Les classes abstraites :

Les classes abstraites en Java sont utilisées pour définir des modèles communs qui peuvent être étendus par d'autres classes. Ce qu'il faut retenir sur les classes abstraites :

1- Une classe abstraite ne peut pas être instanciée directement. Cela signifie que vous ne pouvez pas créer d'objets à partir d'une classe abstraite. Elle sert principalement de modèle pour d'autres classes qui en étendent les fonctionnalités.
2- Une classe abstraite peut contenir des méthodes abstraites, qui sont des méthodes déclarées sans implémentation. Ces méthodes doivent être implémentées par les sous-classes. En revanche, une classe abstraite peut également contenir des méthodes concrètes (avec implémentation) qui sont héritées par les sous-classes.
3- Les méthodes abstraites doivent être déclarées avec le mot-clé abstract, tandis que les méthodes concrètes peuvent être définies normalement. Les méthodes abstraites ne peuvent exister que dans les classes abstraites.
4- Une classe peut étendre une seule classe abstraite. Cela signifie qu'une classe qui étend une classe abstraite doit fournir une implémentation pour toutes les méthodes abstraites de la classe abstraite parente, à moins que la sous-classe ne soit elle-même déclarée comme abstraite.
5- Les classes abstraites peuvent contenir des variables, des méthodes et d'autres membres de classe comme n'importe quelle autre classe Java. Cela signifie que les classes abstraites peuvent avoir des champs de données et des méthodes concrètes en plus des méthodes abstraites.
6- Les classes abstraites sont souvent utilisées pour définir un comportement de base commun pour un groupe de sous-classes, tout en laissant certaines parties du comportement à être implémentées par les sous-classes spécifiques.

Exemple :

```java
abstract class Figure {
    // Méthode abstraite
    abstract void dessiner();

    // Méthode concrète
    void redimensionner() {
        // Traitement: implémentation (corps) de la méthode
    }
}
```

Les interfaces :

Une interface en Java est une collection de méthodes abstraites et de constantes qui peuvent être utilisées comme un type de données référençable par d'autres classes. Une interface est déclarée à l'aide du mot-clé **interface** et définit les méthodes que toute classe implémentant cette interface doit fournir. Ce qu'il faut retenir des interfaces :

1- Une interface en Java est une collection de méthodes abstraites et de constantes (qui sont implicitement statiques et finales). Elle ne peut contenir que des signatures de méthodes et des constantes, et ces méthodes n'ont pas d'implémentation dans l'interface elle-même.

2- Toutes les méthodes déclarées dans une interface sont implicitement publiques et abstraites. Cela signifie que les classes qui implémentent une interface doivent fournir une implémentation concrète de toutes les méthodes déclarées dans l'interface.
3- Une classe peut implémenter plusieurs interfaces en même temps, permettant ainsi une certaine flexibilité dans la définition du comportement.
4- Les interfaces sont largement utilisées pour définir des contrats communs entre les classes sans se soucier de l'implémentation concrète de ces méthodes. Cela favorise la modularité, l'extensibilité et la réutilisabilité du code.
5- Les interfaces peuvent être utilisées pour réaliser une sorte de polymorphisme en Java, ce qui signifie qu'une référence de type d'interface peut être utilisée pour faire référence à des instances de classes qui implémentent cette interface.
6- Les constantes définies dans l'interface sont implicitement **public, static et final.**
7- Les méthodes déclarées dans l'interface sont par défaut **public et abstract**, elles n'ont pas d'implémentation dans l'interface elle-même.

Exemple :

```java
public interface MonInterface {
    // Constantes
    int CONSTANTE = 5;

    // Méthodes (sans implémentation)
    void methode1();
    int methode2(String param);
}
```