

Kurumsal Java .com

Java Enterprise Architecture



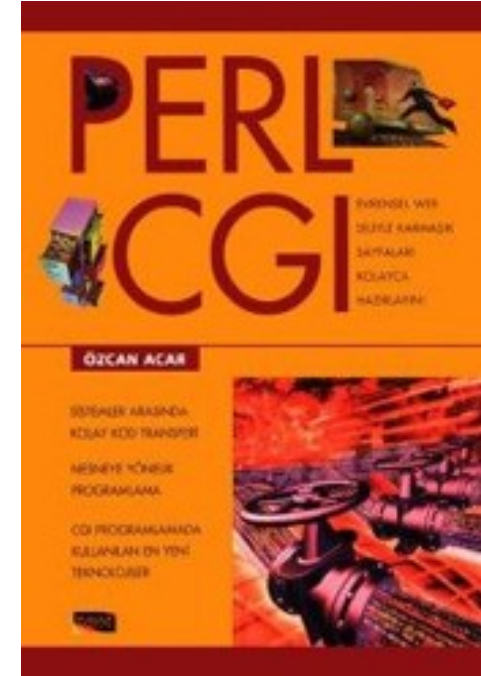
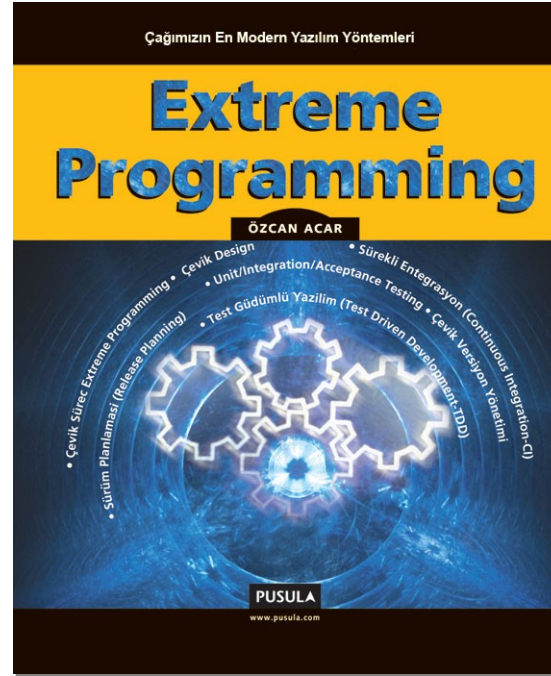
Tasarım Prensipleri

Özcan Acar
acar@unitedinter.net
<http://www.ozcanacar.com>
<http://www.kurumsaljava.com>

Özcan Acar Hakkında

```
public class OezcanAcar
{
    public static void main(String[] args)
    {
        Acar oezcan = new Acar();
        oezcan.setBirthday("18.07.1974");
        oezcan.setBirthplace("Izmir");
        oezcan.setJob("Bilgisayar Mühendisi");
        oezcan.setPassion("Java EE");
    }
}
```

Özcan Acar Hakkında



Java Tasarım Şablonları ve Yazılım Mimarileri



Tasarım şablonu nedir?

Interface / Abstract sınıf nedir?

Oluşturucu Tasarım Şablonları

**Factory, Abstract Factory,
Builder, Prototype,**

Singleton

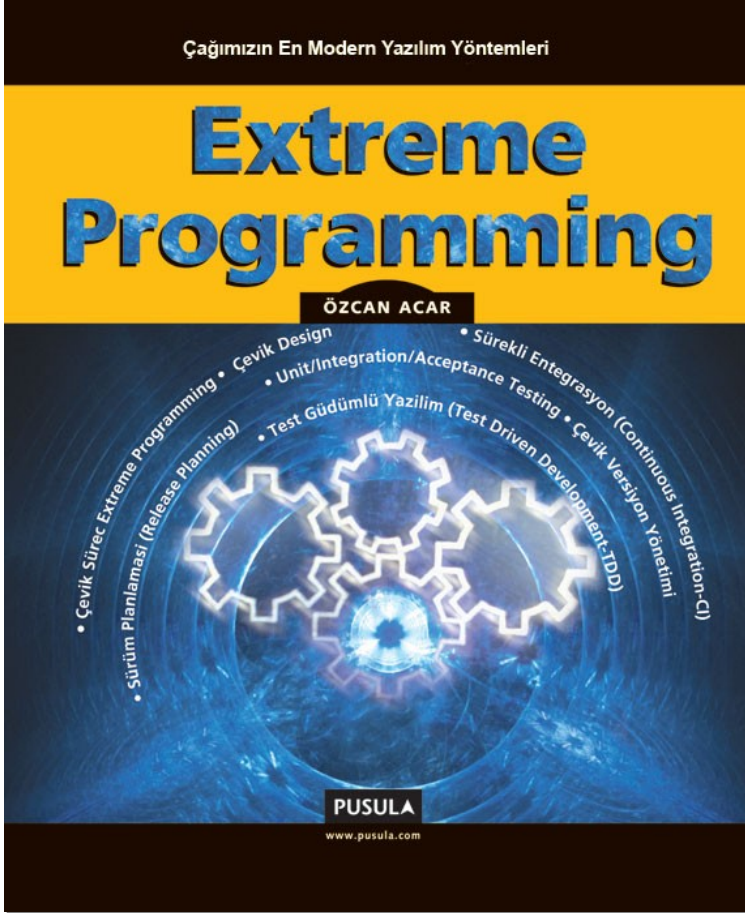
Yapısal Tasarım Şablonları

**Adapter, Bridge, Facade,
Decorator, Composite, Flyweight,
Proxy**

Davranışsal Tasarım Şablonları

**Command, Memento,
Strategy, Iterator, State, Observer,
Visitor**

Extreme Programming



Çevik süreç nedir?

Çevik manifesto

Extreme Programming nedir?

XP değerleri ve prensipleri

Çevik proje planlaması

Çevik süreçlerde iletişim

Çevik tasarım

Sürekli entegrasyon

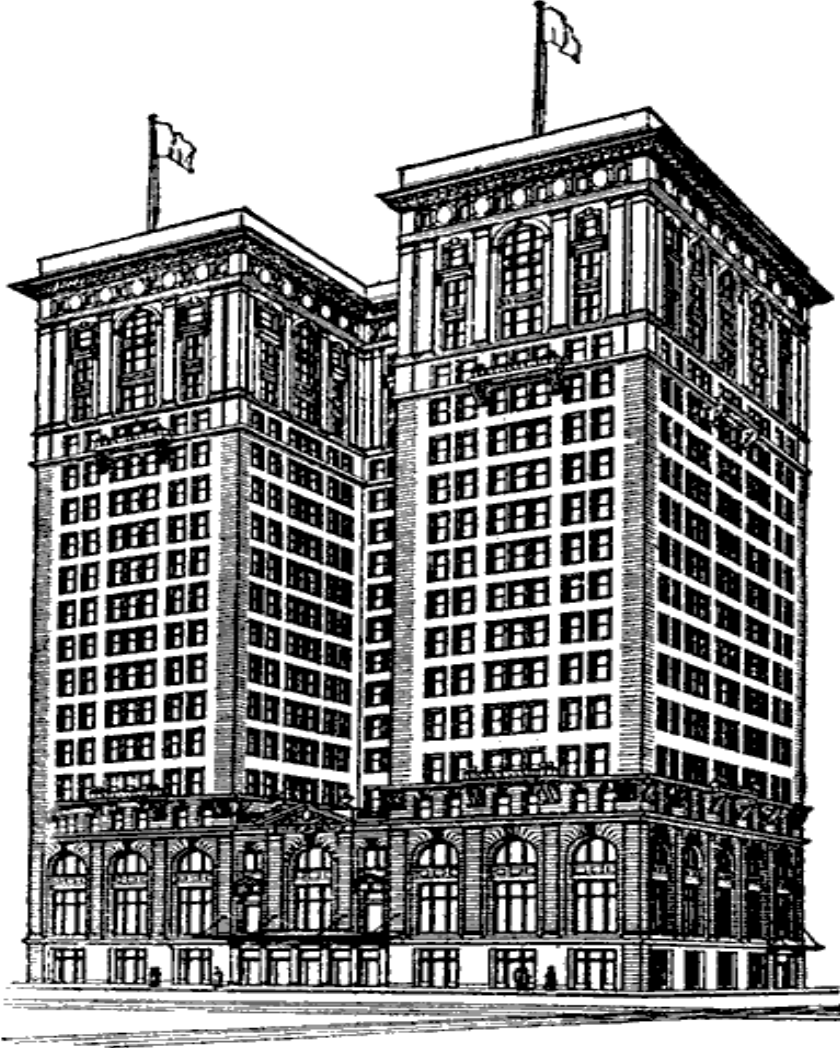
Test güdümlü yazılım

Yazılım metrikleri

Sunumun İçeriği

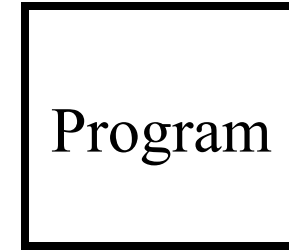
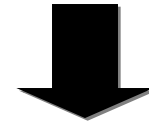
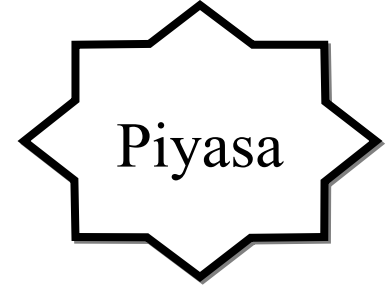
- **Tasarım nedir?**
- **Tasarım nasıl oluşur?**
- **İyi tasarımın amacı nedir?**
- **Kötü tasarım belirtileri**
- **Tasarım Prensipleri**
- **Sınıf bazında tasarım prensipleri**
- **Paket bazında tasarım prensipleri**
- **Test edilebilir tasarım**

Tasarım Nedir?



- Tasarım, bir program için tasarım şablonları ve prensipleri kullanılarak oluşturulan yapıdır.
- İyi bir tasarım, oluşturulan program için hayat sigortasıdır.

Tasarım Nasıl Oluşur?



İyi Tasarımın Amacı

- **Yazılım sisteminin esnekliği**
- **Yazılım sisteminin geliştirilebilirliği**
- **Yazılım sisteminin bakılabilirliği**

Kötü Tasarım Belirtileri

- **Binlerce satırlık sınıflar ve metotlar**
- **Birden fazla sorumluluk yüklenen sınıflar**
- **Paketler (package) arası döngü oluşması**
- **Unit testlerinin olmaması ve dolaylı olarak kodun tekrar yapılandırılabilme (refactoring) özelliğinin zayıflaması**
- **Switch komutunun kullanılmış olması**

Kötü Tasarım Belirtileri

- **Cyclomatic complexity değerinin yüksek olması**
- **Hiçbir tasarım şablonunun kullanılmaması ya da yerli yersiz bilimium tasarım şablonlarının kullanılmış olması**
- **Kod erozyonu...**

Tasarımda Dikkat Edilmesi Gereken Hususlar

- **Bakım (Maintainability)**
- **Performans (Performance)**
- **Genişletilebilirlik (Extensibility)**
- **Uyumluluk (Compatibility)**
- **Güvenilirlik (Reliability, Fault-tolerance)**
- **Tekrar kullanılabilirlik (Reusability)**
- **Güvenlik (Security)**
- **Kullanılabilirlik (Usability)**
- **Modülerlik (Modularity)**

Tasarım Prensipleri

Sınıf Bazında

- **Loose Coupling (LC)**
- **Open Closed Principle (OCP)**
- **Single Responsibility Principle (SRP)**
- **Liskov Substitution Principle (LSP)**
- **Dependency Inversion Principle (DIP)**
- **Interface Segregation Principle (ISP)**

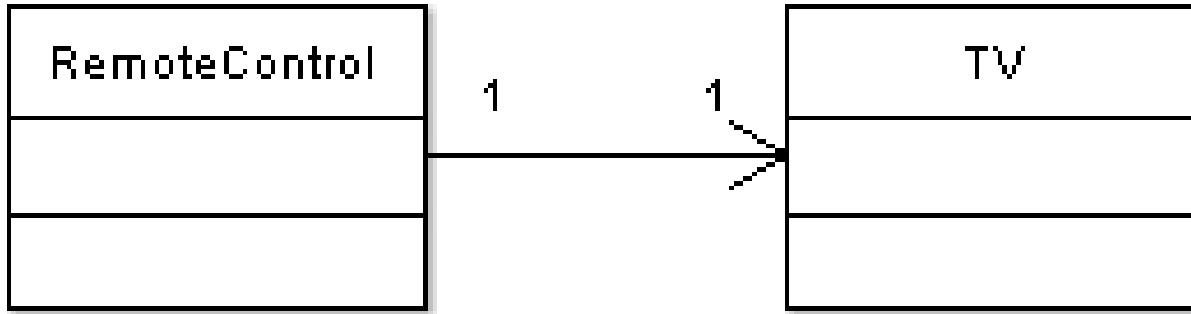
Tasarım Prensipleri

Paket Bazında

- **Reuse-Release Equivalence Principle (REP)**
- **Common Reuse Principle (CRP)**
- **Common Closure Principle (CCP)**
- **Acyclic Dependency Principle (ADP)**
- **Stable Dependencies Principle (SDP)**
- **Stable Abstractions Principle (SAP)**

