

SOLID PRENSİPLERİ ve SÜRDÜRÜLEBİLİR YAZILIM

ESRANUR SEVİLMİŞ
BİLGİSAYAR MÜHENDİSLİĞİ 3.SINIF

20360859026

28.03.2024

İÇERİK

1

SOLID KAVRAMI

2

ÖZELLİKLER

3

SINGLE RESPONSIBILITY

4

OPEN-CLOSED

5

LISKOV SUBSTITUTION

6

INTERFACE SEGREGATION

7

DEPENDENCY INVERSION

8

SÜRDÜRÜLEBİLİR YAZILIM

9

GEREKLİLİKLER

10

KAYNAKÇA

11

SORU-CEVAP

12

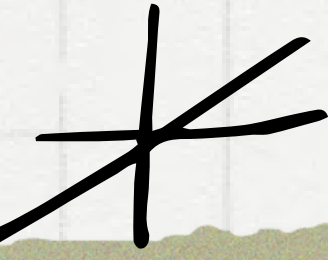
TEŞEKKÜR

SOLID NEDİR?

- ✧ Kuralların baş harflerinden oluşan S.O.L.I.D prensipler bütünüdür.
- ✧ Yazılım mühendisliğinde geliştiricilerin daha kaliteli yazılımlar oluşturmalarına yardımcı olan beş tasarım prensibinden oluşan bir settir.

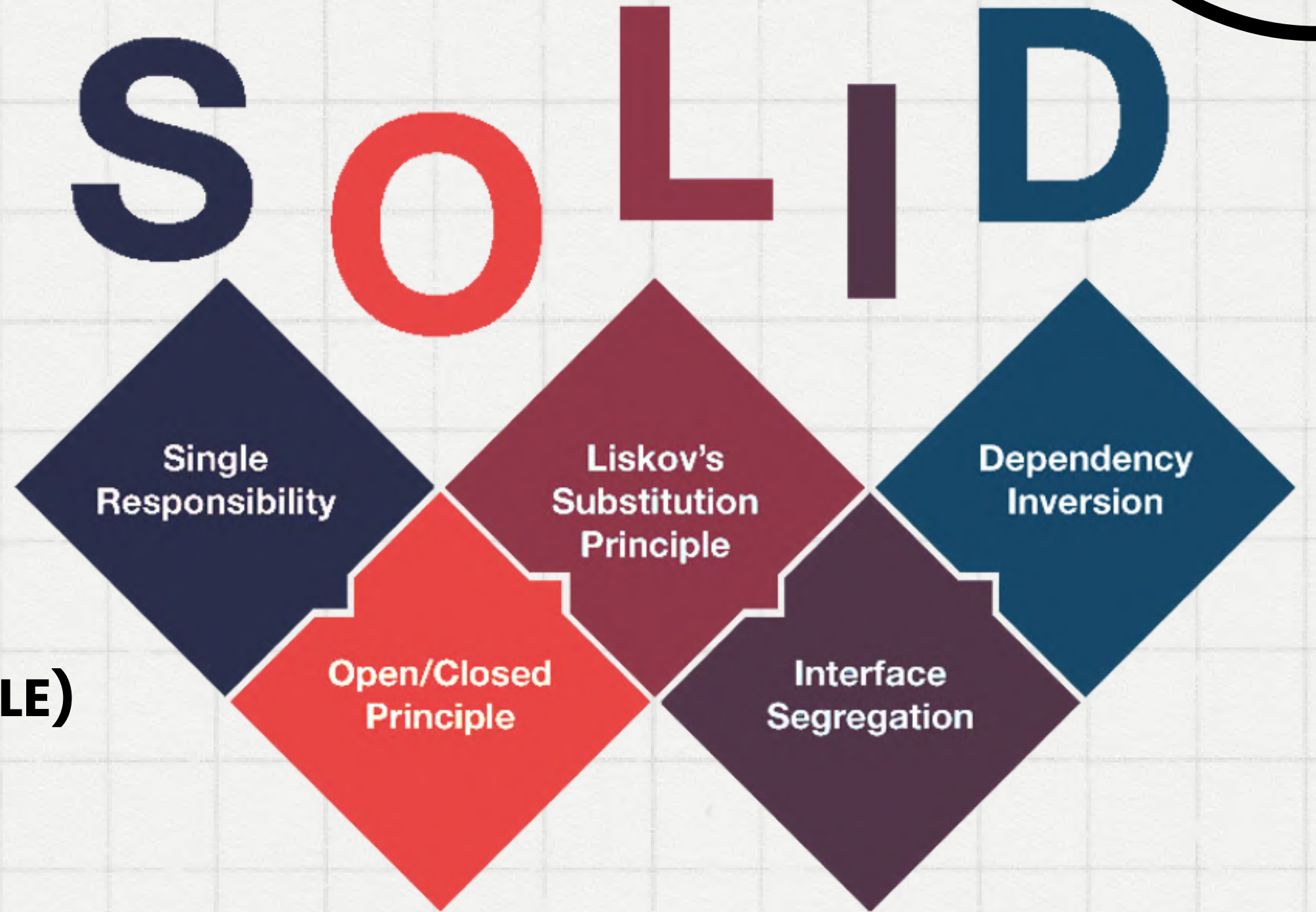


Bu set, Amerikalı yazılım mühendisi ve eğitmen **Robert C. Martin** tarafından ilk olarak 2000 yılında yayımladığı "**Design Principles and Design Patterns**" başlıklı makalesinde tanıtılmıştır.

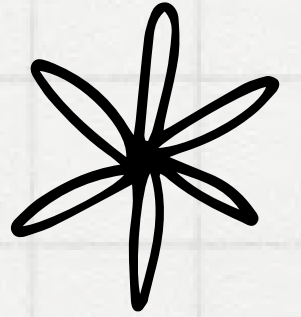


Özellikler

- * ANLAŞILIR (UNDERSTANDABLE)
- * ESNEK (FLEXIBLE)
- * BAKIMI KOLAY (MAINTAINABLE)
- * ÖLÇEKLENEBİLİR (SCALABLE)
- * YENİDEN KULLANILABİLİR (REUSABLE)
- * OKUNABİLİR (READABLE)
- * TEST EDİLEBİLİR (TESTABLE)



