Règles examen Java

Voici une **liste de questions de cours** (théoriques et pratiques) possibles pour ton examen, basées sur les thèmes du TP et le programme habituel en Java orienté objet :

🎯 1. Polymorphisme

1. Qu'est-ce que le polymorphisme en Java ? Donnez un exemple concret.
2. Quelle est la différence entre le polymorphisme statique et dynamique ?
3. Pourquoi utiliser une interface dans un tableau polymorphe ?
4. Que se passe-t-il si on appelle une méthode non redéfinie dans une classe fille ?

📚 2. Tableaux et Collections

5. Quelle est la différence entre un tableau (`Animal[]`) et une collection (`ArrayList<Animal>`) ?
6. Donnez un exemple de parcours d'un tableau d'objets en Java.
7. Peut-on stocker des objets de différentes classes dans un même tableau ? Pourquoi ?

🧩 3. Classes Abstraites

8. Quelle est la différence entre une classe abstraite et une interface ?
9. Peut-on instancier une classe abstraite ? Justifiez votre réponse.
10. Peut-on définir un constructeur dans une classe abstraite ? À quoi sert-il ?

📦 4. Interfaces et Packages

11. À quoi sert une interface en Java ?
12. Peut-on avoir des attributs dans une interface ? De quel type ?
13. Comment organiser les classes dans des packages ? Donnez un exemple.
14. Quelle est la syntaxe pour importer une classe d'un autre package ?

🚨 5. Gestion des Exceptions

15. Quelle est la différence entre une exception **checked** et **unchecked** ?
16. Comment crée-t-on une exception personnalisée ?
17. À quoi sert un bloc `try-catch` ? Donnez un exemple.
18. Que signifie le mot-clé `throw` ? Et `throws` ?
19. Que se passe-t-il si une exception n'est pas capturée ?
20. Peut-on avoir plusieurs blocs `catch` pour un seul `try` ? Dans quel ordre sont-ils traités ?

⚙️ 6. Mini-code à analyser / compléter

21. Complétez cette interface :

```java
interface Vehicule {
    _______ void demarrer();
    String type();
}
```

22. Que produit ce code ?

```java
```

```
Animal a = new Chien();
a.crier();
```

23. Quel est le problème dans ce code ? Comment le corriger ?

```java
Animal a = new Animal("Léo", -3);
```

24. Écrire une classe `Lion` qui implémente `Animal` (interface) avec :

- `crier()` → "Rugir"
- `regimeAlimentaire()` → "Carnivore"

Souhaites-tu que je te fasse une fiche de QCM ou des questions à choix multiples pour t'entraîner ?

Voici les **réponses aux questions de cours** que je t'ai proposées — tu pourras les utiliser pour réviser efficacement.

🎯 1. Polymorphisme

1. **Qu'est-ce que le polymorphisme en Java ?**

   > Le polymorphisme permet d'utiliser une même interface ou classe parent pour manipuler des objets de différentes classes filles. Exemple :

   ```java
   Animal a = new Chien(); // Animal est l'interface
   a.crier(); // Appelle la version de Chien
   ```

2. **Différence entre polymorphisme statique et dynamique ?**

   - *Statique* : surcharge des méthodes (même nom, paramètres différents).

- *Dynamique* : redéfinition des méthodes (héritage, méthode override).

3. **Pourquoi utiliser une interface dans un tableau polymorphe ?**

   > Pour manipuler plusieurs objets différents (Chien, Chat, Vache) via une seule référence commune (`Animal[]`).

4. **Appel d'une méthode non redéfinie dans une classe fille ?**

   > La méthode héritée de la classe mère (ou interface par défaut) est utilisée.

📚 2. Tableaux et Collections

5. **Différence entre tableau et collection ?**
   - *Tableau* : taille fixe, syntaxe simple, rapide.
   - *Collection* (ex : `ArrayList`) : taille dynamique, plus flexible.

6. **Exemple de parcours de tableau :**

```java
for (Animal a : animaux) {
    a.crier();
}
```

7. **Objets différents dans un même tableau ?**

   > Oui, s'ils héritent ou implémentent la même classe ou interface.

🧩 3. Classes Abstraites

8. **Différence classe abstraite vs interface :**
   - Classe abstraite : peut avoir des attributs, méthodes concrètes + abstraites.
   - Interface : 100% abstraction (avant Java 8), pas de constructeurs.

9. **Instancier une classe abstraite ?**

> Non, car elle est incomplète (elle contient des méthodes sans corps).

10. **Constructeur dans une classe abstraite ?**

> Oui, utilisé pour initialiser les attributs hérités.

📦 4. Interfaces et Packages

11. **À quoi sert une interface ?**

> Définir un contrat que les classes doivent respecter (méthodes obligatoires).

12. **Attributs dans une interface ?**

> Oui, mais ils sont automatiquement `public static final`.

13. **Organisation en packages ?**

> Crée des dossiers pour regrouper les classes liées :

```java
package interfaces.animaux;
```

14. **Importer une classe d'un autre package :**

```java
import interfaces.animaux.Chien;
```

🚨 5. Gestion des Exceptions

15. **Exception checked vs unchecked ?**

- *Checked* : doivent être déclarées ou capturées (ex : IOException).
- *Unchecked* : erreurs à l'exécution (ex : NullPointerException).

16. **Créer une exception personnalisée :**

```java
public class ErrAge extends Exception {
    public ErrAge(String message) {
        super(message);
    }
}
```

17. **Bloc try-catch :**

> Capture et gère les exceptions :

```java
try { ... } catch (Exception e) { ... }
```

18. **Différence `throw` vs `throws` :**

- `throw` : pour lancer une exception.
- `throws` : pour déclarer qu'une méthode peut lancer une exception.

19. **Exception non capturée ?**

> Programme plante avec une trace d'erreur (stack trace).

20. **Plusieurs catchs ?**

> Oui, traités de haut en bas. Les plus spécifiques doivent être en premier.

⚙️ 6. Questions de code

21. **Compléter l'interface `Vehicule` :**

```java
public interface Vehicule {
    void demarrer();
    String type();
}
```

22. **Ce code produit :**

> `"Aboyer"` si `Chien` redéfinit `crier()` pour afficher cela.

23. **Problème du code :**

> Âge négatif → une exception `ErrAge` doit être levée.

24. **Classe Lion :**

```java
public class Lion implements Animal {
    public void crier() {
        System.out.println("Rugir");
    }

    public String regimeAlimentaire() {
        return "Carnivore";
    }
}
```

Tu veux aussi un **QCM** ou des **exercices interactifs** pour t'entraîner ?