Esnek Bağ

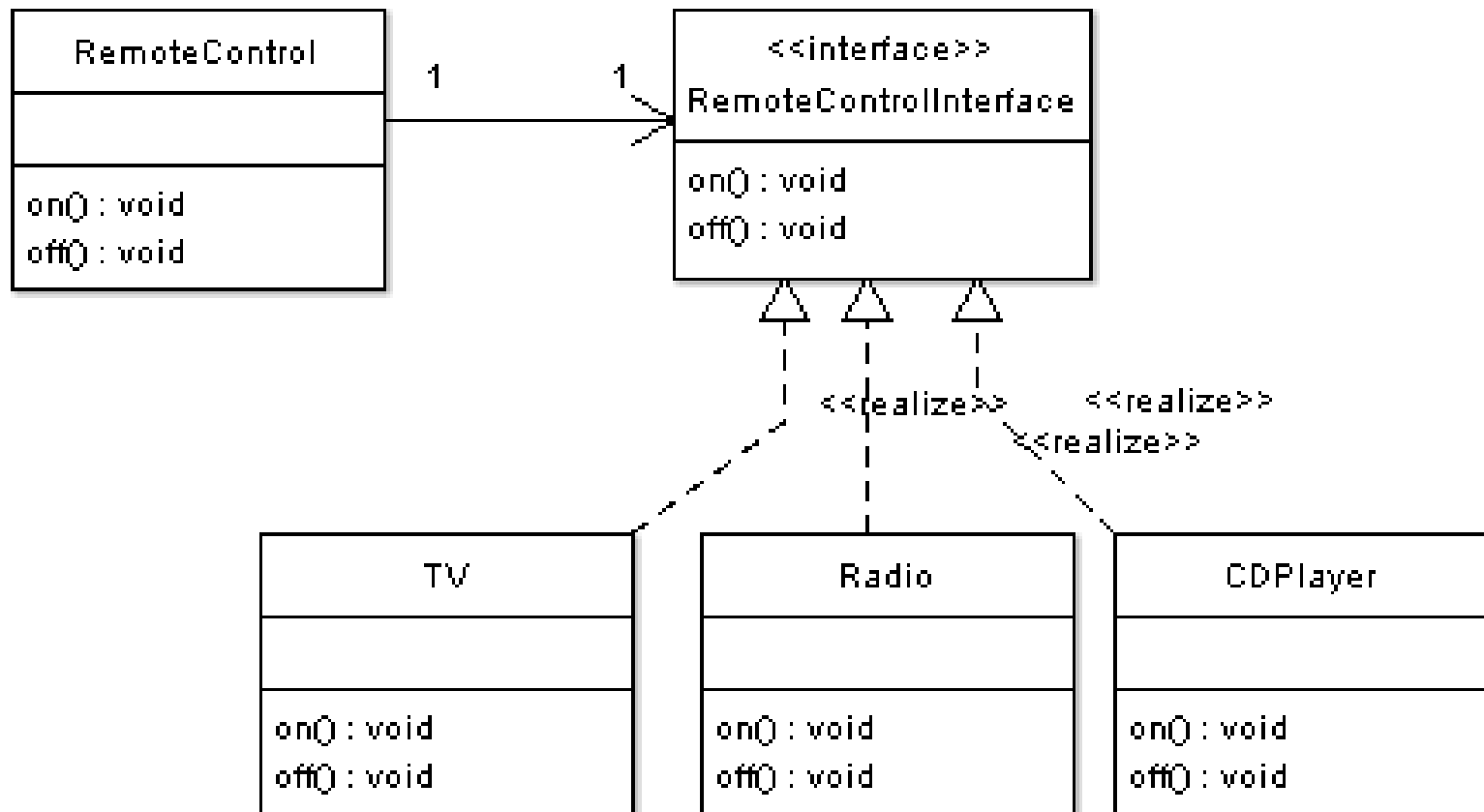
Loose Coupling (LC)



- RemoteControl tek başına var olamaz, tekrar kullanılabilirliği düşüktür.
- TV sınıfında meydana gelen değişiklikler RemoteControl sınıfını etkiler.
- RemoteControl sınıfı sadece TV sınıfı ile beraber çalışabilir.

Esnek Bağ

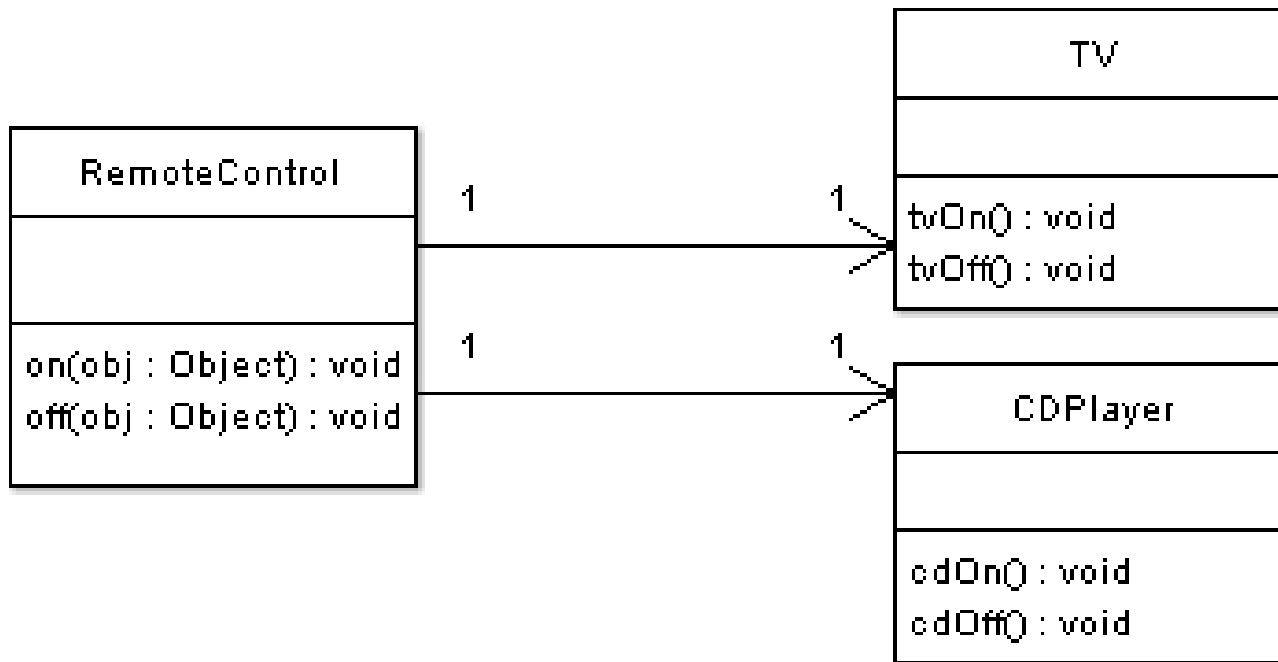
Loose Coupling (LC)



Açık Kapalı Prensibi

Open Closed Principle (OCP)

Programlar geliştirilmeye açık ama değiştirilmeye kapalı olmalıdır.



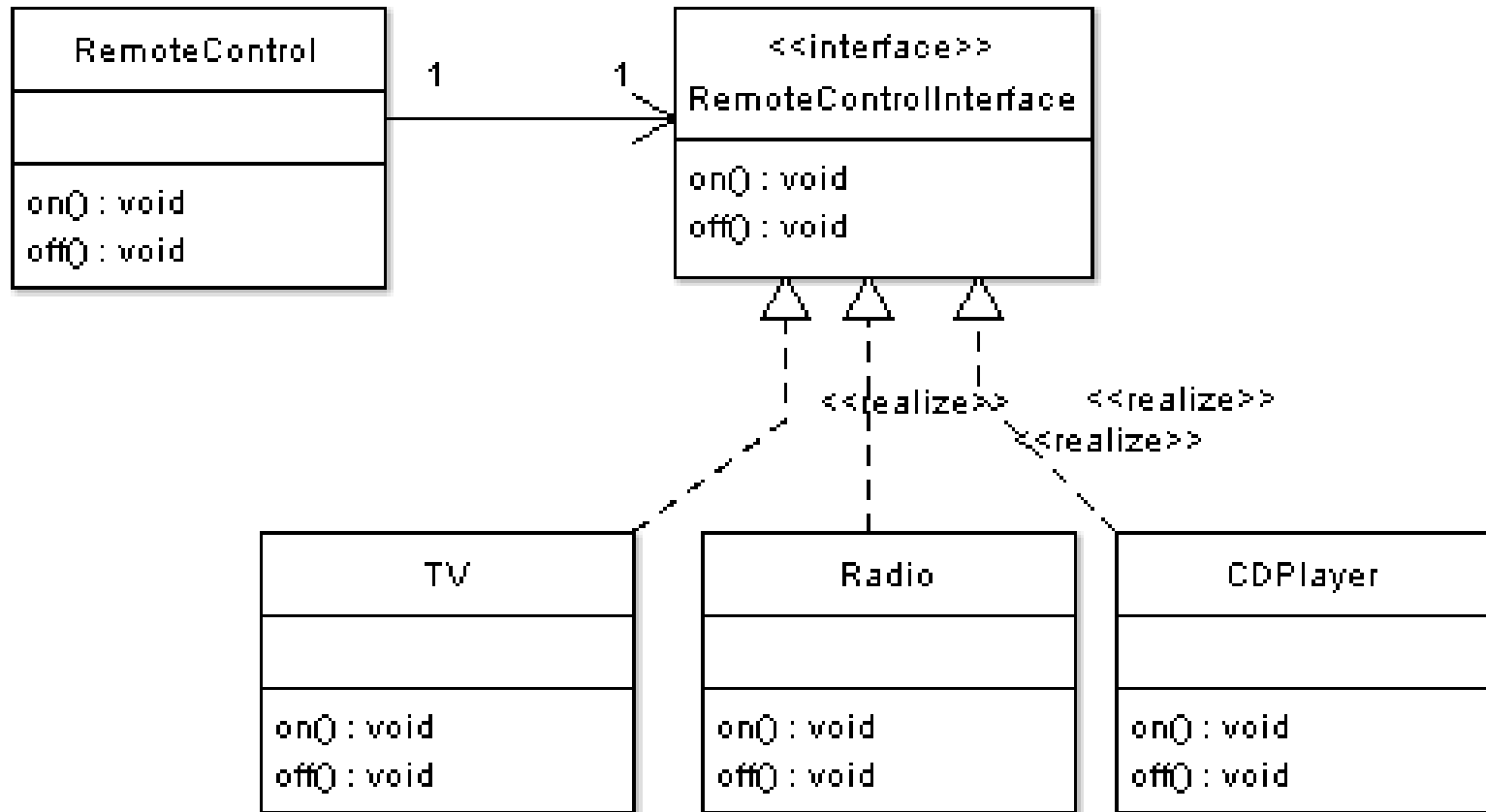
Açık Kapalı Prensibi

Open Closed Principle (OCP)

```
public void on(Object obj)
{
    if(obj instanceof TV)
    {
        ((TV) obj).tvOn();
    }
    else if(obj instanceof CDPlayer)
    {
        ((CDPlayer) obj).cdOn();
    }
}
```

Açık Kapalı Prensibi

Open Closed Principle (OCP)



Açık Kapalı Prensibi

Open Closed Principle (OCP)

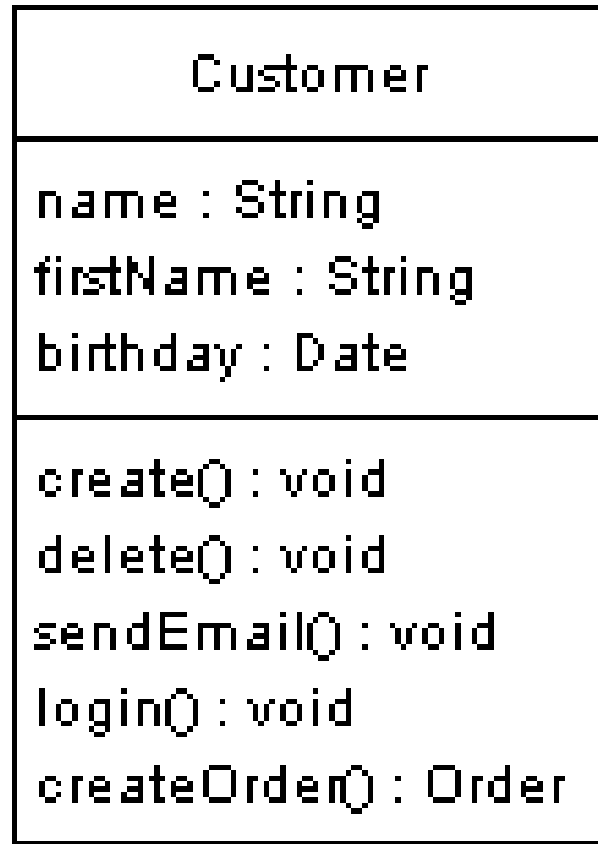
```
private RemoteControlInterface remote;  
  
public RemoteControl(RemoteControlInterface _remote)  
{  
    this.remote = _remote;  
}  
  
public void on()  
{  
    remote.on();  
}  
  
public void off()  
{  
    remote.off();  
}
```

Stratejik Kapama

- Eğer kapama tam sağlanamıyorsa, kapamanın stratejik olarak implemente edilmesi gerekir.
- Programcı implementasyon öncesi meydana gelebilecek değişiklikleri kestirerek, implemente ettiği metotların kapalılık oranını yükseltmelidir. Bu tecrübe gerektiren stratejik bir karardır.
- Programcı her zaman ne gibi değişikliklerin olabileceğini kestiremeyebilir. Bu durumda konu hakkında araştırma yaparak, oluşabilecek değişiklikleri tespit edebilir. Eğer olabilecek değişikliklerin tespiti mümkün değilse, beklenen değişiklikler meydana gelene kadar beklenir ve implementasyon yeni değişiklikleri de yansıtacak şekilde OCP uyumlu hale getirilir.

