Examen Java Résumé

Voici un résumé des points clés que tu devrais connaître pour chaque sujet :

1. Introduction à Java

- **Syntaxe de base** : Comprendre la structure d'un programme Java (classes, méthodes `main`, blocs de code).
- **Types de données primitifs** : int, double, boolean, char, etc.
- **Opérateurs** : arithmétiques (+, -, *, /), logiques (&&, ||), de comparaison (==, !=, >, <), etc.
- **Contrôles de flux** : structures conditionnelles (`if` , `else` , `switch`) et boucles (`for` , `while` , `do-while`).
- **Exceptions** : `try` , `catch` , `finally` , gestion des erreurs avec les exceptions (`throw` , `throws`).

2. Classes et Objets

- **Définition d'une classe** : Une classe contient des attributs (variables d'instance) et des méthodes.
- **Création d'objets** : Utiliser le mot-clé `new` pour instancier un objet à partir d'une classe.
- **Encapsulation** : Utilisation des modificateurs d'accès (`private` , `public` , `protected`), et des méthodes d'accès (getters et setters).
- **Méthodes et attributs d'instance** : Les méthodes sont des actions qu'un objet peut effectuer. Les attributs d'instance stockent l'état de l'objet.

3. Constructeurs

- **Définition** : Un constructeur est une méthode spéciale qui initialise un objet lors de sa création.
- **Constructeur par défaut** : Si tu ne définis pas de constructeur, Java crée un constructeur par défaut (sans paramètres).
- **Constructeur paramétré** : Tu peux créer des constructeurs avec des paramètres pour initialiser des attributs spécifiques lors de la création de l'objet.
- **Surcharge de constructeur** : Avoir plusieurs constructeurs dans une classe avec des paramètres différents (la signature du constructeur doit être différente).

4. Héritage

- **Concept d'héritage** : Une classe peut hériter des attributs et méthodes d'une autre classe via le mot-clé `extends`.
- **Classe parente et classe enfant** : La classe enfant hérite de la classe parente et peut ajouter ou modifier des comportements.
- **Méthode `super`** : Utilisée pour appeler un constructeur ou une méthode de la classe parente.
- **Redéfinition de méthodes (Override)** : Dans la classe enfant, tu peux redéfinir une méthode de la classe parente pour la personnaliser.
- **Polymorphisme** : Capacité d'un objet à prendre plusieurs formes. Par exemple, une méthode dans la classe enfant peut remplacer la méthode de la classe parente tout en ayant le même nom.

Si tu veux plus de détails sur l'un de ces points, n'hésite pas à me le dire !

Voici des exemples pratiques pour chaque sujet :

1. Introduction à Java

Exemple de programme simple :

```java
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Bonjour tout le monde !");
    }
}
```

Ce programme affiche "Bonjour tout le monde !" à l'écran. C'est le programme de base pour tester si l'environnement Java est bien installé.

---

2. Classes et Objets

Exemple d'une classe simple :

java

```java
public class Voiture {
    // Attributs (variables d'instance)
    String marque;
    int annee;

    // Méthode (fonction)
    void conduire() {
        System.out.println("La voiture " + marque + " roule !");
    }
}

public class Main {
    public static void main(String[] args) {
        // Création d'un objet de la classe Voiture
        Voiture maVoiture = new Voiture();
        maVoiture.marque = "Toyota";
        maVoiture.annee = 2020;
        maVoiture.conduire();  // Appel de la méthode
    }
}
```

Ce code crée un objet `maVoiture` de la classe `Voiture` et lui donne des valeurs pour ses attributs. Ensuite, on appelle la méthode `conduire()` pour afficher un message.

---

3. Constructeurs

Exemple avec constructeur par défaut et constructeur paramétré :

java

```java
public class Personne {
    String nom;
    int age;

    // Constructeur par défaut
    public Personne() {
        nom = "Inconnu";
        age = 0;
```

```
    }

    // Constructeur paramétré
    public Personne(String nom, int age) {
        this.nom = nom;
        this.age = age;
    }

    void afficherInfo() {
        System.out.println("Nom : " + nom + ", Age : " + age);
    }

    public static void main(String[] args) {
        // Utilisation du constructeur par défaut
        Personne p1 = new Personne();
        p1.afficherInfo();  // Affiche : Nom : Inconnu, Age : 0

        // Utilisation du constructeur paramétré
        Personne p2 = new Personne("Alice", 25);
        p2.afficherInfo();  // Affiche : Nom : Alice, Age : 25
    }
}
```

Ici, il y a deux constructeurs : un par défaut qui initialise les attributs à des valeurs par défaut, et un constructeur paramétré pour initialiser les attributs avec des valeurs spécifiques.

---

4. Héritage

Exemple d'héritage avec classe parente et classe enfant :

```java
// Classe parente
class Animal {
    String nom;

    public void manger() {
        System.out.println(nom + " mange.");
```

```
    }
}

// Classe enfant qui hérite de la classe Animal
class Chien extends Animal {
    public void aboyer() {
        System.out.println(nom + " aboie !");
    }
}

public class Main {
    public static void main(String[] args) {
        // Création d'un objet Chien
        Chien monChien = new Chien();
        monChien.nom = "Rex";
        monChien.manger();  // Méthode héritée de Animal
        monChien.aboyer();  // Méthode spécifique à Chien
    }
}
```

Dans cet exemple, la classe `Chien` hérite de la classe `Animal`. Elle peut donc utiliser la méthode `manger()`, et en plus, elle a sa propre méthode `aboyer()`.

Exemple de redéfinition de méthode (`Override`) :

java

```
class Animal {
    public void faireDuBruit() {
        System.out.println("L'animal fait du bruit.");
    }
}

class Chat extends Animal {
    @Override
    public void faireDuBruit() {
        System.out.println("Le chat miaule.");
    }
}

public class Main {
    public static void main(String[] args) {
        Animal monChat = new Chat();
```

```
        monChat.faireDuBruit();  // Affiche : Le chat miaule.
    }
}
```

Dans cet exemple, la méthode `faireDuBruit()` est redéfinie dans la classe `Chat`, et le programme utilise la méthode redéfinie même si l'objet est référencé par une variable de type `Animal`.

Ces exemples couvrent les bases de Java, les classes, les objets, les constructeurs et l'héritage. Si tu souhaites plus de détails ou des exemples supplémentaires, fais-le moi savoir !