Règles examen Java

Voici un résumé bien organisé des règles à suivre pour ton examen basé sur le document **TP_Generals.pdf**. Ce résumé couvre les points clés suivants : **Polymorphisme, Tableaux et collections, Classes abstraites, Interfaces et packages, Gestion des exceptions**.

✅ 1. Polymorphisme

- Le polymorphisme permet d'utiliser des références de type interface ou classe mère pour manipuler des objets de classes dérivées.
- Exemple :

```java
Animal[] animaux = new Animal[3];
animaux[0] = new Chien();
animaux[1] = new Chat();
animaux[2] = new Vache();
for (Animal a : animaux) {
    System.out.println(a.crier() + " - " + a.regimeAlimentaire());
}
```

✅ 2. Tableaux et Collections

- Utiliser un **tableau d'objets** permet de stocker plusieurs instances de différentes classes implémentant une même interface.
- Exemple :

```java
Animal[] animaux = { new Chien(), new Chat(), new Vache() };
for (Animal a : animaux) {
    a.crier();
```

```
    System.out.println(a.regimeAlimentaire());
}
```

✅ 3. Classes Abstraites

Bien que non directement mentionné dans le PDF, à titre de règle :

- Une classe abstraite peut contenir des méthodes abstraites (sans corps) et des méthodes concrètes.
- Elle ne peut pas être instanciée.
- Exemple générique :

```java
abstract class Animal {
    String nom;
    int age;
    abstract void crier();
}
```

✅ 4. Interfaces et Packages

- Une **interface** contient uniquement des méthodes sans corps (par défaut).
- Implémentée par les classes : elles doivent redéfinir toutes les méthodes.
- Organisation en **packages** pour mieux structurer le projet :

```markdown
interfaces.animaux
    ↳ Animal.java
interfaces.animaux.mammiferes
    ↳ Chien.java, Chat.java
interfaces.animaux.herbivores
    ↳ Vache.java
```

- Exemple d'interface :

```java
public interface Animal {
    void crier();
    String regimeAlimentaire();
}
```

✅ 5. Gestion des Exceptions

- **Exception personnalisée** : créer ses propres classes d'exception en héritant de `Exception`.
  - `ErrAge` : si âge < 0
  - `ErrVieillissement` : si âge > 150
- Utilisation de `try-catch` pour capturer les erreurs :

```java
try {
    Animal a = new Animal("Léo", -5); // ErrAge
} catch (ErrAge e) {
    System.out.println(e.getMessage());
}
```

- Exemple de méthode `vieillir()` :

```java
public void vieillir(int annees) throws ErrVieillissement {
    if (this.age + annees > 150)
        throw new ErrVieillissement("Âge trop élevé !");
    this.age += annees;
}
```

Si tu veux, je peux aussi te faire une **fiche récap visuelle** ou **carte mentale** pour tout ça. Tu veux ?