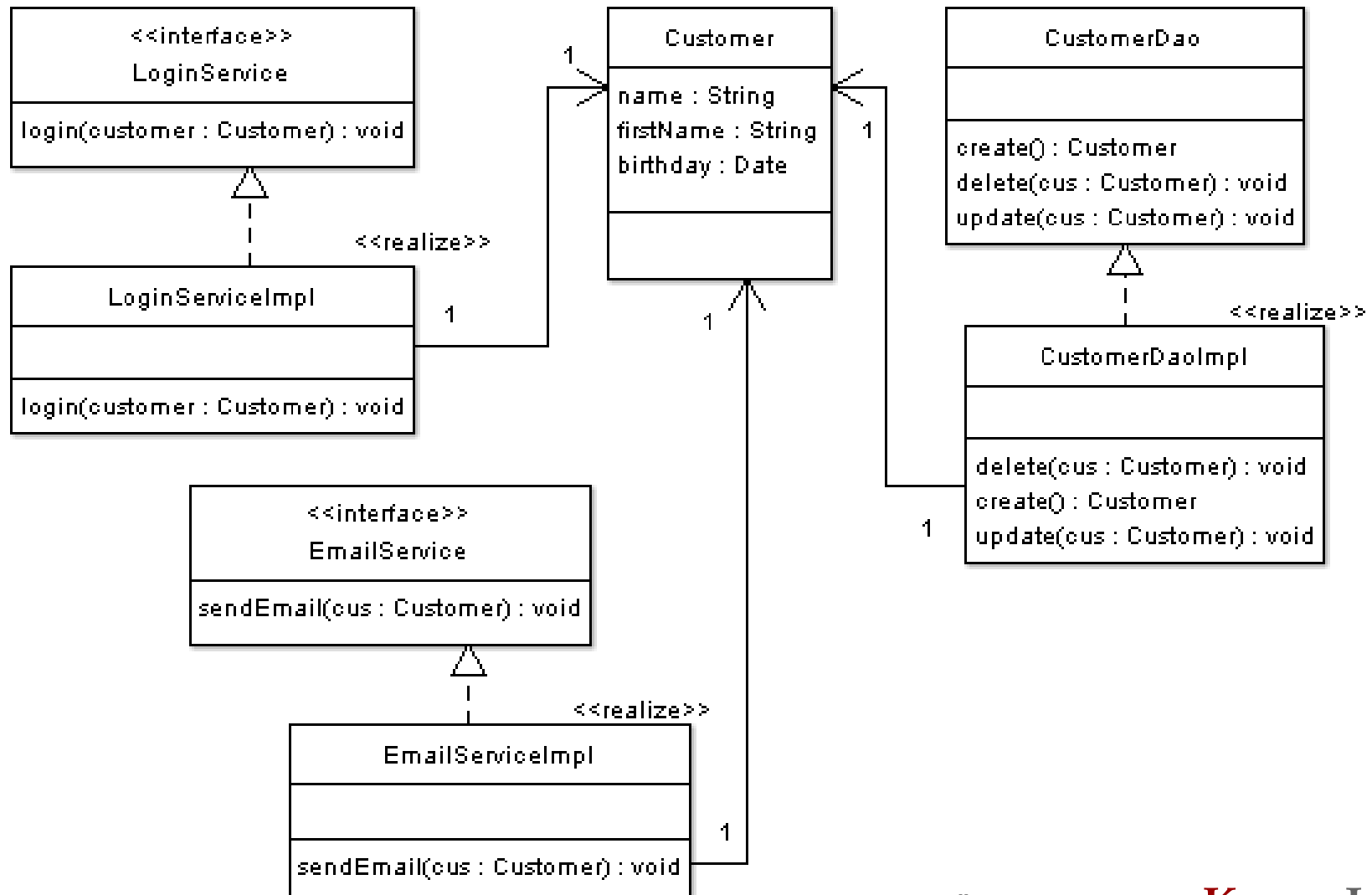
Tek Sorumluk Prensipleri

Single Responsibility Principle (SRP)



Tek Sorumluk Prensipli

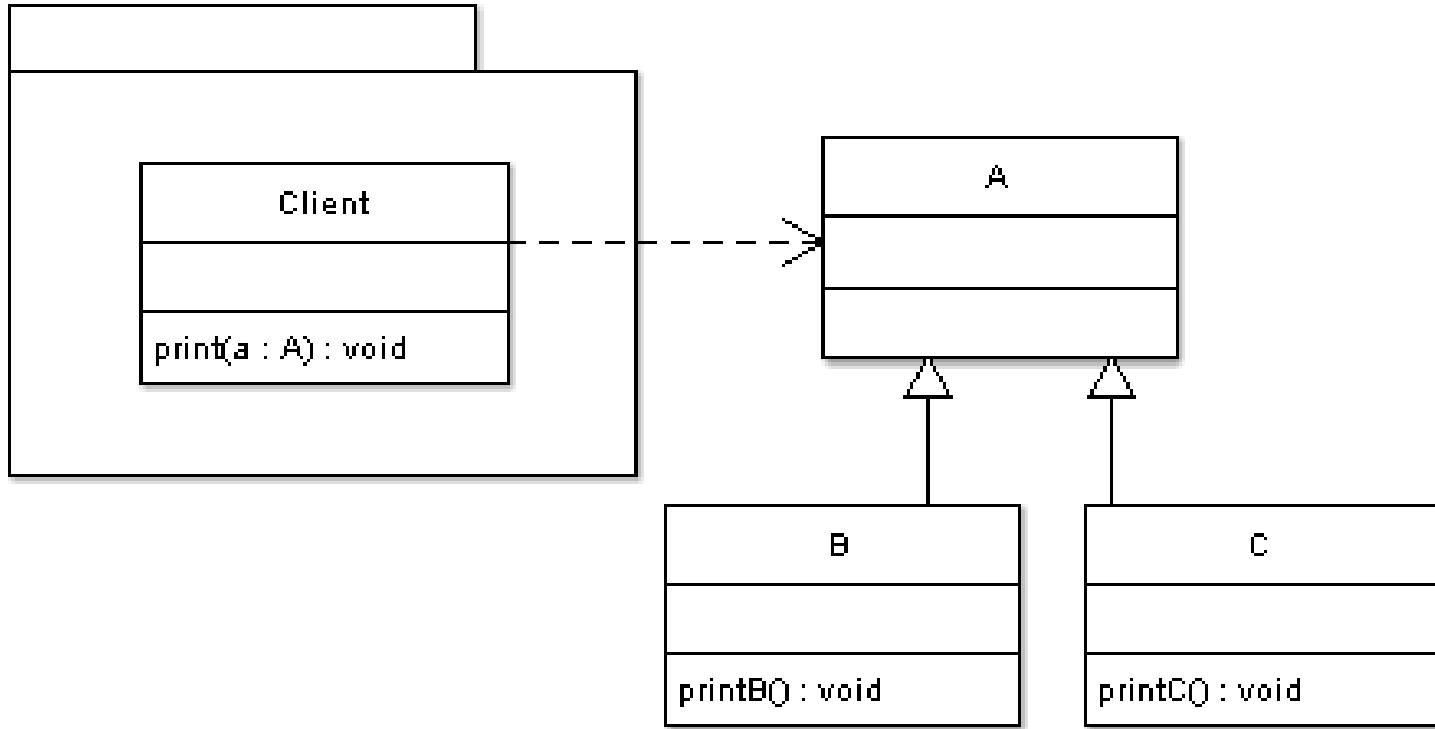
Single Responsibility Principle (SRP)



Liskov Yerine Geçme Prensipli

Liskov Substitution Principle (LSP)

Barbara Liskov: Alt sınıflardan oluşturulan nesnelere üst sınıfların nesneleriyle yer değiştirdiklerinde aynı davranışı göstermek zorundadırlar.



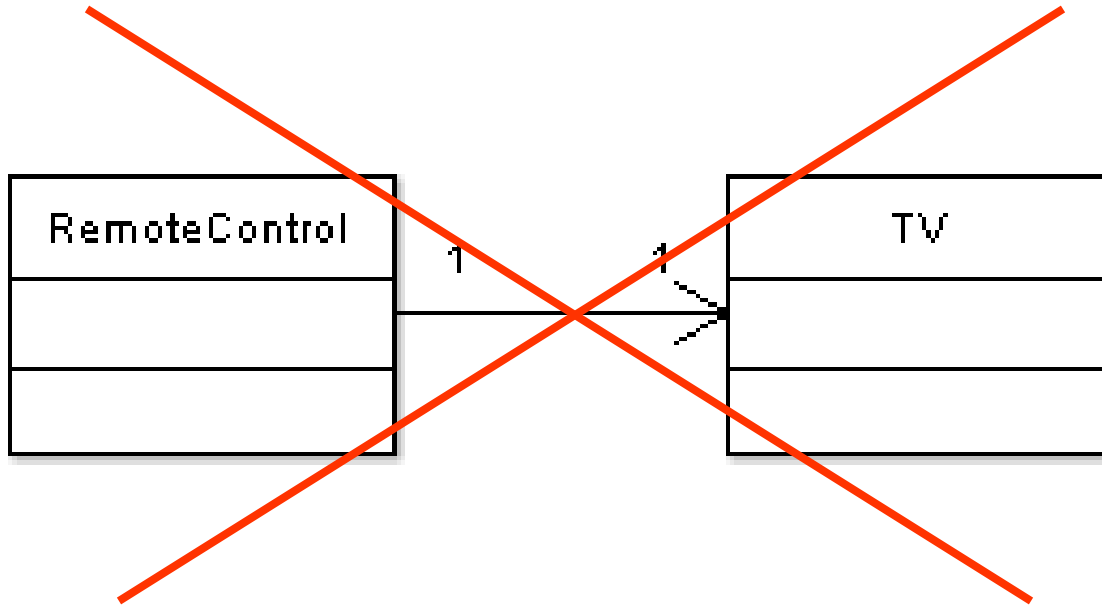
Liskov Yerine Geçme Prensipli

Liskov Substitution Principle (LSP)

```
public void print(A a)
{
    if(a instanceof B)
    {
        ((B)a).printB();
    }
    else if(a instanceof C)
    {
        ((C)a).printC();
    }
}
```