Single Responsibility



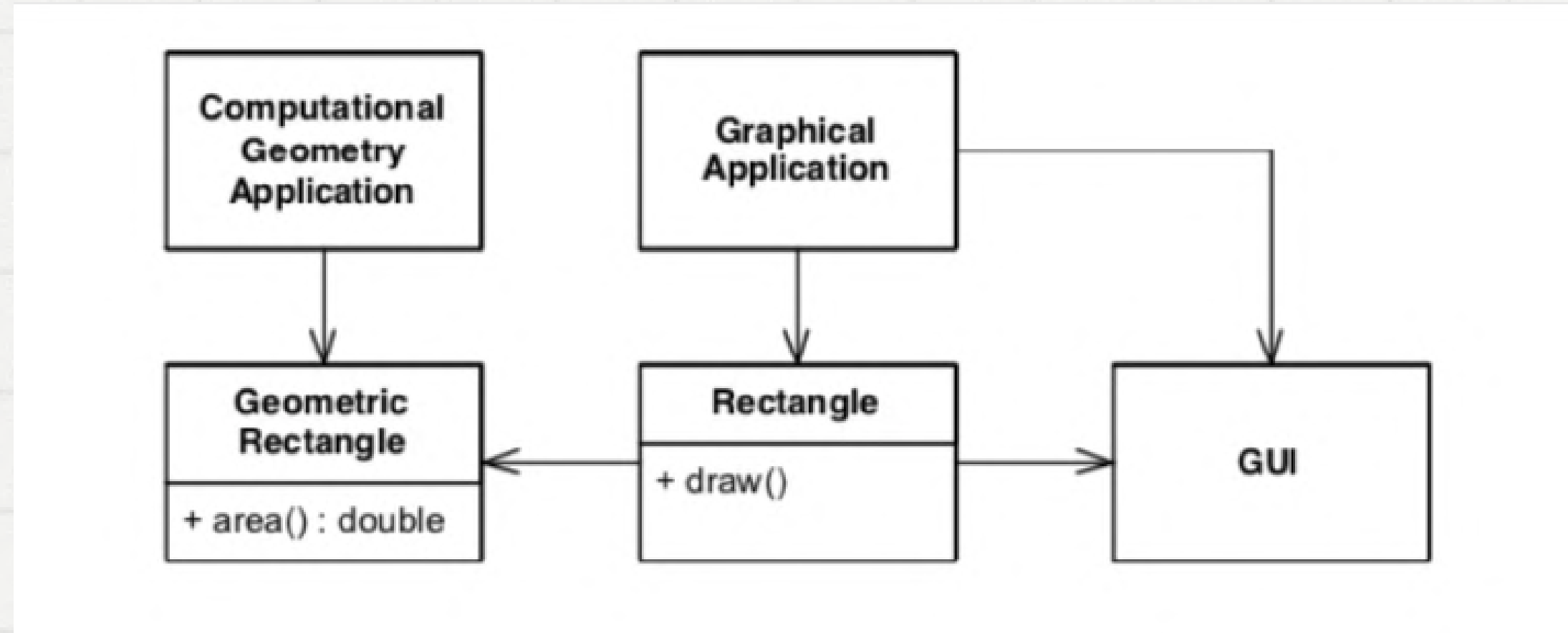
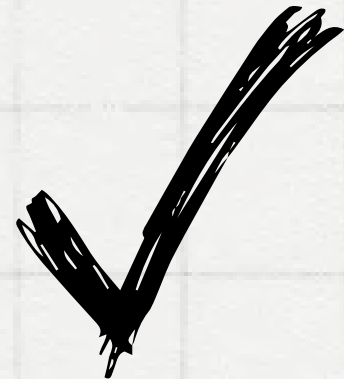
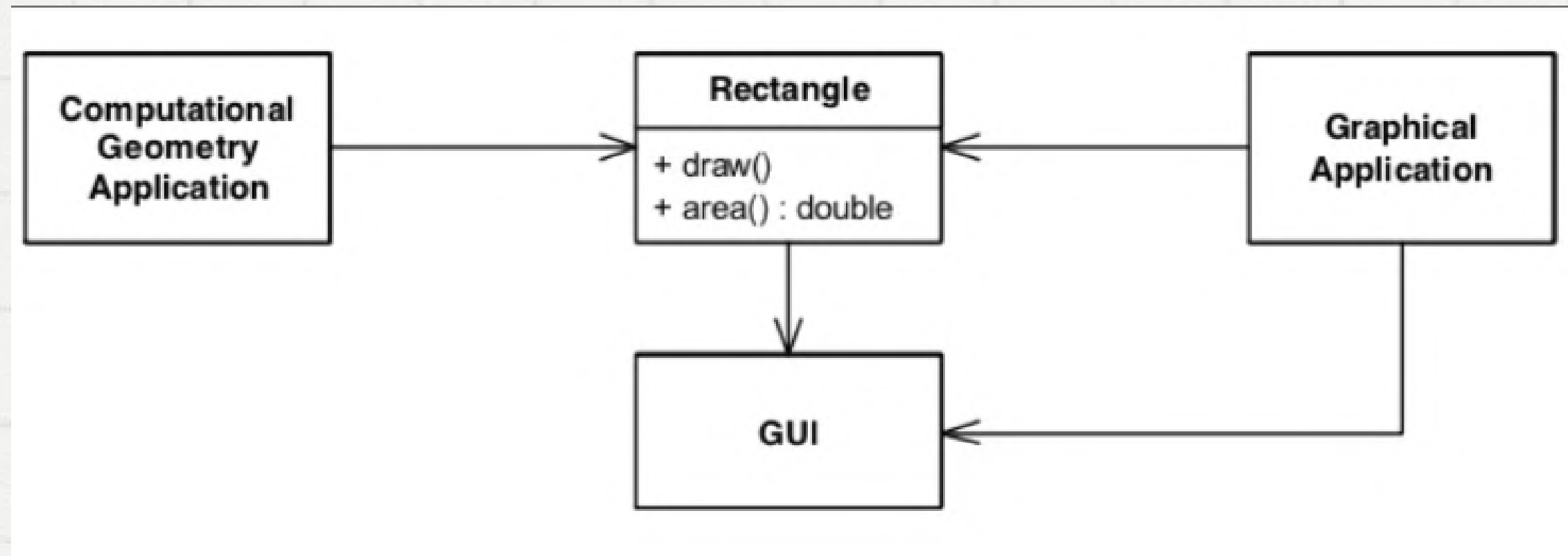
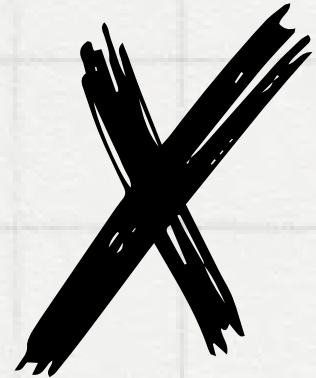
"A class should have one and only one reason to change, meaning that a class should have only one job."



- > Tek bir sorumluluğu olan bir sınıfın çok daha az test senaryosuna sahip olur.
- > Tek bir sınıfta daha az işlevsellik, daha az bağımlılığa sahip olacaktır.
- > Daha küçük, iyi organize edilmiş sınıfları, monolitik olanlardan aramak daha kolaydır.

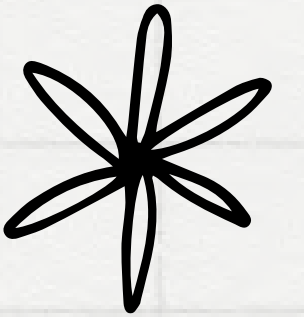
Single Responsibility Principle

Just because you can doesn't mean you should.



