Java

Encapsulation

Définition :

Principe qui consiste à protéger les données internes d'un classe en les rendant accessibles uniquement via des méthodes publiques (gettes/setters).

Code :

```
public class User {
    private String name; // caché
    private int age;

    public String getName() { return name; }       // accès contrôlé
    public void setName(String name) { this.name = name; }

    public int getAge() { return age; }
    public void setAge(int age) {
        if (age >= 0) this.age = age; // validation ici
    }
}
```

Polymorphism

Définition :

Capacité d’un même objet à se comporter différemment selon son type concret. On peut manipuler plusieurs objets différents via une même référence parent.

Code :

```
Animal a1 = new Dog();
Animal a2 = new Cat();

List<Animal> animals = List.of(a1, a2);
for (Animal a : animals) {
```

```
    a.speak(); // chaque animal a son comportement propre
}
```

Héritage

Définition :

Mécanisme permettant à une classe d'hériter des attributs et comportements d'une autre classe. La classe dérivée peut réutiliser ou redéfinir (`@Override`) ces comportements.

Code :

```
public class Animal {
    void speak() { System.out.println("Some sound"); }
}

public class Dog extends Animal {
    @Override
    void speak() { System.out.println("Woof!"); } // spécialisation
}

Animal a = new Dog();
a.speak(); // → Woof!
```

Interface

Définition :

Contrat qui définit un ensemble de méthodes qu’une classe doit implémenter. Ne contient pas d’état (sauf constantes).

Code :

```
public interface Flyable {
    void fly(); // méthode abstraite
}

public class Bird implements Flyable {
    public void fly() { System.out.println("Flies with wings"); }
}
```

```
public class Plane implements Flyable {
    public void fly() { System.out.println("Flies with engines"); }
}
```

Revision #1
Created 23 October 2025 15:43:01 by Hugo
Updated 23 October 2025 19:17:31 by Hugo