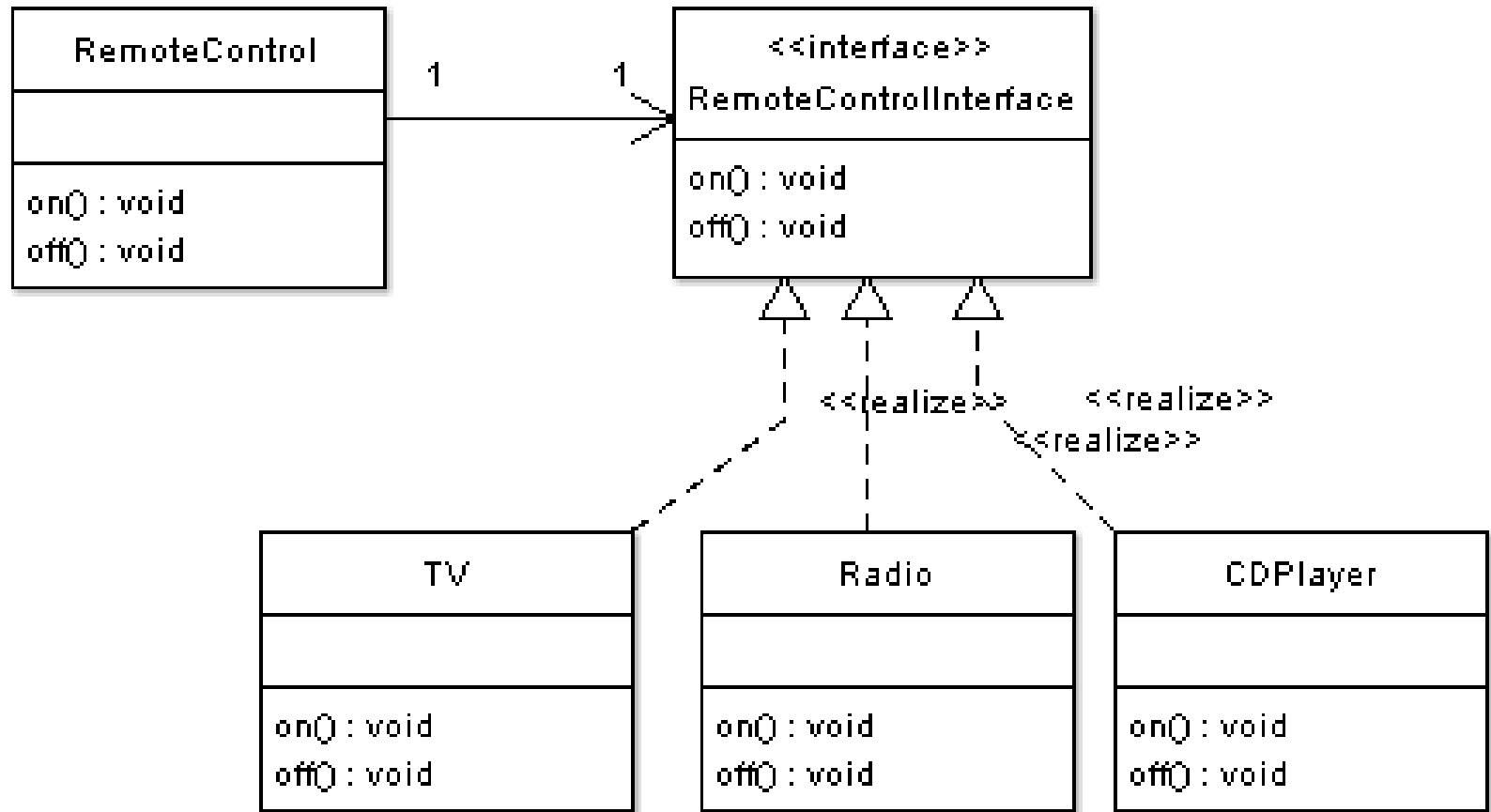
Bağımlılıkların Tersine Çevrilmesi Prensipleri

Dependency Inversion Principle (DIP)



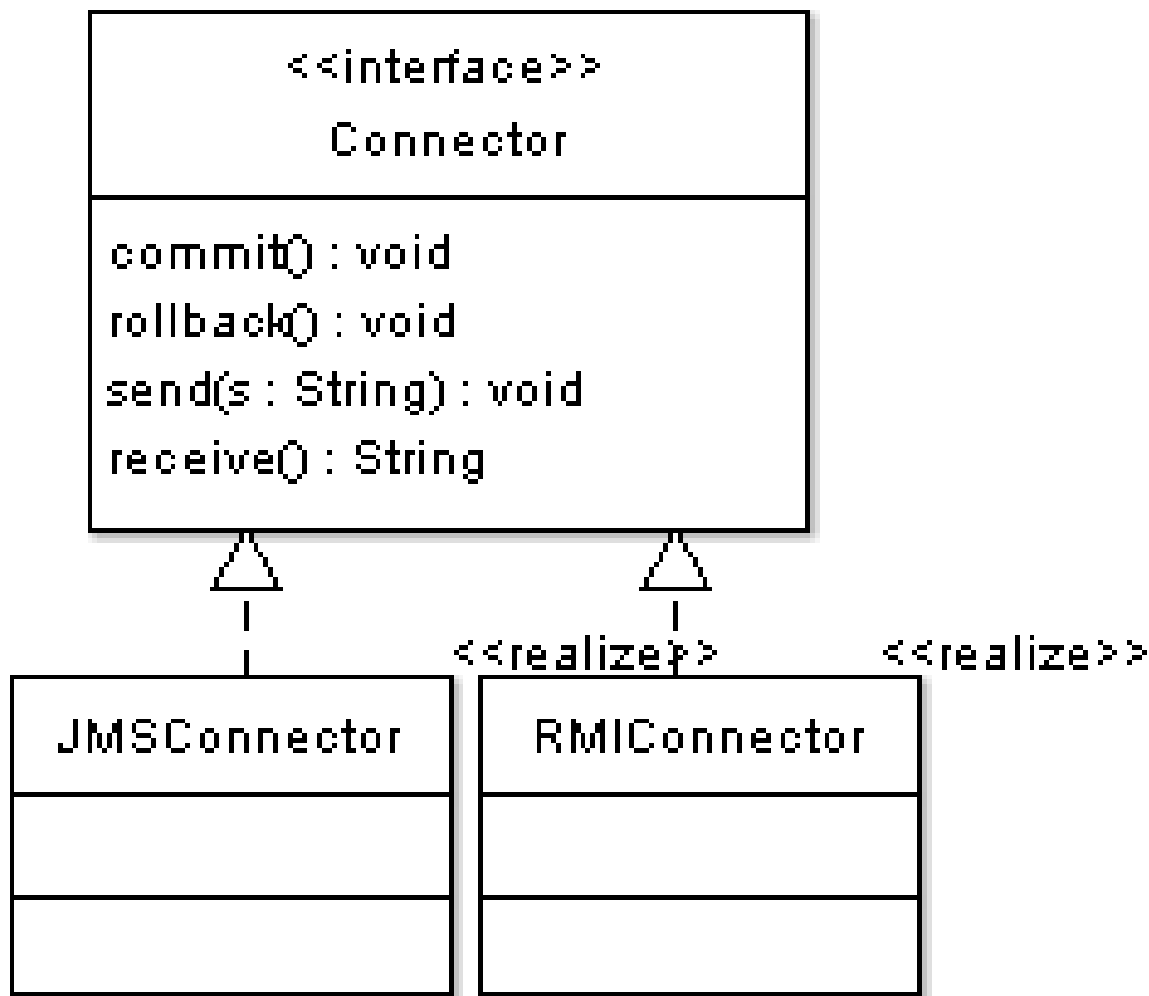
Bağımlılıkların Tersine Çevrilmesi Prensipleri

Dependency Inversion Principle (DIP)



Arayüz Ayırma Prensipli

Interface Segregation Principle (ISP)



Arayüz Ayırma Prensipli

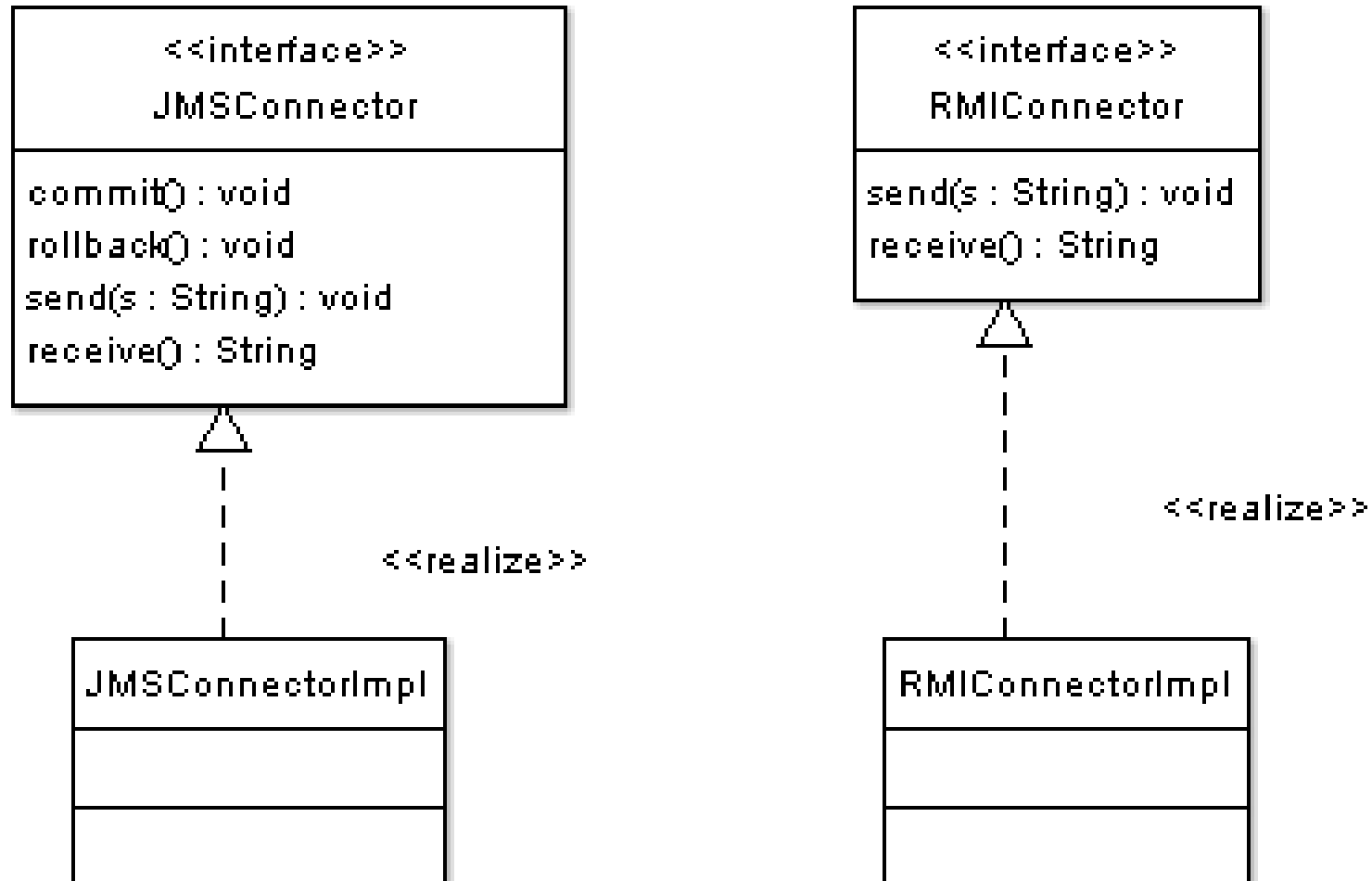
Interface Segregation Principle (ISP)

```
public class RMISeparator implements Connector
{
    public void commit()
    {
        throw new RuntimeException("not implemented");
    }

    public void rollback()
    {
        throw new RuntimeException("not implemented");
    }
}
```

Arayüz Ayırma Prensipli

Interface Segregation Principle (ISP)



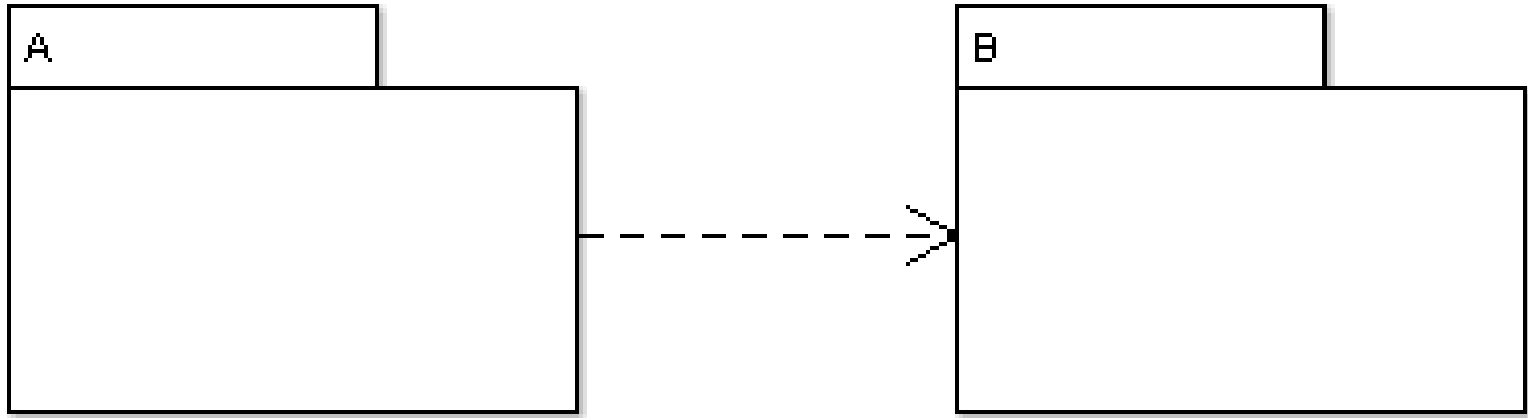
Tasarım Prensipleri

Paket Bazında

- **Reuse-Release Equivalence Principle (REP)**
- **Common Reuse Principle (CRP)**
- **Common Closure Principle (CCP)**
- **Acyclic Dependency Principle (ADP)**
- **Stable Dependencies Principle (SDP)**
- **Stable Abstractions Principle (SAP)**