Open/Closed

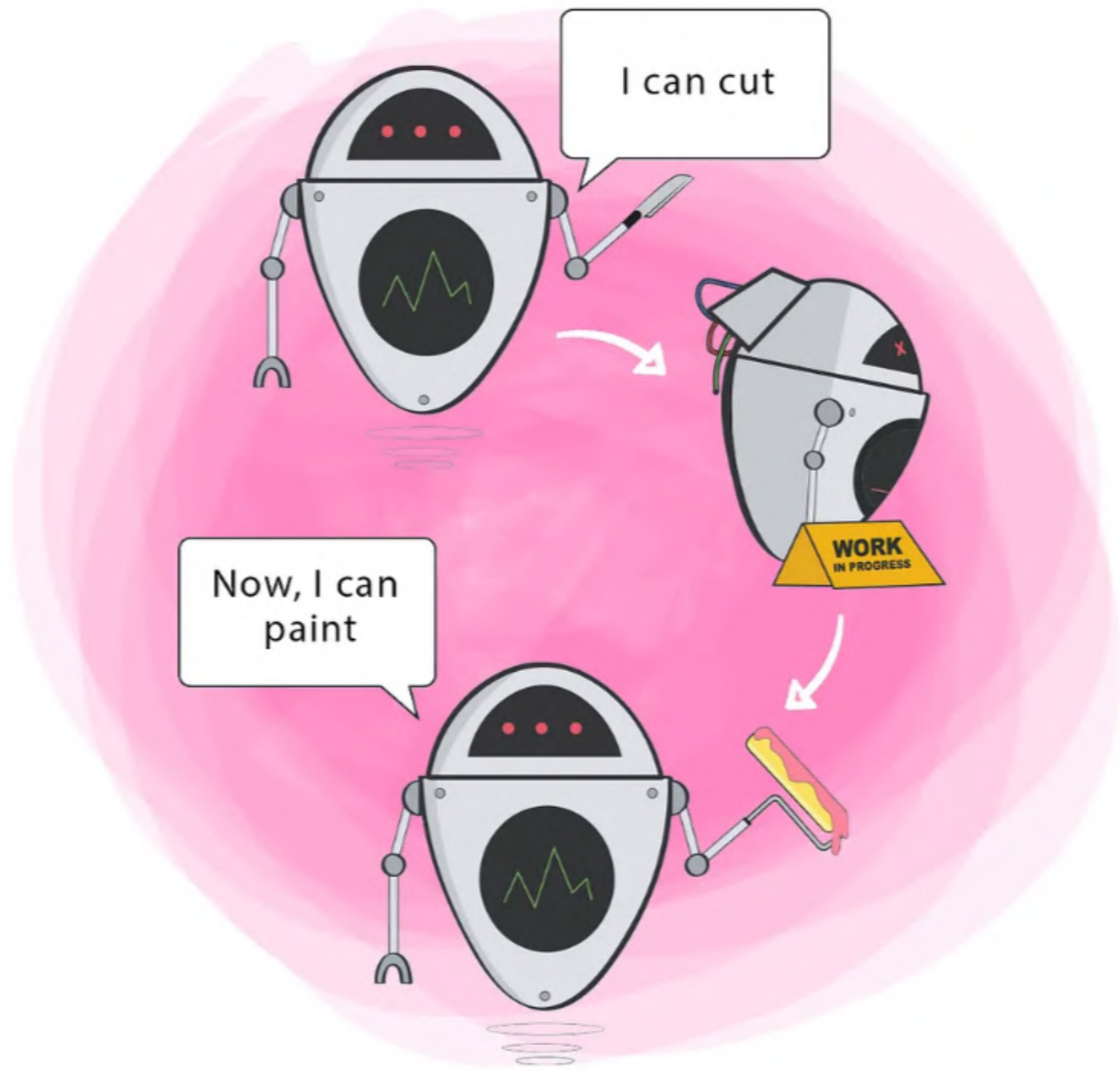
"Software entities should open for extension but closed for modification"



Open-Closed Principle

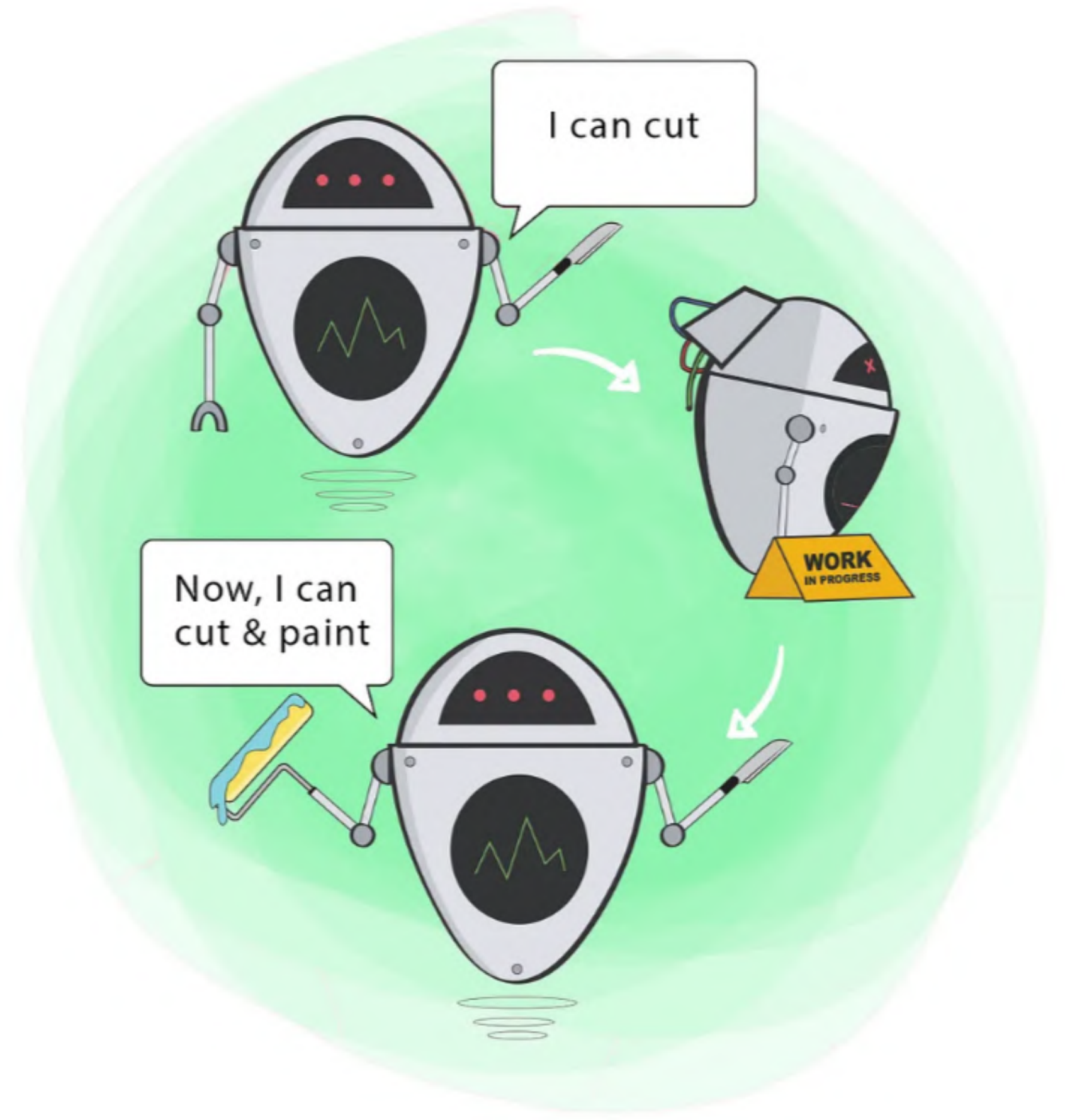
Open-chest surgery isn't needed when putting on a coat.

- > Her şeyi baştan yazmak yerine, eklemeler yaparak yeni özelliği entegre edebiliriz.
- > Bu sayede kod daha anlaşılır, daha az hata barındırır ve ileride bakımı kolaylaşır.
- > Aksi takdirde , bu program "kötü" tasarımıla ilişkilendirdiğimiz istenmeyen nitelikleri sergiler.



✗

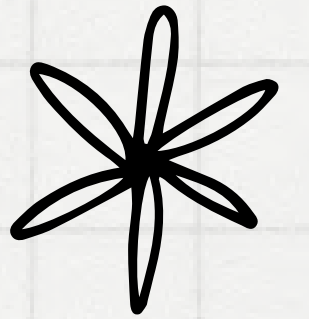
Open-Closed



✓

7/17

Liskov Substitution



“Subclasses should be substitutable for their base classes.”

