantes, laquelle est exacte ?

A) Une classe est une instance d'un objet

B) Un objet est une description abstraite ; la classe est l'instance en mémoire

C) Une classe décrit un ensemble d'objets partageant attributs et méthodes ; un objet est une instance

D) Classe et objet sont strictement synonymes en Java

Question 16 — Encapsulation

QCM

Pourquoi utilise-t-on des attributs privés avec getters et setters dans une classe comme Voiture ?

A) Pour accélérer l'exécution du programme

B) Pour protéger les données (encapsulation) et contrôler lecture / modification

C) Parce que Java interdit les attributs publics

D) Uniquement pour pouvoir utiliser le mot-clé static

Question 17 — Création d'objets et null

Erreurs

Que se passe-t-il avec le code suivant ? Corrigez-le.

```
public static void main(String args[]) {
    Voiture v;
    v.setMarque("Tesla");
}
```

Précisez : erreur de compilation ou d'exécution ? Pourquoi ?

Proposez une correction en utilisant new et, si besoin, un test != null.

Question 18 — Références et affectation

Prédiction

Après exécution du fragment suivant, que valent les vitesses affichées ?

```
Voiture v1 = new Voiture();
Voiture v2 = v1;
v1.setMarque("Toyota");
v1.setVitesse(80);
v2.setVitesse(120);
System.out.println(v1.getVitesse());
System.out.println(v2.getVitesse());
```

A) 80 puis 120

B) 120 puis 120

C) 80 puis 80

D) Erreur de compilation

Rappel : Voiture v2 = v1 ; les deux variables référencent le même objet.

Question 19 — Deux objets distincts

Prédiction

Comparez avec la question précédente. Que produit maintenant ce code ?

```
Voiture v1 = new Voiture();
Voiture v2 = new Voiture();
v1.setMarque("Toyota"); v1.setVitesse(80);
v2.setMarque("Peugeot"); v2.setVitesse(60);
v2.setVitesse(100);
System.out.println(v1.getMarque() + " " + v1.getVitesse());
System.out.println(v2.getMarque() + " " + v2.getVitesse());
System.out.println(v1 == v2);
```

A) Toyota 80 / Peugeot 100 / true

B) Toyota 100 / Peugeot 100 / false

C) Toyota 80 / Peugeot 100 / false

D) Toyota 80 / Peugeot 60 / false

Chaque new crée un objet distinct en mémoire.

Question 20 — Comparaison d'objets

QCM

Concernant la comparaison d'objets en Java, quelle affirmation est correcte ?

A) L'opérateur == compare toujours le contenu des attributs

B) equals() compare toujours le contenu, même sans redéfinition

C) == compare les références ; equals() compare le contenu si elle est redéfinie

D) clone() sert uniquement à comparer deux objets

Question 21 — Mot-clé this

Prédiction

Que se passe-t-il dans le setter suivant ? Quelle est la sortie ?

```
public void setMarque(String marque) {
    this.marque = marque;
}
// dans main :
Voiture v = new Voiture();
v.setMarque("BMW");
System.out.println(v.getMarque());
```

A) Affiche null — this.marque et marque désignent la même variable locale

B) Affiche BMW — this.marque désigne l'attribut d'instance ; marque est le paramètre

C) Erreur de compilation : this est interdit dans un setter

D) Affiche marque (le mot littéral)

Question 22 — Mot-clé this et packages

Vrai / Faux

Indiquez Vrai ou Faux.

1) this désigne l'objet courant (l'instance qui exécute la méthode).

2) La déclaration package doit être la première instruction du fichier source.

3) import java.util.*; importe toutes les classes de java.util.

4) La structure des dossiers doit correspondre à la hiérarchie des packages.

5) On peut qualifier une classe top-level avec protected.

Question 23 — Modificateurs d'accès

QCM

Quel modificateur offre le niveau de protection le plus fort pour un attribut ?

A) public

B) protected

C) package-private (aucun modificateur)

D) private

Rappel : private = visible uniquement à l'intérieur de la classe.

Question 24 — Surcharge de méthodes

Prédiction

Soit une classe Calculatrice avec additionner surchargée (int/int, double/double, int/int/int). Que renvoie chaque appel ?

```
Calculatrice c = new Calculatrice();
System.out.println(c.additionner(2, 3));        // (1)
System.out.println(c.additionner(2.5, 3.5));    // (2)
System.out.println(c.additionner(1, 2, 3));     // (3)
```

Indiquez le type et la valeur retournés pour (1), (2) et (3).