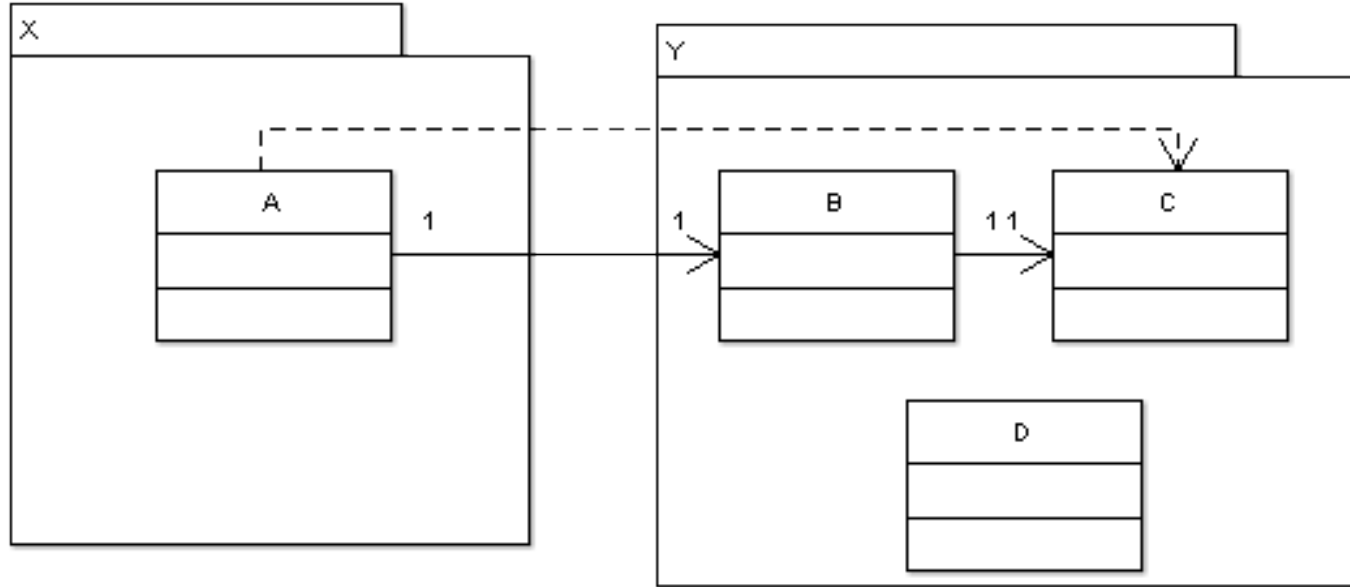
Tekrar Kullanım ve Sürüm Eşitliği

Reuse-Release Equivalence Principle (REP)



Tekrar kullanımını kolaylaştırmak için paket sürümlerinin oluşturulması şarttır. REP'e göre tekrar kullanılabilirlik (**reuse**) sürüm (**release**) ile direkt orantılıdır. Sürüm ne ihtiva ediyorsa, o tekrar kullanılabilir.

Yeniden Ortak Kullanım Prensipli Common Reuse Principle (CRP)



- Beraberce tekrar kullanılabilir yapıda olan sınıfların aynı paketi içinde yer alması gerekir.
- Y paketindeki D sınıfı değişikliğe uğradığı takdirde Y paketinin yeni bir sürümü oluşturulur. Bu durumdan, D sınıfına bağımlı olmayan A sınıfı etkilenir.

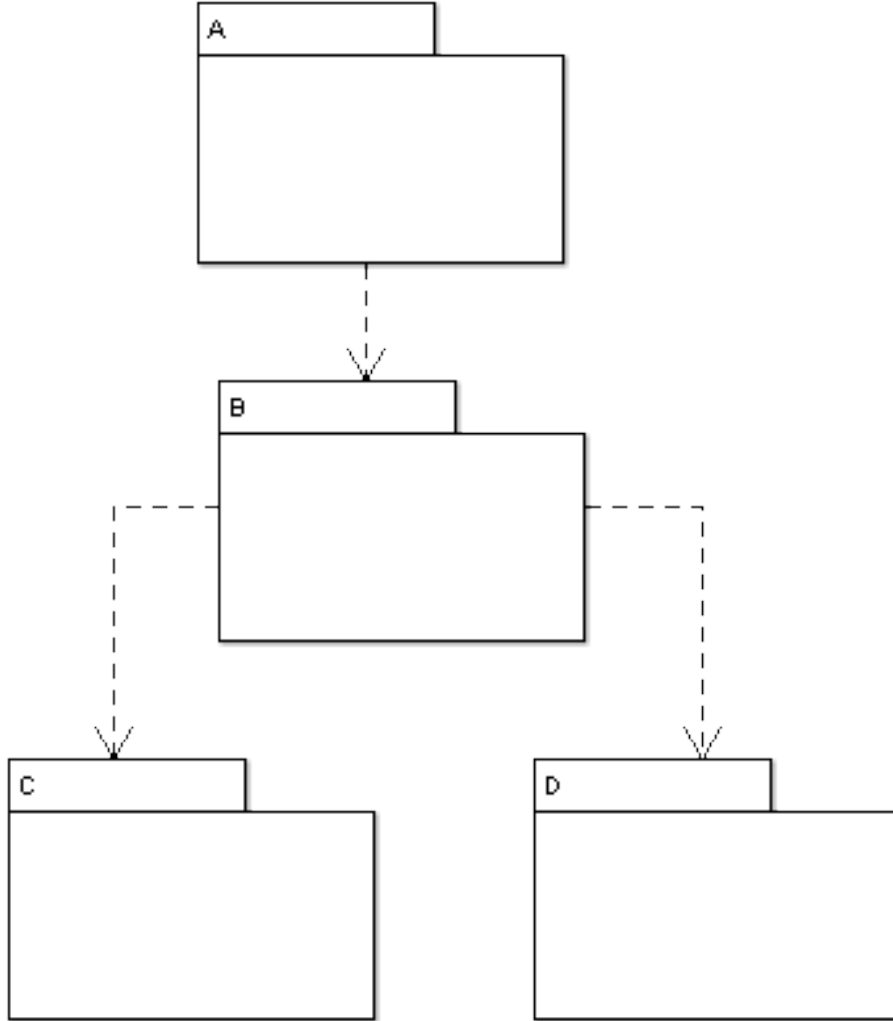
Ortak Kapama Prensipleri

Common Closure Principle (CCP)

- Aynı sebepten dolayı değişikliğe uğrayabilecek sınıfların aynı paket içinde yer alması gerekir.
- CCP daha önce incelediğimiz, sınıflar için uygulanan Single Responsibility (SRP) prensibinin paketler için uygulanan halidir.
- Her paketin değişmek için sadece bir sebebi olmalıdır.
- CCP uygulandığı takdirde sistemin bakılabilirliği artırılır ve test ve yeni sürüm için harcanan zaman ve emek azaltılır.

Döngüsüz Bağımlılık Prensipleri

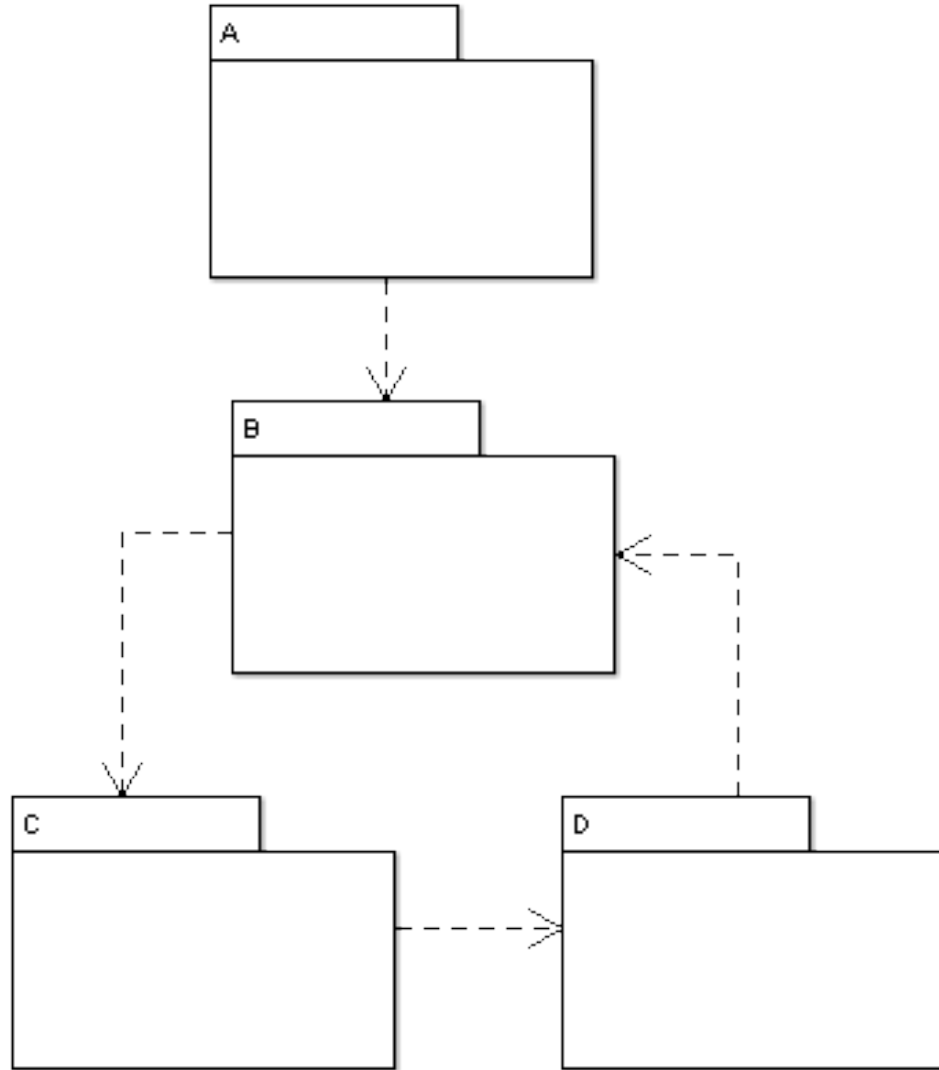
Acyclic Dependency Principle (ADP)



- Paket A, B ve dolaylı olarak C paketine bağımlıdır.
- Paket A içindeki bir sınıfı test etmek için paket C ve D'yi testlere dahil etmemiz gerekmektedir.
- C ve D paketleri diğer paketlerden bağımsız olarak test edilebilir.