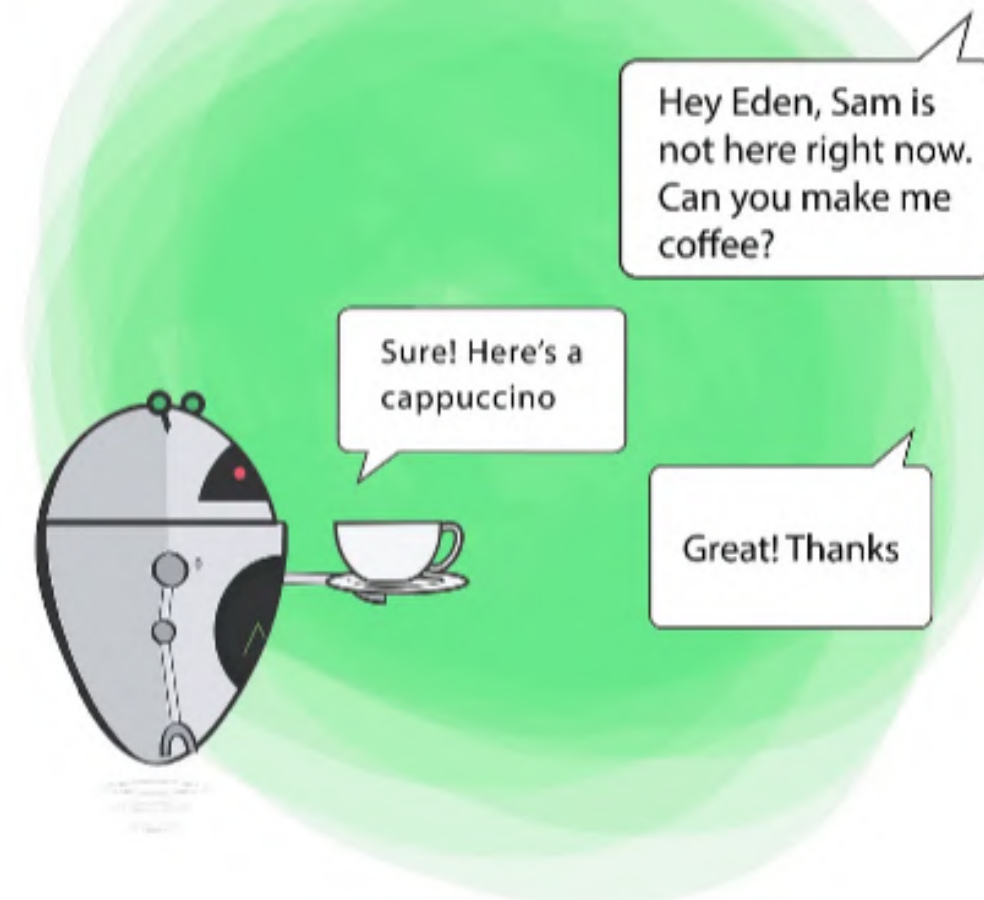
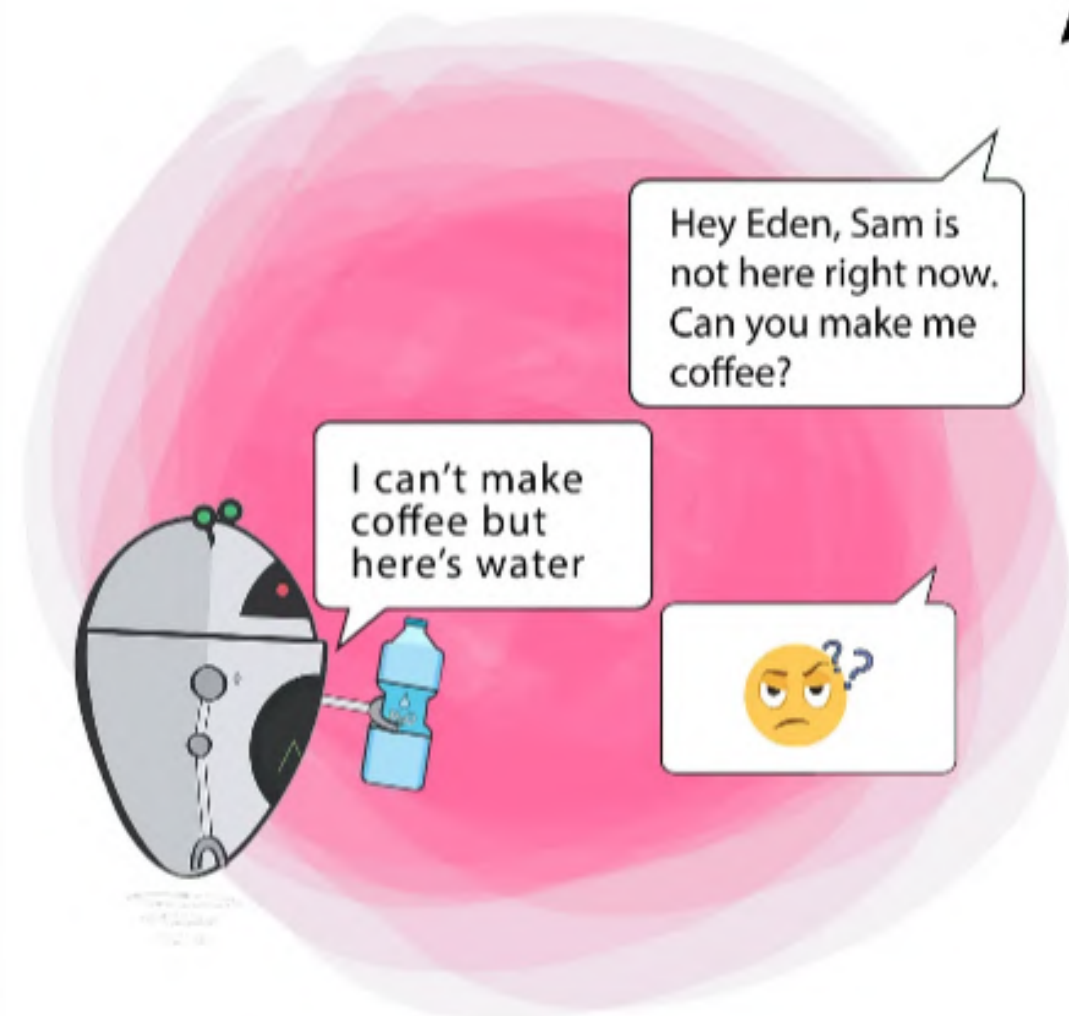
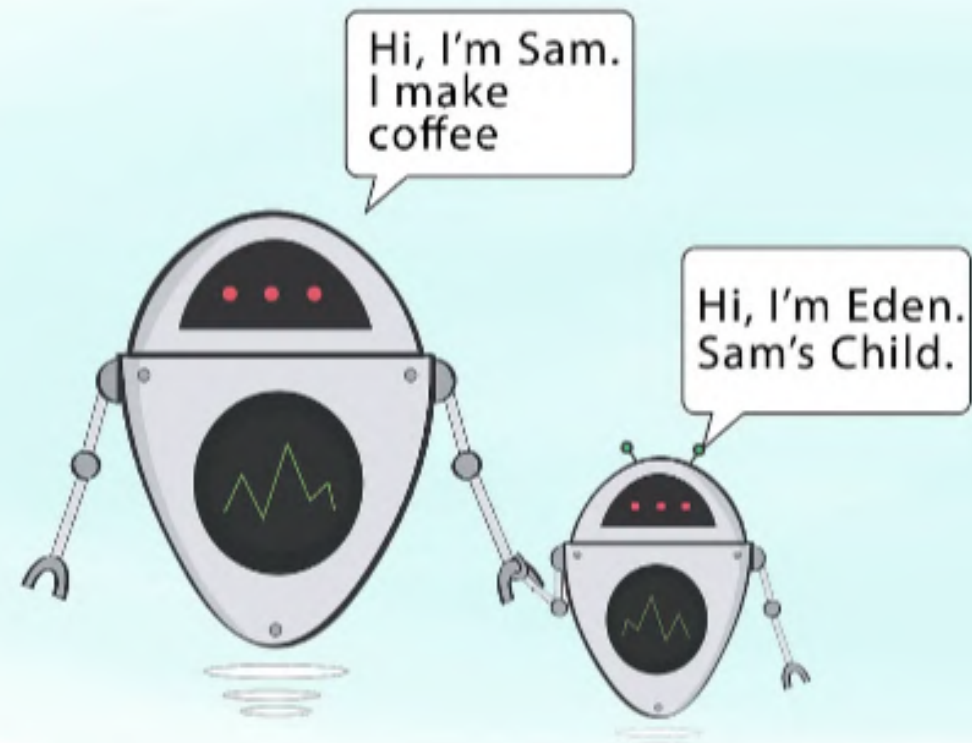
→ MIT programlama metodolojileri grup liderliği yapan **Barbara Liskov** tarafından öne sürülmüştür.