Quel type de polymorphisme associe-t-on à la surcharge de méthodes ?

Question 25 — Ambiguïté de surcharge

Erreurs

Expliquez pourquoi l'appel suivant provoque une erreur. Proposez une correction en changeant uniquement l'appel (pas les signatures).

```
public class TestAmbigu {
    void afficher(int a, double b) { System.out.println("int - double"); }
    void afficher(double a, int b) { System.out.println("double - int"); }
    public static void main(String[] args) {
        TestAmbigu t = new TestAmbigu();
        t.afficher(5, 5); // Erreur : appel ambigu !
    }
}
```

Piste : conversions implicites byte → short → int → long → float → double.

Question 26 — Ambiguïté (classe Point)

Erreurs

Avec les signatures suivantes, quels appels compilent ? Lequel est ambigu ?

```
public void deplace(int dx, byte dy) { x += dx; y += dy; }
public void deplace(byte dx, int dy) { x += dx; y += dy; }
Point a = new Point();
int n; byte b;
a.deplace(n, b); // (1)
a.deplace(b, n); // (2)
a.deplace(b, b); // (3)
```

Pour (1), (2) et (3) : compilable ou ambigu ? Justifiez avec la compatibilité des types.

Question 27 — Constructeurs

QCM

Parmi les affirmations suivantes sur les constructeurs, laquelle est fausse ?

A) Un constructeur porte le même nom que la classe

B) Un constructeur n'a pas de type de retour (pas même void)

C) Il est automatiquement appelé lors de la création d'une instance avec new

D) Si aucun constructeur n'est défini, Java refuse de compiler la classe

Question 28 — Surcharge de constructeurs

Prédiction

Compte tenu des constructeurs suivants, indiquez les valeurs de marque et couleur pour chaque objet créé.

```
public Voiture() { marque = "Inconnue"; couleur = "Blanche"; }
public Voiture(String marque) { this.marque = marque; couleur = "Noire"; }
public Voiture(String marque, String couleur) {
    this.marque = marque; this.couleur = couleur;
}

Voiture a = new Voiture();
Voiture b = new Voiture("Toyota");
Voiture c = new Voiture("Tesla", "Grise");
Voiture d = new Voiture(c);  // constructeur par copie
```

Complétez : a(?, ?) ; b(?, ?) ; c(?, ?) ; d(?, ?)

d et c ont-ils la même référence mémoire ? Pourquoi ?

Question 29 — Membres static

Prédiction

Que produit l'exécution du code suivant ?

```
Voiture v1 = new Voiture("Toyota", "Bleue");
Voiture v2 = new Voiture("Tesla", "Grise");
Voiture.afficherNombreVoitures();
```

On suppose : static int nombreVoitures = 0 ; incrémenté dans le constructeur,

et afficherNombreVoitures() affiche : "Nombre total de voitures : " + nombreVoitures

Rappel : une variable static est partagée par tous les objets de la classe.

Question 30 — Attribut final

Erreurs

Pour chaque extrait, dites s'il compile. Sinon, expliquez pourquoi.

```
// (1)
private final int n = 20;
n = 30;   // dans une méthode

// (2)
class A { private final int n; public A() { n = 10; } }

// (3)
class B { private final int n; }

// (4)
class C { private final int n; public C(int nn) { n = nn; } }
```

Question 31 — Classe CompteBancaire

Programmation

Écrire une classe CompteBancaire avec encapsulation, constructeurs et méthodes.

- Attributs privés : titulaire (String), solde (double)
- Constructeur paramétré (titulaire, soldeInitial)
- Constructeur par copie
- Getters / setters pour titulaire ; getter seul pour solde
- Méthodes : deposer(double montant), retirer(double montant) — retirer seulement si solde suffisant
- Méthode afficher() affichant titulaire et solde
- Attribut static nombreComptes incrémenté à chaque création ; méthode static afficherNombreComptes()

Pas besoin d'héritage. Utilisez this dans les constructeurs / setters.

