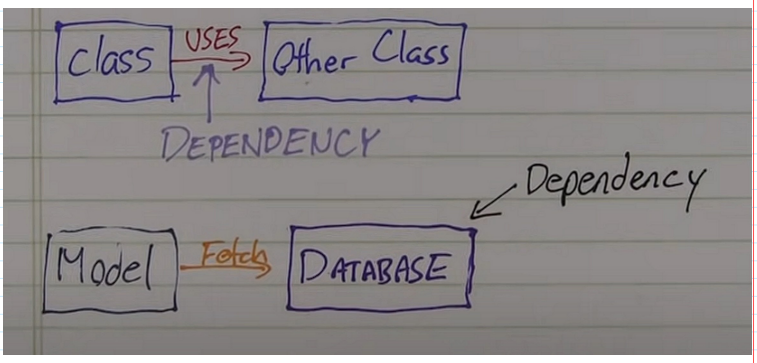
Dependency Injection/Circular Dependency

8:42 PM Thursday, August 1, 2024

- Dependency/תלות

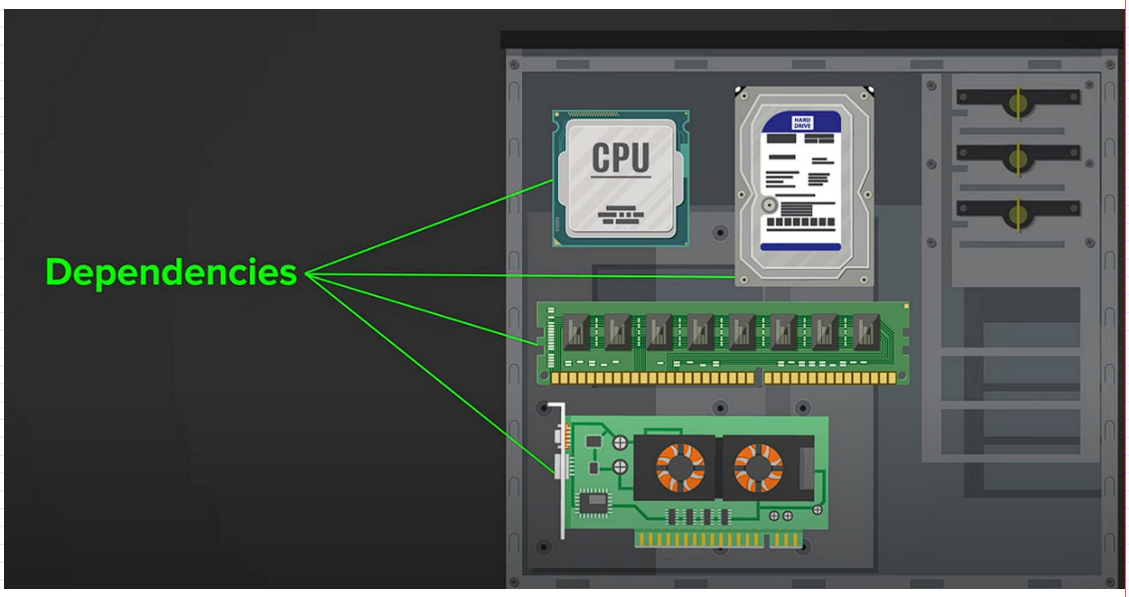
זהו השימוש באובייקט של מחלקה מסוימת שנשענת על מחלקה שונה ממנה. (המחלקה תלויה באובייקט ומכאן השם)



- Dependency Injection

Dependency Injection הוא Design pattern שמטרתו הוא לאפשר בחירה מבין מגוון מימושים של תלות מסוימת.

דוגמא: נרצה לבנות קוד שידמה תבנית של מחשב נייח, לכל מחשב נייח יש חלקים שהוא בנוי ממנו (כרטיס מסך, מעבד, Ram...) ושבלעדיהם הוא לא היה עובד, אלה הם התלויות שלו.



לכל חלק שממנו בונים את המחשב יש גרסאות שונות, חברות שונות שייצרו אותם צורה שונה ועוד...
שנרצה לבנות את הקוד נתחשב בזה ולא נגדיר את המשתנים של הגרסאות, חברות וכו' ... בתור משתנים שברגע ששינינו אותם כל המחשבים שבקוד ישתנו לאותו המשתנה, אלא שלכל מחשב שונה יהיו תכונות שונות של החלקים שלו. זוהי הבעיה אותה Dependency Injection פותרת.

```
val computer1 = Computer(
    Processor("Intel i7"),
    RAM(32),
    HardDrive(1024),
    GraphicsCard("NVIDIA GeForce")
)

val computer2 = Computer(
    Processor("AMD Ryzen"),
    RAM(16),
    HardDrive(2048),
    GraphicsCard("AMD")
)
```

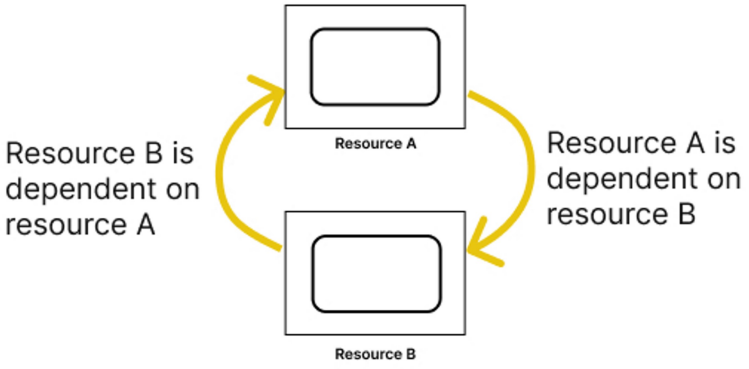
```
class Computer {

    private val processor = Processor("Intel i7")
    private val ram = RAM(32)
    private val hardDrive = HardDrive(1024)
    private val graphicsCard = GraphicsCard("NVIDIA GeForce")

}
```

- Circular Dependency

זהו מקרה שבו שתי אובייקטים נשענים זה על זה, כל אחד קורא להפעלה של השני ובעצם נוצר כך לופ אין סופי ובו כל אובייקט בונה את האובייקט השני, או לחילופין איננו מביר את האובייקט השני כי עדיין לא בנה אותו.
את זאת ניתן לפתור על ידי הפקודה **fowardref** המאפשרת להתייחס להפניות שעדיין לא הוגדרו.


Resource B is dependent on resource A
Resource A
Resource A is dependent on resource B
Resource B