→ Bu prensip, hata olmadan ana sınıfın veya alt sınıfının aynı şekilde kullanılabilmesi için tutarlılığı sağlamayı amaçlar.



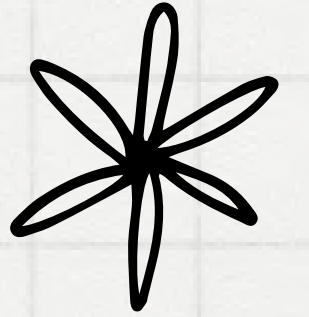
LISKOV SUBSTITUTION

If it looks like a duck, quacks like a duck, but needs batteries —
you probably have the wrong abstraction.



Liskov Substitution

Interface Segregation



"Many client specific interfaces are better than one general purpose interface."

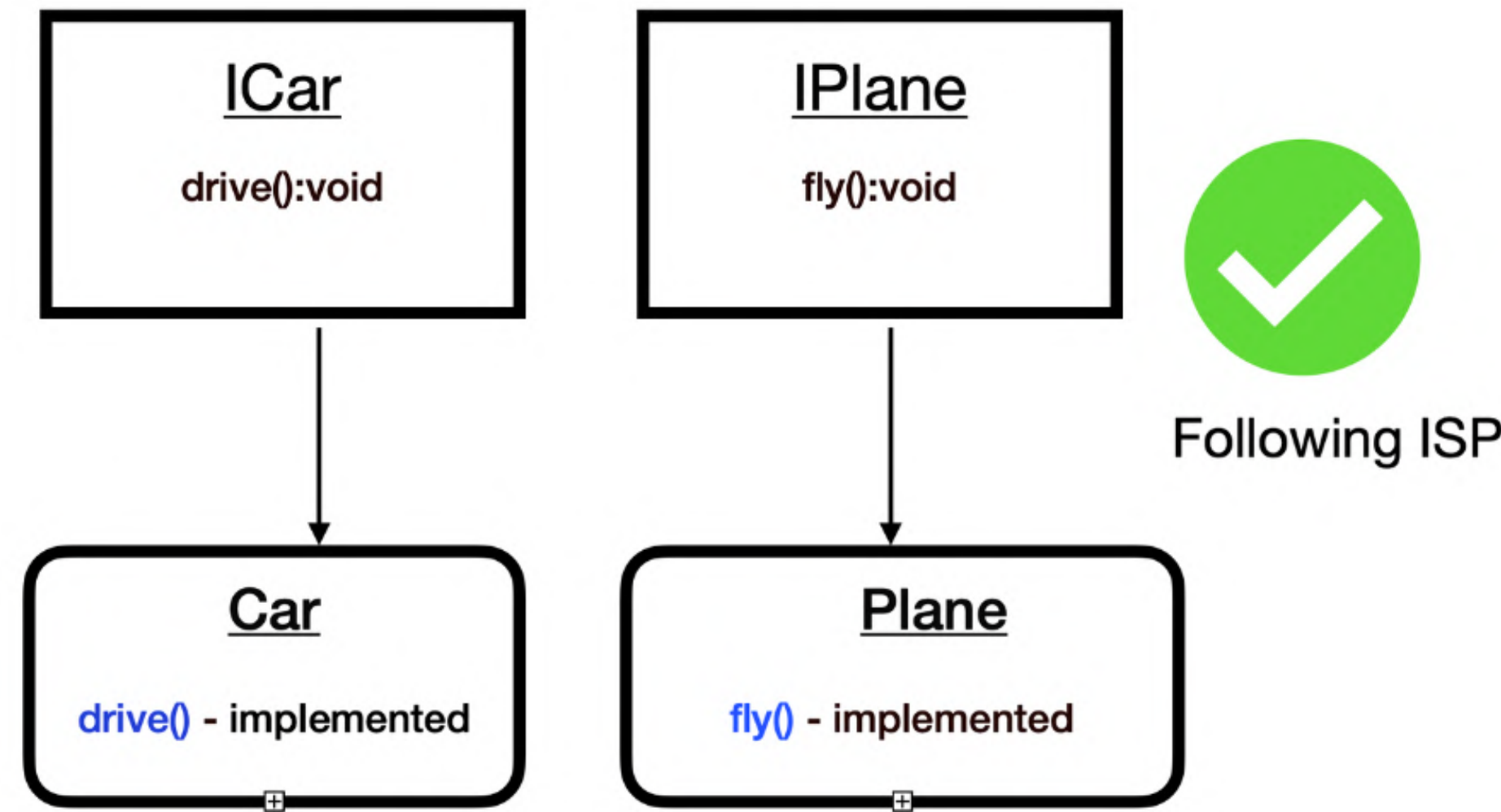
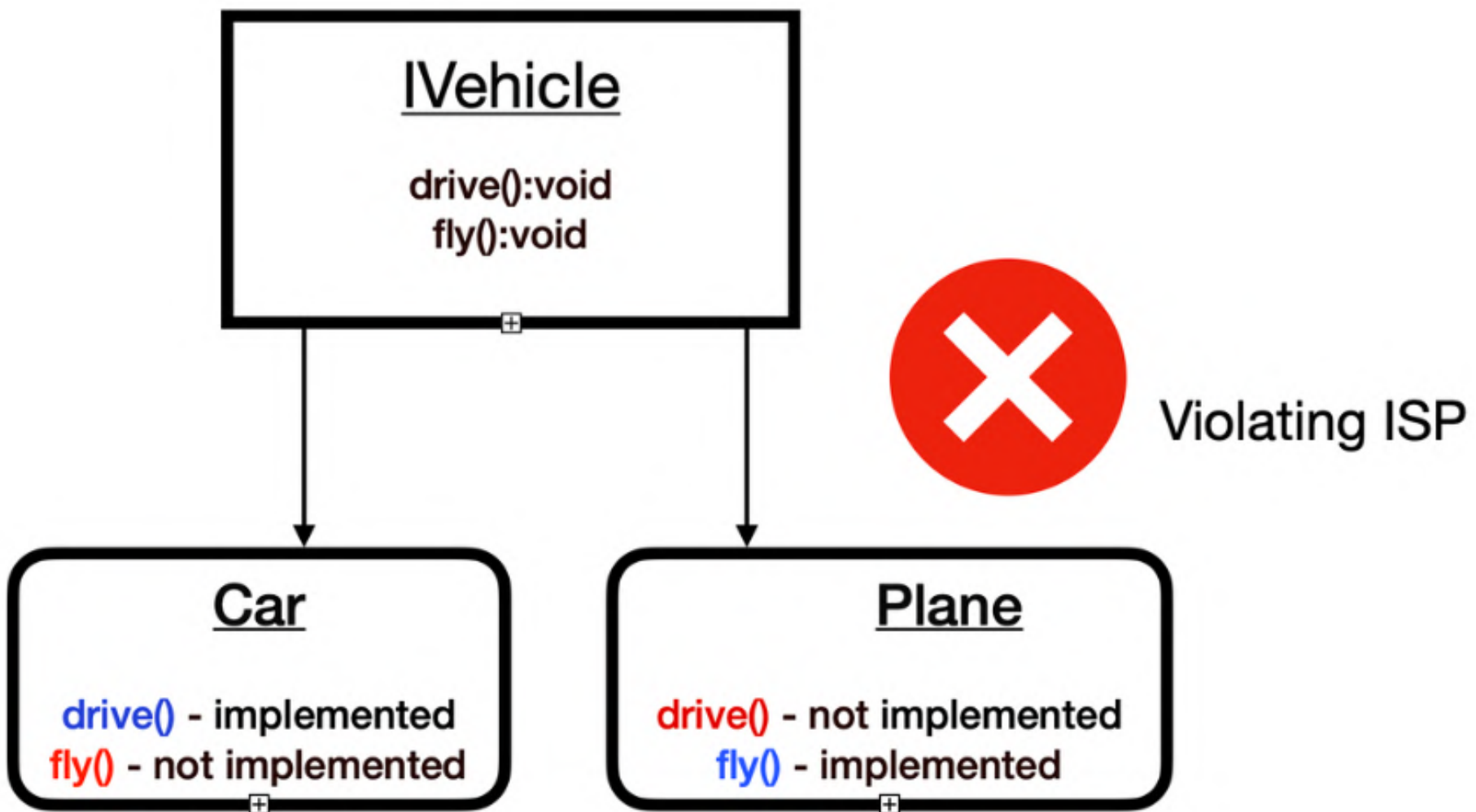
Nesneler asla ihtiyacı olmayan property/metot vs içeren interfaceleri implement etmeye zorlanmamalıdır.