Döngüsüz Bağımlılık Prensipleri

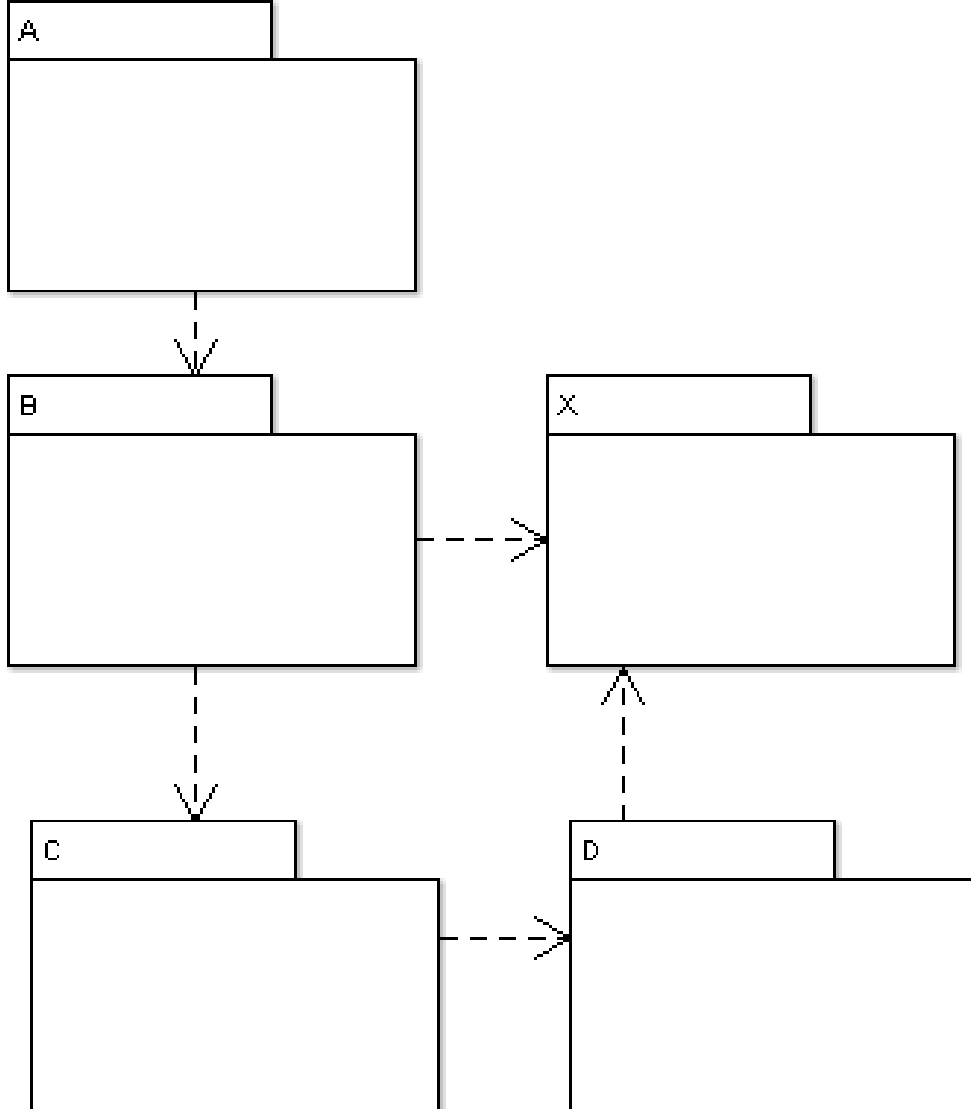
Acyclic Dependency Principle (ADP)



- B, C ve D paketleri arasında döngü mevcuttur.
- D paketini test edebilmek için B ve C paketlerine ihtiyaç duyulmaktadır.
- Döngü test edilebilirliği zorlaştırır. Oluşan bağımlılıklardan dolayı dolaylı olarak bağımlı olan paketlerin yapısal değişme riskosu artar.

Döngüsüz Bağımlılık Prensipli

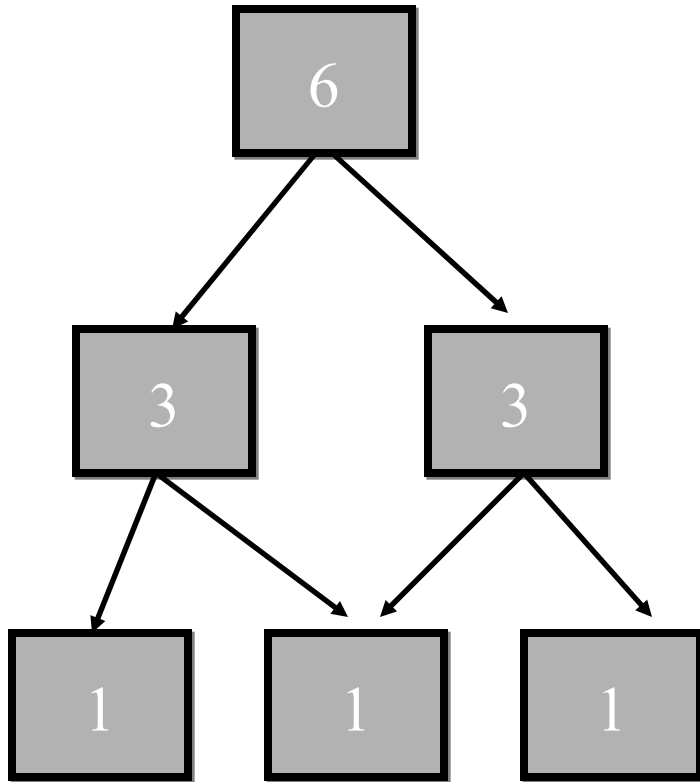
Acyclic Dependency Principle (ADP)



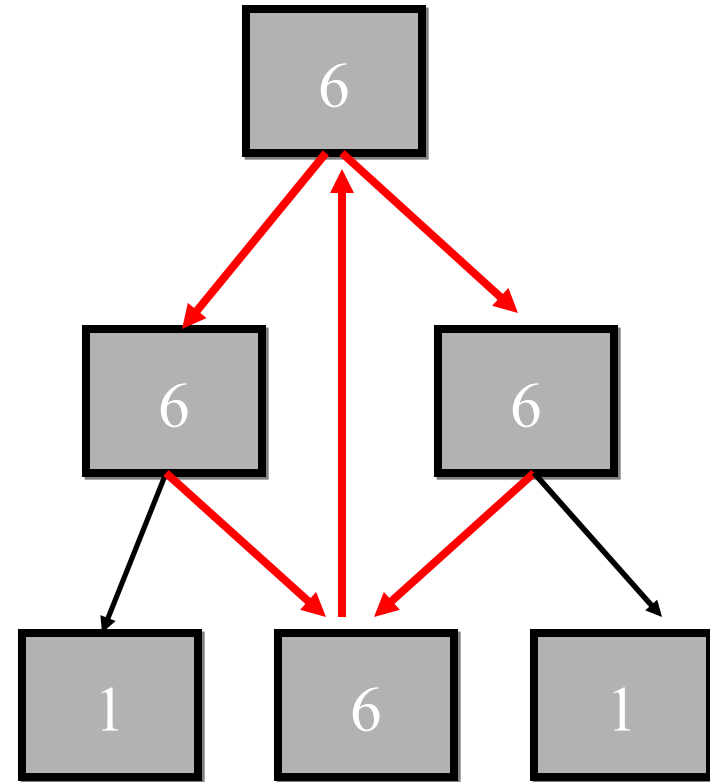
- Paketler arası döngüyü yok etmek için, paket B ve D'nin bağımlı olacağı yeni bir paket oluşturulabilir.

Ortalama Bileşen Bağımlılığı

Avarage Component Dependency (ACD)



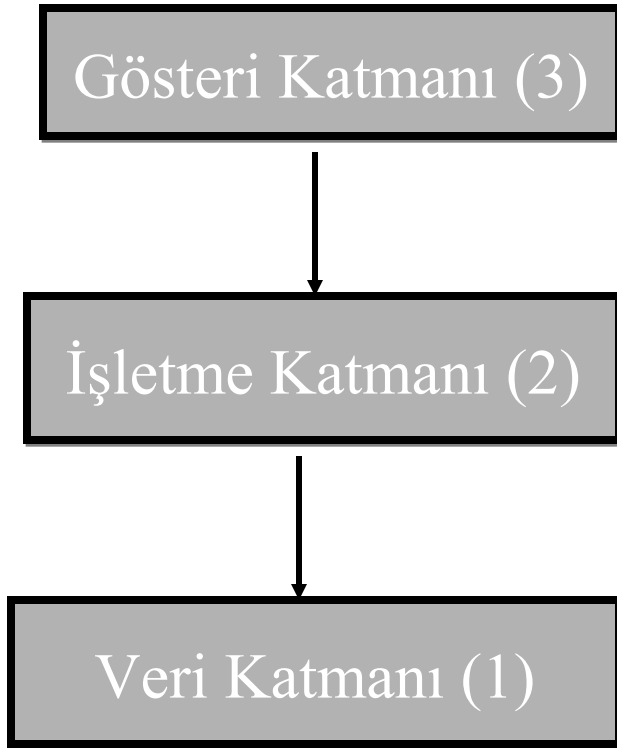
$$\text{ACD} = 15/6 = 2,5$$



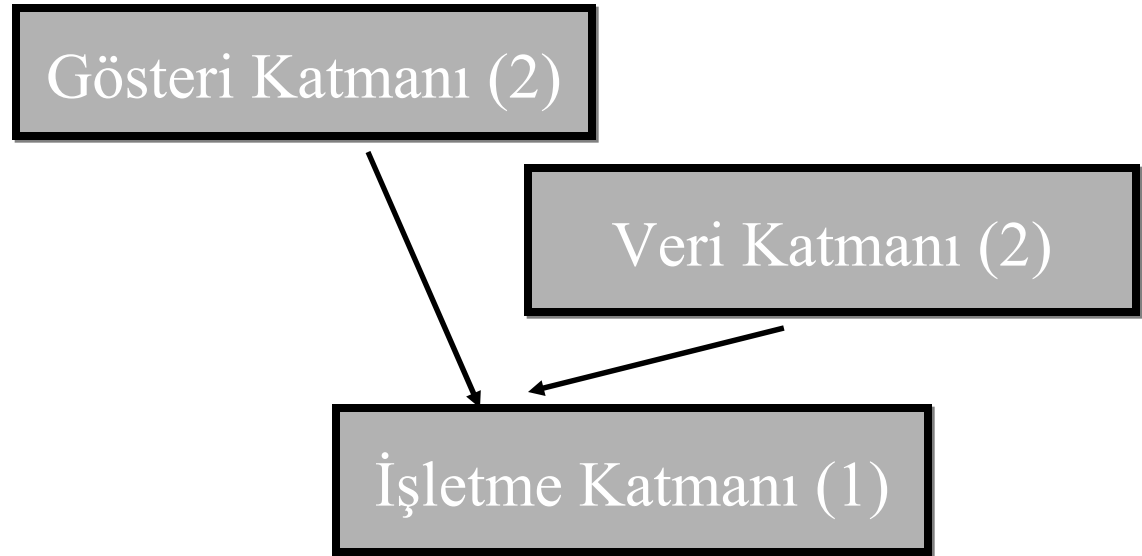
$$\text{ACD} = 26/6 = 4,33$$

Ortalama Bileşen Bağımlılığı

Average Component Dependency (ACD)