Question 32 — equals et scénario complet

Programmation

À partir d'une classe Voiture (marque, couleur, …) :

1) Rappeler pourquoi equals() de Object ne suffit pas pour comparer le contenu
2) Écrire une redéfinition cohérente de equals
3) Dans un main : créer v1, cloner le contenu via constructeur par copie dans v2,
   puis prédire les résultats de (v1 == v2) et v1.equals(v2)
4) Expliquer la différence avec : Voiture v3 = v1 ; (v1 == v3)

CORRECTION

Corrigé

Réponses et explications

Corrigé — Questions 1 à 4

Question 1 — QCM

Réponse : C. L'approche procédurale se demande « Que doit faire mon programme ? » ; l'approche OO se demande « De quoi doit être composé mon programme ? » (entités / objets).

Question 2 — Vrai / Faux

1) Vrai — portabilité via la JVM.
2) Faux — développé par Sun Microsystems, racheté ensuite par Oracle.
3) Vrai — console, fenêtres, web, mobile.
4) Faux — un programme Java est compilé puis interprété (bytecode → JVM).

Question 3 — QCM

Réponse : B. La plateforme Java = JVM + API Java.

Question 4 — Identificateurs

Valides : A (_total), C ($montant), E (vitesseMax).
Invalides : B (commence par un chiffre), D (mot-clé class), F (tiret « - » interdit).

Corrigé — Questions 5 à 8

Question 5 — Nommage

1) Non → CompteBancaire (1re lettre majuscule, CamelCase).
2) Non → com.banque.util (tout en minuscules).
3) Oui — constante en MAJUSCULES_AVEC_UNDERSCORE.
4) Non (convention méthodes) → calculerSolde (minuscule initiale).
5) Acceptable syntaxiquement, mais convention usuelle : CompteBancaire sans underscore.

Question 6 — Types primitifs

Réponse : C. int = 4 octets ; long = 8 octets.
Compléments : boolean = true/false (pas 0/1) ; char = 2 octets ; float = 4, double = 8.

Question 7 — Wrappers

1) Vrai 2) Vrai 3) Faux (ils héritent de Object) 4) Vrai.

Question 8 — Boucle

Réponse : B) 6. Les entiers pairs entre 1 et 5 sont 2 et 4 ; s = 0+2+4 = 6.

Corrigé — Questions 9 à 14

Question 9 — Outils

Réponse : B. JRE = exécution (JVM) ; JDK = développement (javac, jar, javadoc…) ; IDE = environnement intégré (Eclipse, NetBeans…).

Question 10 — Premier programme

Problèmes : (1) nom de classe bienvenue ≠ convention / fichier ; (2) Main doit être main (minuscules).
Correction : class Bienvenue { public static void main(String args[]) { … } }
Compilation : javac Bienvenue.java → Exécution : java Bienvenue

Question 11 — Scanner

Réponse : A) Somme est 20 (printf avec %d et n1+n2 = 20). Scanner est dans java.util depuis 1.5.

Question 12 — Moyenne

Solution type : lire 3 entiers avec Scanner, moyenne = (a+b+c)/3.0 , puis println.

Question 13 — Attributs / méthodes

Réponse : B. Attributs = données ; méthodes = actions / comportements de la classe.

Question 14 — do-while

Réponse : A) 1 2 3 fin=4. Le corps s'exécute tant que n < 4 ; après la boucle n vaut 4.

Corrigé — Questions 15 à 19

Question 15 — Classe / objet

Réponse : C. La classe est l'abstraction ; l'objet est une instance en mémoire.

Question 16 — Encapsulation

Réponse : B. Attributs privés + getters/setters pour protéger et contrôler l'accès.

Question 17 — null

Erreur de compilation : v non initialisée.
Correction : Voiture v = new Voiture(); v.setMarque("Tesla");
Bonne pratique : if (v != null) avant un appel de méthode.

Question 18 — Références partagées

Réponse : B) 120 puis 120. v2 = v1 copie la référence : un seul objet.

Question 19 — Deux new

Réponse : C) Toyota 80 / Peugeot 100 / false. Deux objets distincts ; == compare les références.

Corrigé — Questions 20 à 25

Question 20 — Comparaison

Réponse : C. == → références ; equals() → contenu si redéfinie (sinon comme ==).

Question 21 — this

Réponse : B) BMW. this.marque = attribut ; marque = paramètre. this lève l'ambiguïté de nommage.