Tek bir arayüze toplamak yerine daha özelleştirilmiş birden fazla arayüz oluşturma esastır.



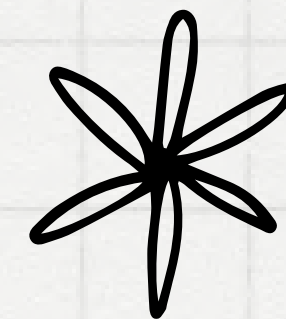
INTERFACE SEGREGATION PRINCIPLE

Don't force the client to depend on things they don't use.



Dependency Inversion

"Depend upon Abstractions. Do not depend upon concretions."



- > Sınıflar arası bağımlılıklar olabildiğince az olmalıdır özellikle üst seviye sınıflar alt seviye sınıflara bağımlı olmamalıdır.
- > Bir alt sınıfta yapılan değişiklikler üst sınıfları etkilememelidir.

Dependency Inversion Principle

Would you solder a lamp directly to the electrical wiring in a wall?

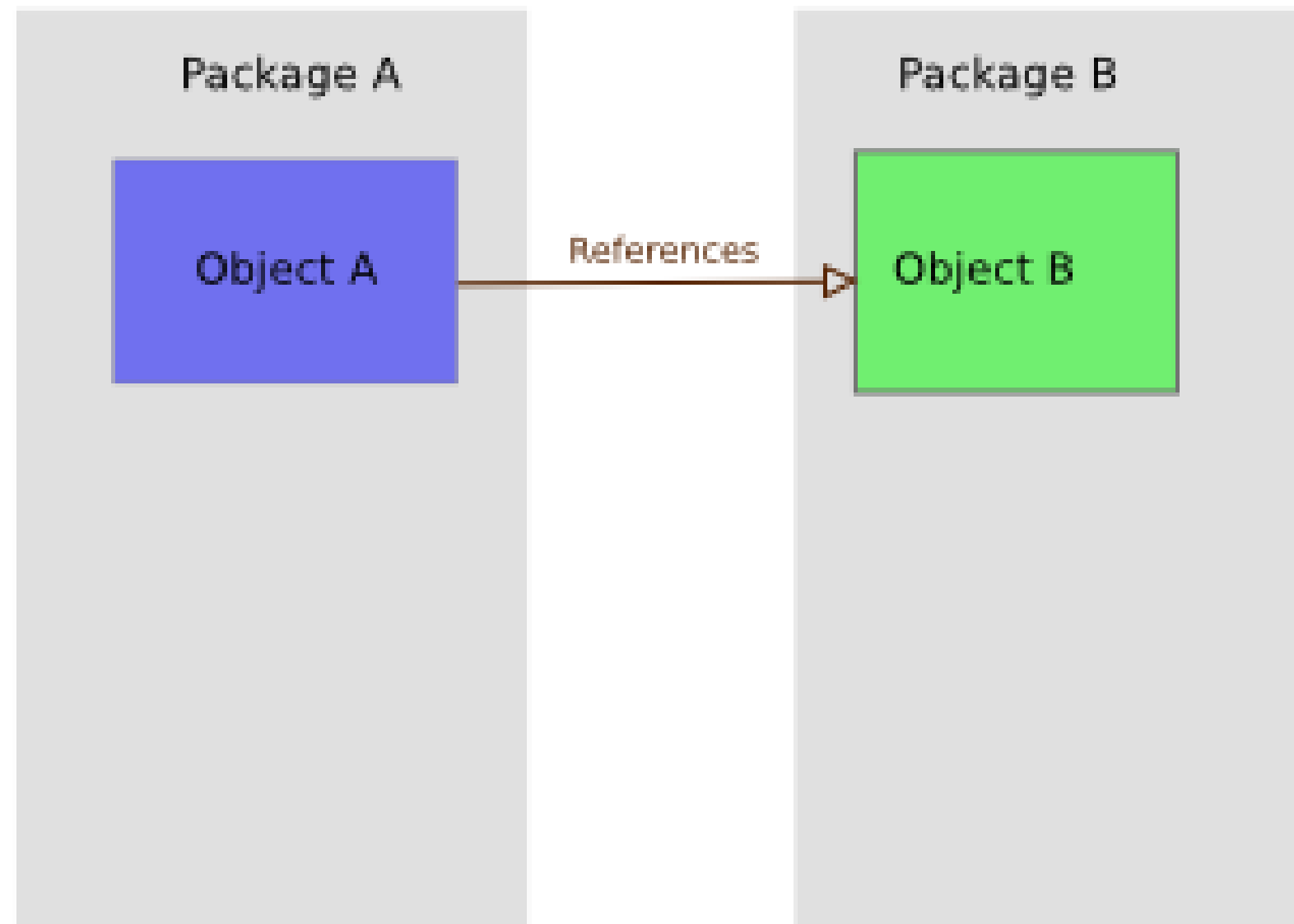


Figure 1

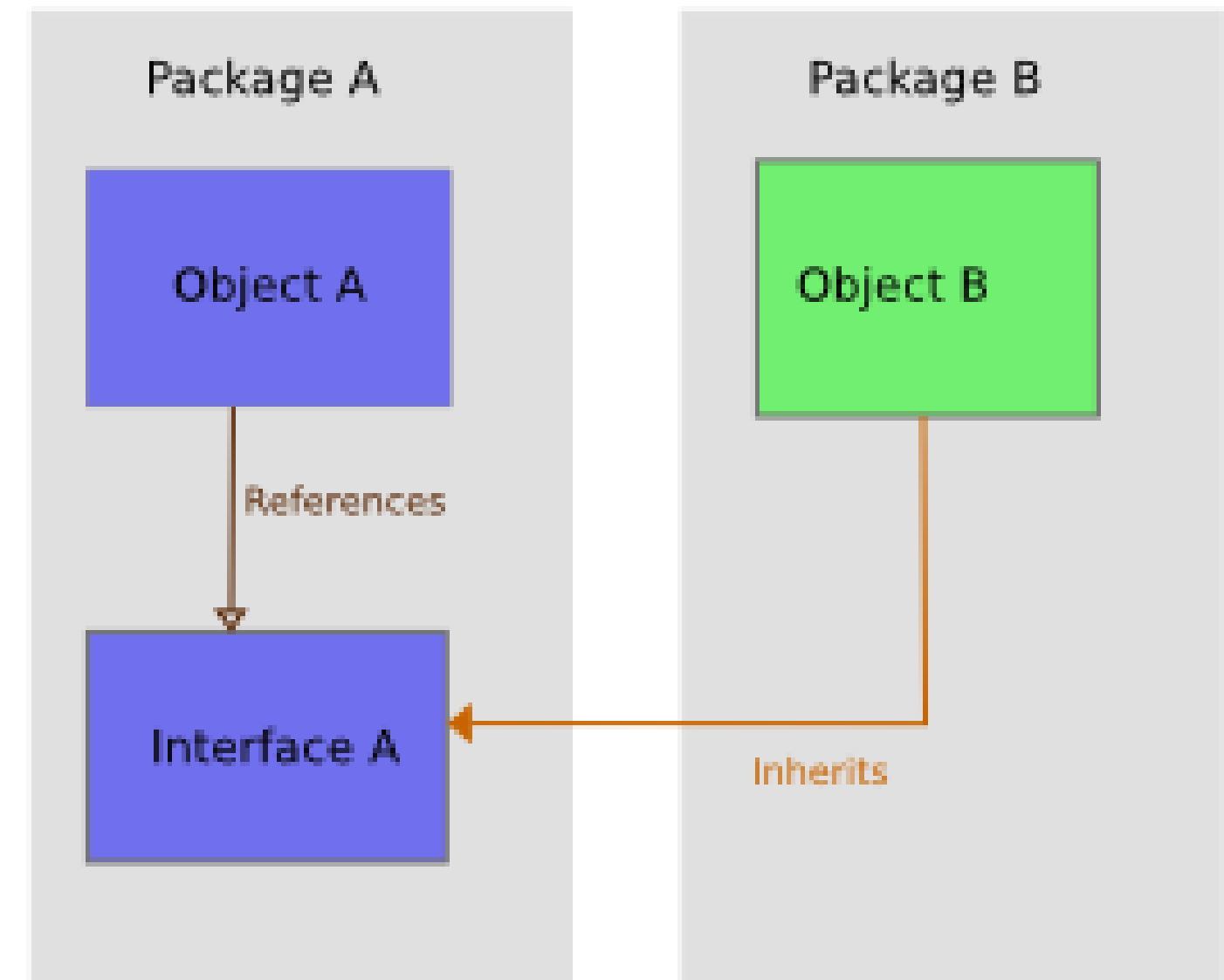