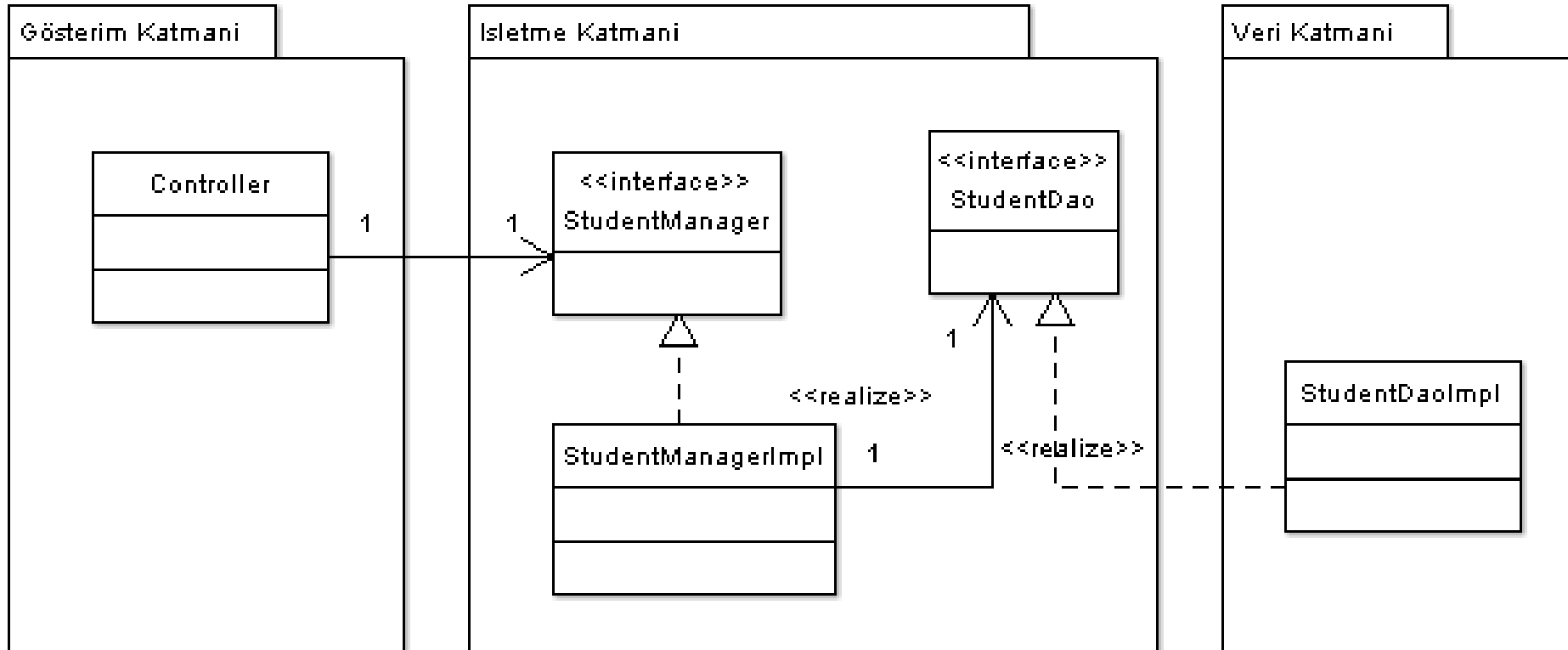
$$ACD = 6/3 = 2$$



$$ACD = 5/3 = 1,66$$

Ortalama Bileşen Bağımlılığı

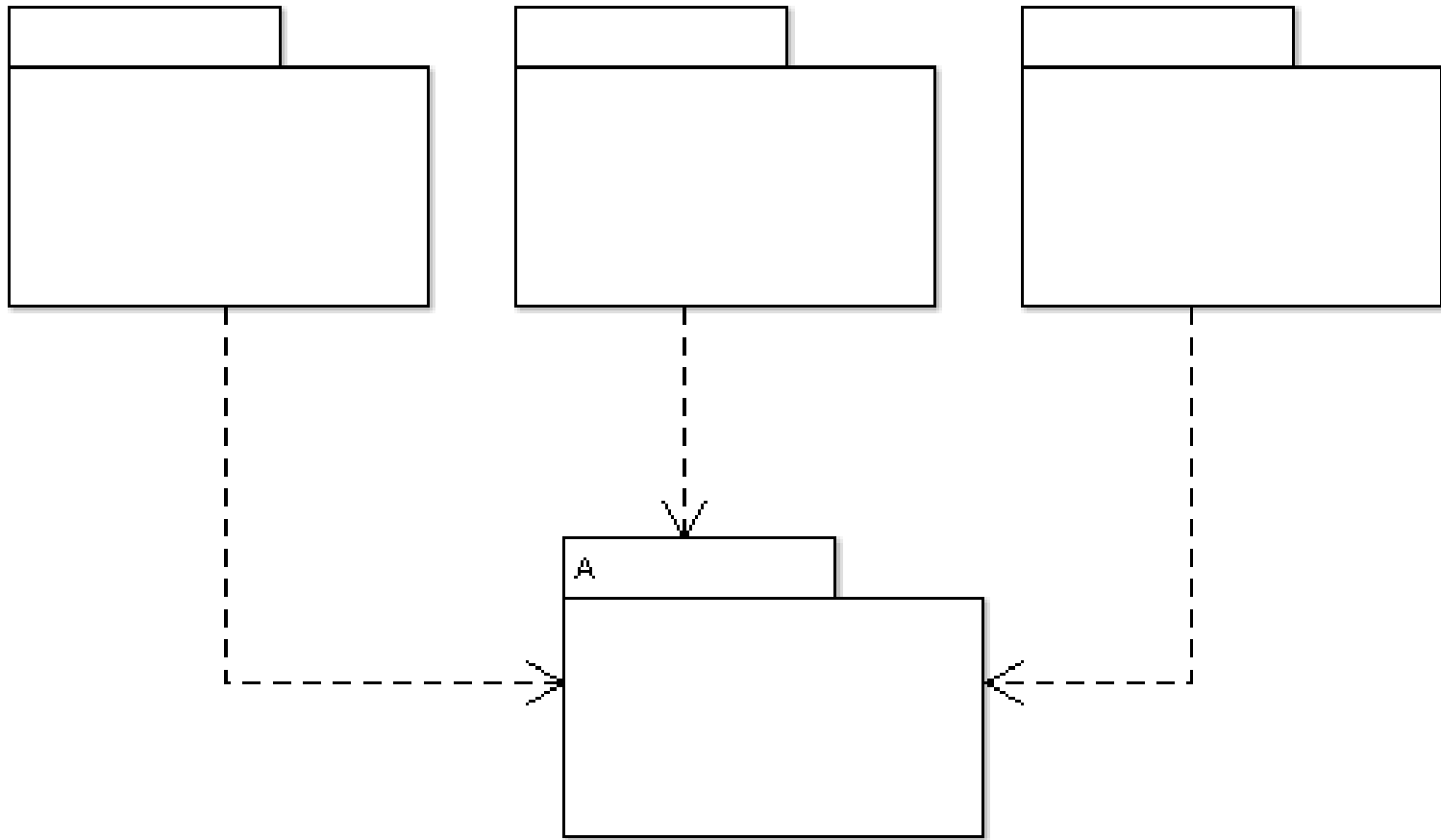
Average Component Dependency (ACD)



$$ACD = 5/3 = 1,66$$

Dayanıklı Bağımlılıklar Prensipli

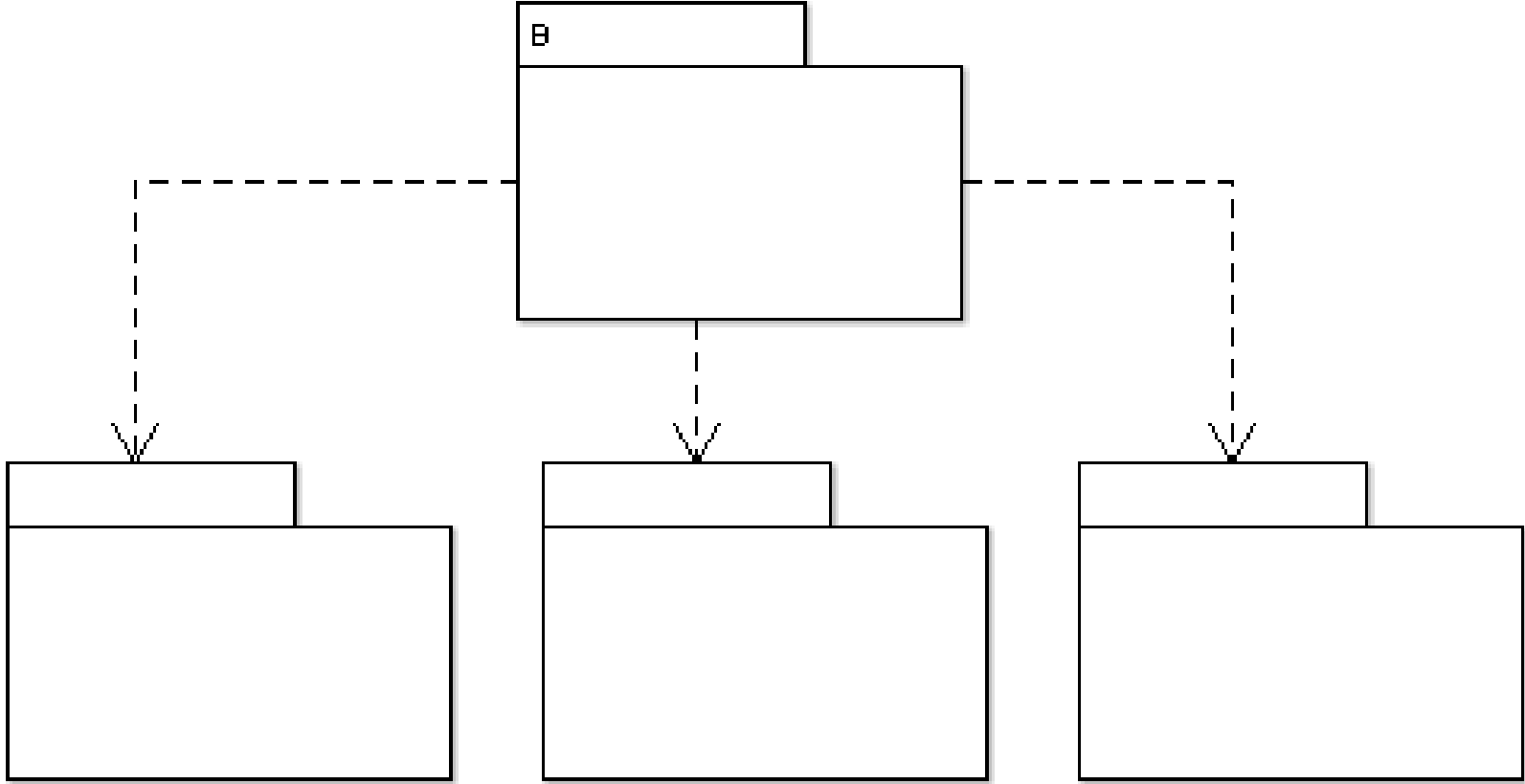
Stable Dependencies Principle (SDP)



A sorumlu (responsible) ve bağımsız (independent)

Dayanıklı Bağımlılıklar Prensipli

Stable Dependencies Principle (SDP)



B sorumsuz (irresponsible) ve bağımlı (dependent)

Dayanıklı Bağımlılıklar Prensipleri

Stable Dependencies Principle (SDP)

- SDP'ye göre bağımlılığın yönü dayanıklı paketlere doğru olmalıdır, çünkü
- dayanıklı olmayan bir pakete bağımlılık duyan bir paket doğal olarak değişikliklere maruz kalacaktır ve dayanıklılığını yitirecektir.
- Dayanıklı paketler diğer paketlere nazaran daha az değişikliğe uğrarlar.
- Bir paketin dayanıklılık oranı bağımlılık yönünü tayin eder.

Dayanıklılığın Ölçülmesi

Dayanıklılık ölçülebilir bir yazılım metriğidir.

Bu metriği elde etmek için kullanabileceğimiz iki değer vardır:

Afferent Couplings (Ca):

Paket içinde bulunan sınıflara bağımlı olan diğer paketlerin adedi

Efferent Couplings (Ce):

Paket içinde bulunan sınıfların bağımlı olduğu diğer paketlerin adedi

Dayanıklığın Ölçülmesi

$$I = \frac{C_e}{C_a + C_e}$$

I (instability) paketin dayanıklılık değerini gösteren 0 ile 1 arasında bir değerdir. 0 değerine yaklaştıkça paketin dayanıklılığı artar. 1 değerine yaklaştıkça paketin dayanıklılığı düşer.

Dayanıklı Soyutluk Prensipleri

Stable Abstractions Principle (SAP)

- Paketlerin dayanıklı olmaları sisteme yeni davranış biçimlerinin kazandırılmasının önünde bir engel olmamalıdır.
- Sisteme yeni davranış biçimlerinin soyut (abstract) ve interface sınıflar kullanarak kazandırabiliriz.
- SAP, dayanıklı olan paketlerin aynı zamanda soyut sınıflara dayandırılarak, yeniliklere açık olmaları gerektiğini söyler.
- Sadece soyut yapılar değişikliklere karşı direnç gösterebilirler.
- SAP dayanıklı olmayan paketlerde somut sınıfların bulunması gerektiğini söyler. Bu şekilde somut sınıfların değiştirilmeleri kolaylaşır.

Dayanıklı Soyutluk Prensibi

Stable Abstractions Principle (SAP)

Soyutluk ölçülebilir bir metriktir.

Bu metriği tespit etmek için kullanabileceğimiz formül şu şekildedir:

$$A = \frac{N_a}{N_c}$$

**0 – 1 arasında
bir değerdir.**

A (Abstractness):

Soyutluk oranı

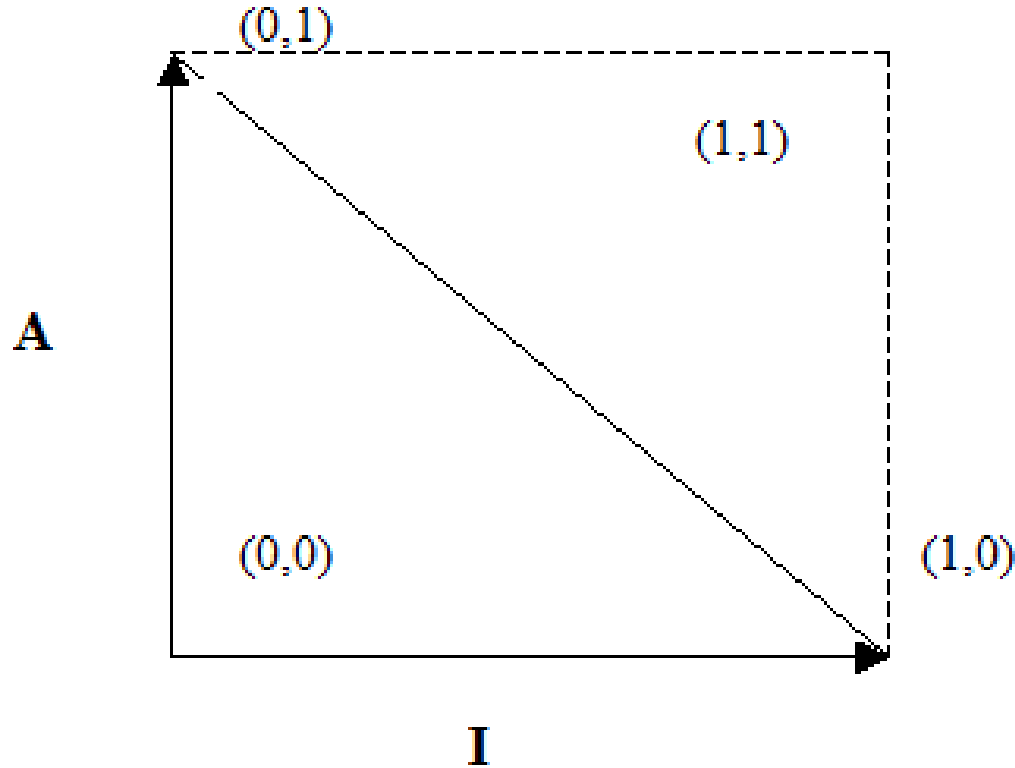
Nc:

Paket içinde bulunan sınıfların adedi

Na:

Paket içinde bulunan soyut sınıfların adedi

Soyutluk (A) ve Dayanıksızlık (I) Arasındaki İlişki



0,1: Soyut ve dayanıklı

1,0: Somut ve dayanıksız

1,1: Soyut ve bağımlılığı olmayan sınıflar. Faydasız paketler!