Question 22 — this / packages

1) Vrai 2) Vrai 3) Vrai 4) Vrai 5) Faux — pas de classe top-level protected.

Question 23 — Visibilité

Réponse : D) private — visible uniquement dans la classe.

Question 24 — Surcharge

(1) int → 5 (2) double → 6.0 (3) int → 6. Polymorphisme statique.

Question 25 — Ambiguïté afficher

Les deux signatures sont acceptables pour (5,5) → erreur.
Correction : t.afficher(5, 5.0); ou t.afficher(5.0, 5);

Corrigé — Questions 26 à 30

Question 26 — Point.deplace

(1) OK → deplace(int, byte)
(2) OK → deplace(byte, int)
(3) Ambigu : byte b est compatible avec int et avec byte pour les deux paramètres.

Question 27 — Constructeurs

Affirmation fausse : D. Sans constructeur explicite, Java fournit un constructeur par défaut.

Question 28 — Constructeurs surchargés

a : Inconnue/Blanche ; b : Toyota/Noire ; c : Tesla/Grise ; d : Tesla/Grise.
d et c : références différentes, contenus identiques (copie).

Question 29 — static

Affichage : Nombre total de voitures : 2 (compteur partagé incrémenté à chaque new).

Question 30 — final

(1) ✗ réaffectation interdite (2) ✓ init constructeur (3) ✗ non initialisé (4) ✓ valeur par objet.

Corrigé — Question 31 (CompteBancaire) · 1/2

```
public class CompteBancaire {
    private String titulaire;
    private double solde;
    private static int nombreComptes = 0;

    public CompteBancaire(String titulaire, double soldeInitial) {
        this.titulaire = titulaire;
        this.solde = soldeInitial;
        nombreComptes++;
    }

    public CompteBancaire(CompteBancaire autre) {
        this.titulaire = autre.titulaire;
        this.solde = autre.solde;
        nombreComptes++;
    }
```

Corrigé — Question 31 (CompteBancaire) · 2/2

```
    public String getTitulaire() { return titulaire; }
    public void setTitulaire(String titulaire) { this.titulaire = titulaire; }
    public double getSolde() { return solde; }

    public void deposer(double montant) {
        if (montant > 0) solde += montant;
    }
    public void retirer(double montant) {
        if (montant > 0 && montant <= solde) solde -= montant;
    }
    public void afficher() {
        System.out.println("Titulaire : " + titulaire + ", Solde : " + solde);
    }
    public static void afficherNombreComptes() {
        System.out.println("Nombre total de comptes : " + nombreComptes);
    }
}
```

Corrigé — Question 32 (equals et références)

1) equals() héritée de Object compare les références (comme ==). Il faut la redéfinir pour comparer les attributs.
2) Exemple de redéfinition :

```
@Override
public boolean equals(Object obj) {
    if (obj == null) return false;
    if (!(obj instanceof Voiture)) return false;
    Voiture autre = (Voiture) obj;
    return marque.equals(autre.marque) && couleur.equals(autre.couleur);
}
```

3) v2 = new Voiture(v1) → objets distincts, même contenu :

v1 == v2 → false (références différentes)

v1.equals(v2) → true (si equals redéfinie correctement)

4) Voiture v3 = v1 ; → v1 == v3 est true (même référence mémoire).

Distinction essentielle : affectation de référence ≠ copie d'objet ≠ comparaison de contenu.

Aide à la correction rapide

Grille récapitulative des réponses

Question	Réponse attendue (résumé) — se référer aux diapositives détaillées pour les justifications						
Q1	C	Q9	B	Q17	new requis	Q25	cast args
Q2	V F V F	Q10	classe+main	Q18	B 120/120	Q26	1✓ 2✓ 3✗
Q3	B	Q11	A	Q19	C	Q27	D fausse
Q4	A,C,E	Q12	exercice	Q20	C	Q28	voir corrigé
Q5	conventions	Q13	B	Q21	B BMW	Q29	… : 2
Q6	C	Q14	A	Q22	V V V V F	Q30	✗✓✗✓
Q7	V V F V	Q15	C	Q23	D	Q31	exercice
Q8	B (6)	Q16	B	Q24	5/6.0/6	Q32	exercice

Fin du quiz

Merci pour votre attention

Bases du langage · Classes et objets