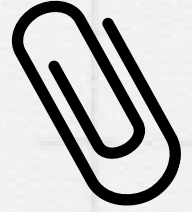
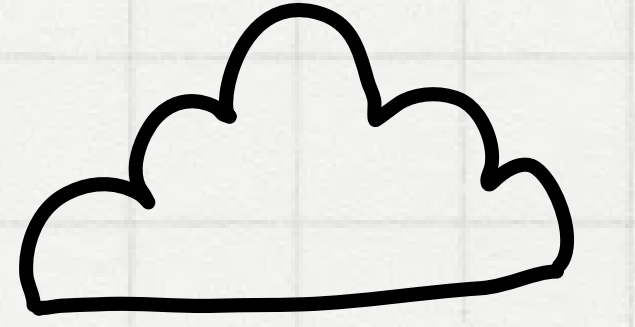
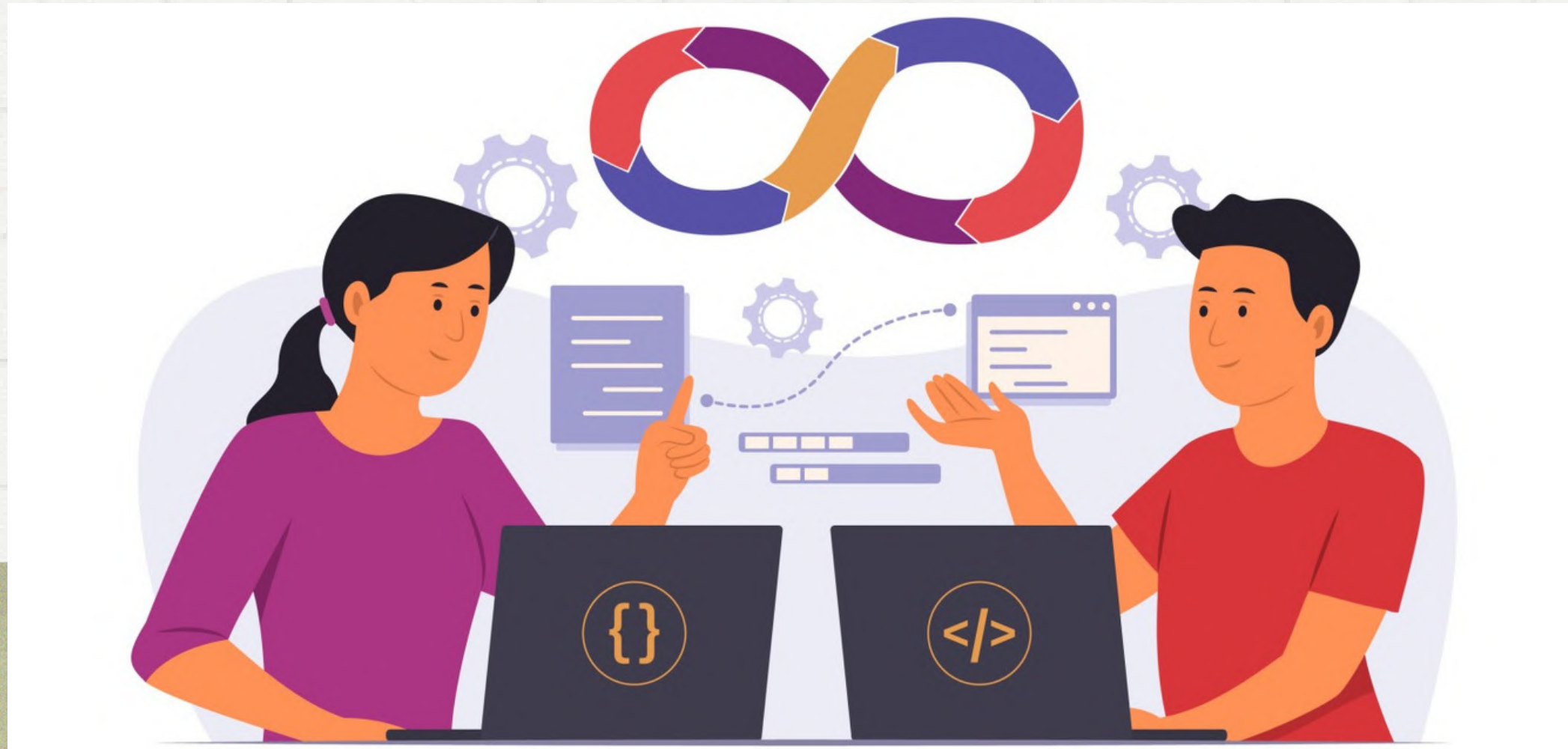


Figure 2

Sürdürülebilir Yazılım Nedir?




- 10 sene sonra bile sorunsuz kullanılabiliyorsa.
- Yeni yetenekler eklenebiliyorsa
- Yeni kurallar konulabiliyorsa
- Yazılımcıdan/firmadan bağımsız geliştirilebiliyorsa buna sürdürülebilir, canlı yazılım diyoruz.



Gereklilikler



- * **Değiştirilebilir olmalıdır.**
- * **Prensipilere uygun olmalıdır.**
- * **Clean Code ile programlanmalıdır.**
- * **Doküman olmalıdır.** 
- * **Versiyon kontrol sistemi ile çalışılmalıdır.(Git,Microsoft)**

Dokümanda şöyle yazıyor:

“En son kesilen fatura tarihinden önce bir tarihe fatura kesilemez.”

