Az **OOP** (Object-Oriented Programming, azaz objektumorientált programozás) alapvető **paradigmája** három fő pillérre épül:

---

1. Kapszulázás (Encapsulation)

Mi ez?

A **kapszulázás** azt jelenti, hogy az objektum belső állapotát (változóit) elrejted a külvilág elől, és csak meghatározott módon (metódusokon keresztül) lehet hozzáférni.

Miért jó?

- Megvédi az adatokat (pl. nem tudsz „véletlenül" érvénytelen értéket beállítani).
- Egyszerűsíti a kódot: a belső működés „rejtve marad".

Példa Kotlinban:

```
class BankAccount(private var balance: Int) {
    fun deposit(amount: Int) {
        if (amount > 0) balance += amount
    }
    fun getBalance(): Int = balance
}
```

> `balance` el van kapszulázva: nem férsz hozzá közvetlenül, csak metóduson keresztül.

---

2. Öröklés (Inheritance)

Mi ez?

Az **öröklés** lehetővé teszi, hogy egy osztály (gyermek) **átvegye** egy másik osztály (szülő) tulajdonságait és viselkedését.

Miért jó?

- Újrahasznosítható kód.
- Hierarchikus kapcsolat a típusok között.

Példa Kotlinban:

```
open class Animal {
    fun eat() = println("Eating...")
}

class Dog : Animal() {
    fun bark() = println("Woof!")
}
```

> A `Dog` osztály örökli az `Animal` osztály `eat()` metódusát.

---

3. Polimorfizmus (Polymorphism)

Mi ez?

A **polimorfizmus** azt jelenti, hogy egy metódus **többféleképpen** viselkedhet attól függően, milyen típusú objektum hívja meg.

Miért jó?

- Rugalmasabb, bővíthetőbb kód.
- Ugyanazt az interfészt (vagy absztrakt metódust) különböző osztályok **máshogy** valósíthatják meg.

Példa Kotlinban:

```
open class Animal {
    open fun makeSound() = println("Some sound")
}

class Cat : Animal() {
    override fun makeSound() = println("Meow")
}

class Dog : Animal() {
    override fun makeSound() = println("Woof")
}

fun playSound(animal: Animal) {
    animal.makeSound()
}
```

> Bármilyen `Animal` típusú objektumot átadhatsz a `playSound()` függvénynek, és az a konkrét osztály szerint fog működni (ez a polimorfizmus).

Összefoglalva:

Fogalom	Jelentés	Haszna
Kapszulázás	Adatok elrejtése	Biztonság, egyszerűség
Öröklés	Tulajdonságok átvétele	Kód újrafelhasználás
Polimorfizmus	Többféle viselkedés ugyanazon hívással	Rugalmasság

---