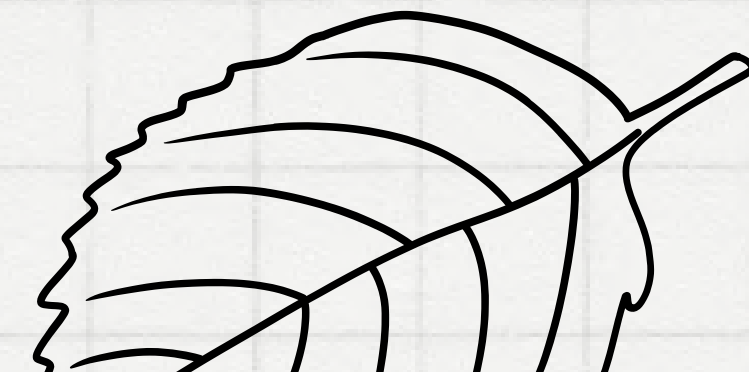
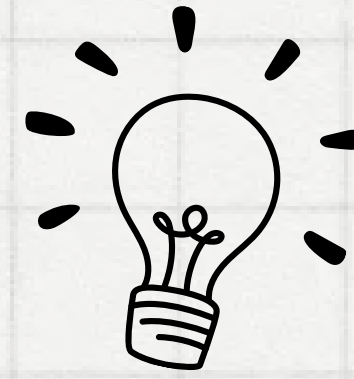
Yazılımda şöyle yazıyor:

```
function checkInvoiceDate(invoiceDate) {  
    if (invoiceDate < lastInvoiceDate)  
        return false;  
}
```

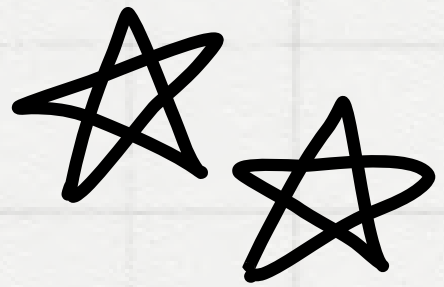
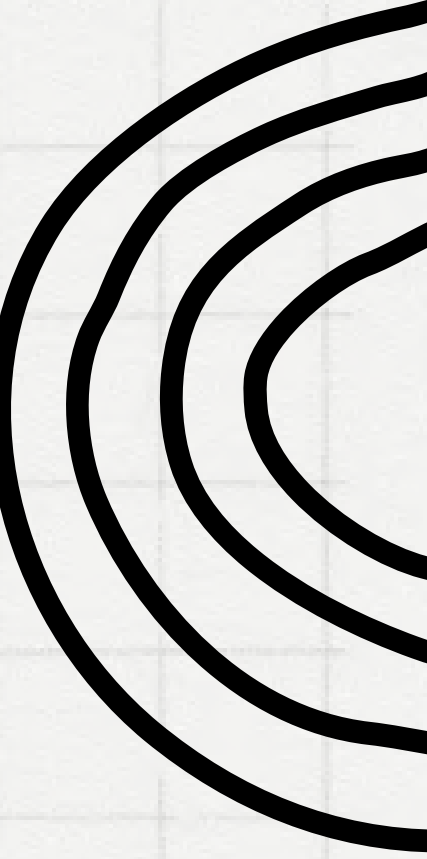
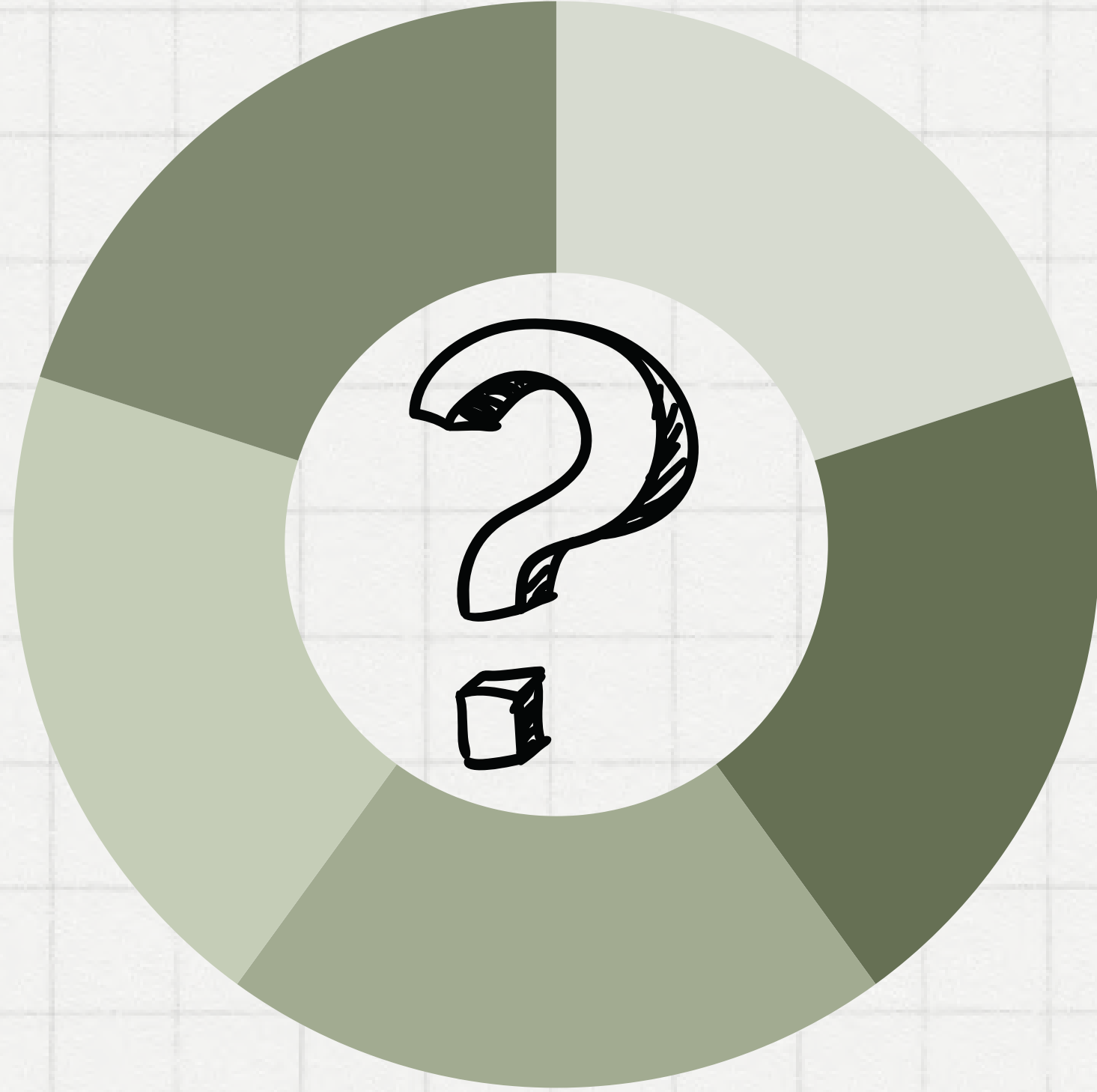


Kaynakça

- <https://www.baeldung.com/solid-principles#:~:text=The%20SOLID%20principles%20were%20introduced,us%20to%20the%20SOLID%20acronym>
- <https://aycaakcay.medium.com/solid-prensipleri-ve-s%C3%BCrd%C3%BCr%C3%BClebilir-yaz%C4%B1%C4%B1m-7fafde6034ba>
- <https://mericgungor.medium.com/kavram-surdurulebilirlik-8f8ea1f540fa>
- https://dev.to/antonov_mike/how-can-applying-the-solid-principles-make-the-code-better-3fam
- <https://code.likeagirl.io/solid-principles-in-java-cf9f5b167600>
- <https://medium.com/backticks-tildes/the-s-o-l-i-d-principles-in-pictures-b34ce2f1e898>



Soru-Cevap



Teşekkürler