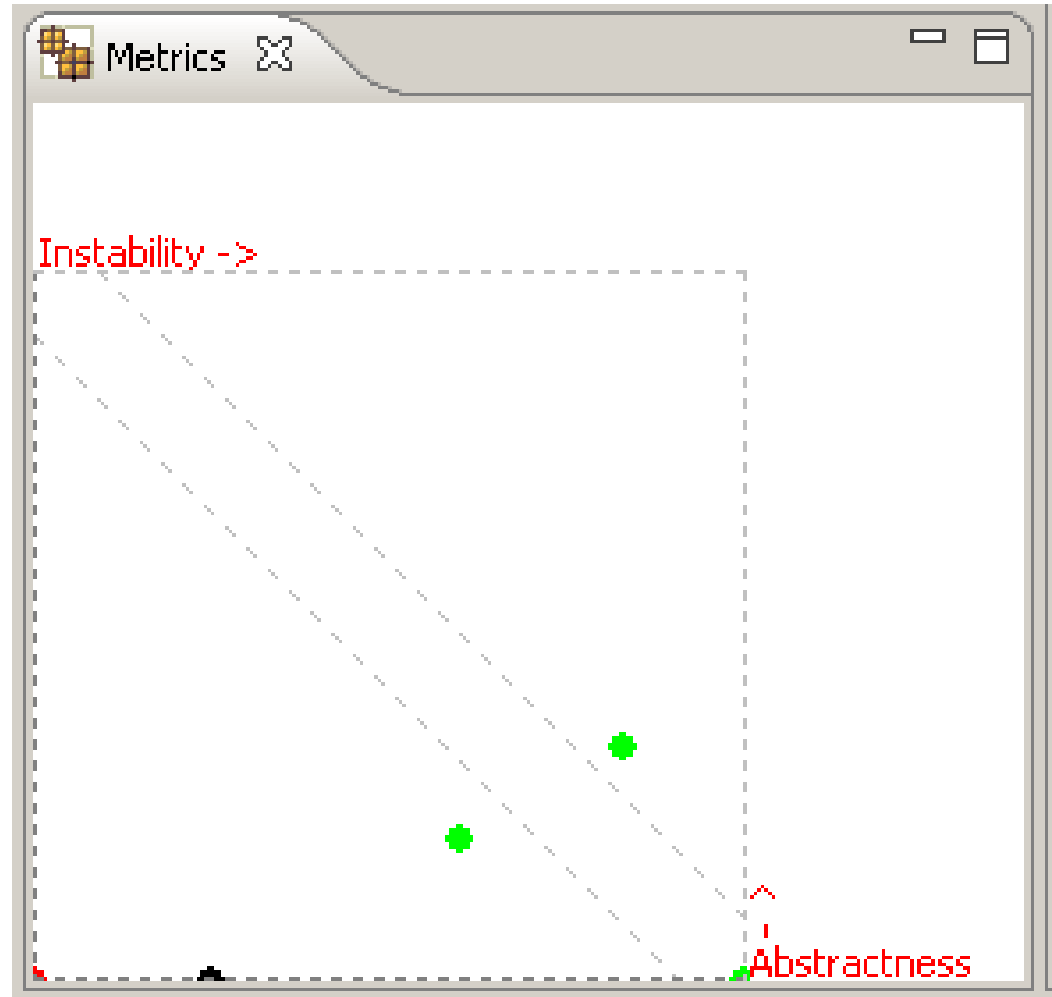
0,0: Somut ve dayanıklı. Geliştirilmesi mümkün değil, çünkü paket içinde soyut sınıf yok!

Paketler ideal şartlarda 0,1 veya 1,0 koordinatlarında olmalıdır.

Main Sequence

$$D = |A + I - 1|$$

- 0-1 arasında bir değerdir.
- 0 değeri paketin direk main sequence çizgisi üzerinde olduğunu gösterir.



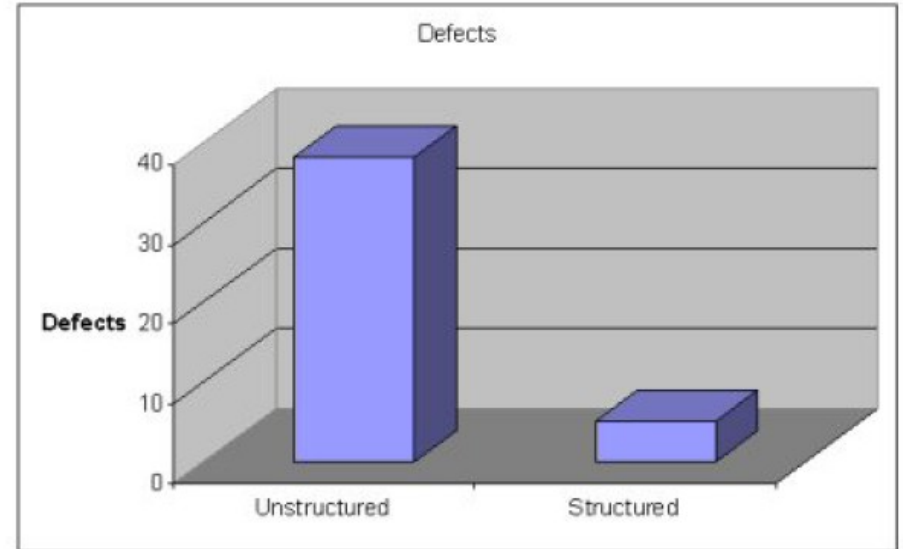
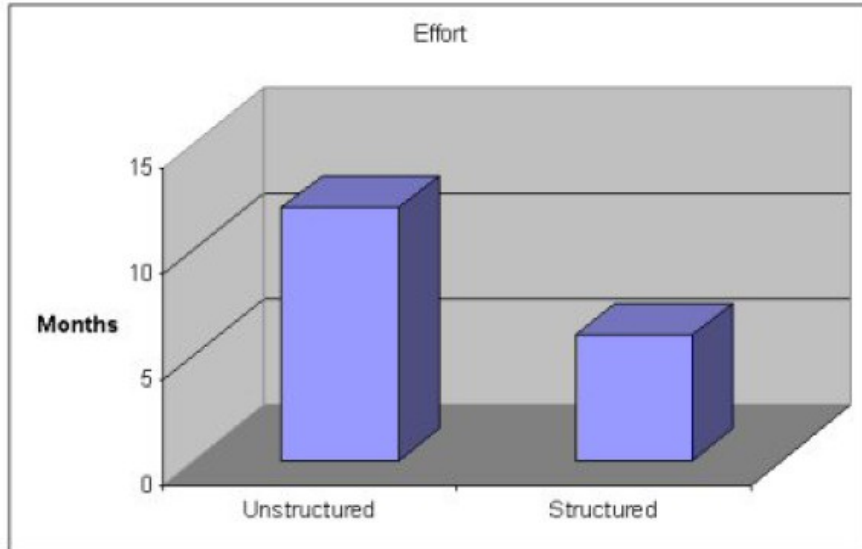
Kod Bazında Tasarım Kuralları

- En fazla 500-700 satırlık sınıflar (LOC – Lines of Code)
- Cyclomatic complexity < 15
- Paketler arası döngü olmamalı
- ACD değeri düşük tutulmalı (%10)
- Mümkün mertebe bağımlılık soyut sınıflara doğru olmalı (interface sınıf kullanımı)
- Kalıtım hiyerarşileri derin olmamalı, kalıtım yerine kompozisyon kullanılmalı
- Projelerde günlük metrik istatistik raporları oluşturulmalı

Değişikliğin Maliyeti

Software Technology Support Center (STSC) tarafından yapılan bir araştırma

- 50 bin satırlık bir programa 3 bin satırlık ekleme yapılacak
- 1. ekip tasarım değişikliği yapılmadan çalışmalarını tamamlıyor
- 2. ekip iyileştirilmiş ve döngüsüz bir tasarım ile çalışmalarını tamamlıyor



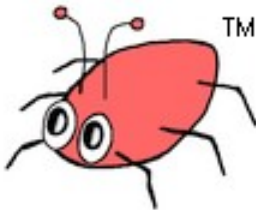
Tasarım Kontrol Araçları



Checkstyle



JDepend



Sun Coding Style Violations - Microsoft Internet Explorer

Adresse C:\workspace\Shop\build\checkstyle-report\checkstyle_report.html

Coding Style Check Results

Summary	
Total files checked	42
Files with errors	24
Total errors	99
Errors per file	2

The following are violations of the Sun Coding:

File: C:\workspace\Shop\src\shop\business\

FindBugs Report - Microsoft Internet Explorer

Adresse C:\workspace\Shop\build\findbugs-report\findbugs-report.html

FindBugs Report

Project Information

Project: <<unnamed project>>
FindBugs version: 1.3.0
Code analyzed: shop\build\classes

EMMA Coverage Report (generated Wed Sep 10 10:02:00 EEST 2008) - Microsoft Internet Explorer

Adresse C:\workspace\Shop\build\emma-report\coverage.html

EMMA Coverage Report (generated Wed Sep 10 10:02:00 EEST 2008)

OVERALL COVERAGE SUMMARY

name	class, %	method, %	block, %
all classes	88% (7/8)	76% (25/33)	90% (355/395)

OVERALL STATS SUMMARY

total packages: 4
total executable files: 8
total classes: 8
total methods: 33
total executable lines: 104

COVERAGE BREAKDOWN BY PACKAGE

name	class, %	method, %	block, %
shop_business_login	75% (3/4)	59% (10/17)	71% (90/127)
shop_presentation_login_controller	100% (1/1)	75% (3/4)	98% (174/177)
shop_domain	100% (1/1)	100% (5/5)	100% (17/17)
shop_persistence_login	100% (2/2)	100% (7/7)	100% (74/74)

[all classes]
EMMA 2.0.5312 (C) Vladimir Roubtsov

PMD - Microsoft Internet Explorer

Adresse C:\workspace\Shop\build\pmd-report\pmd_report.html

PMD report

Problems found

#	File	Line	Problem
1	C:\workspace\Shop\src\shop\business\login\LoginManagerException.java	7	Avoid variables with short names like e
2	C:\workspace\Shop\src\shop\business\login\LoginManagerImpl.java	15	Parameter 'email' is not assigned and could be declared final
3	C:\workspace\Shop\src\shop\business\login\LoginManagerImpl.java	15	Parameter 'password' is not assigned and could be declared final
4	C:\workspace\Shop\src\shop\business\login\LoginManagerImpl.java	17	Local variable 'result' could be declared final
5	C:\workspace\Shop\src\shop\business\login\LoginManagerImpl.java	20	Local variable 'daoResult' could be declared final
6	C:\workspace\Shop\src\shop\business\login\LoginManagerImpl.java	40	Parameter 'daoResult' is not assigned and could be declared final
7	C:\workspace\Shop\src\shop\business\login\LoginManagerImpl.java	41	Parameter 'result' is not assigned and could be declared final
8	C:\workspace\Shop\src\shop\business\login\LoginManagerImpl.java	56	Parameter 'dao' is not assigned and could be declared final

Test Edilebilir Tasarım

1

Kalıtım (inheritance) yerine kompozisyon kullanılmalıdır.

Test Edilebilir Tasarım

2

Static metot ve Singleton yapılar kullanılmamalıdır.

```
public class PdfCreatorTest extends TestCase
{
    public void testPdfCreator()
    {
        assertTrue(PdfCreator.create());
    }
}
```

Test Edilebilir Tasarım

3

Bağımlılıkların isole edilmesi gerekir.

```
public class PdfCreator
{
    public Pdf create()
    {
        PdfCreatorService service =
            PdfCreatorService.getInstance();
        ...
    }
}
```

Test Edilebilir Tasarım

```
public class PdfCreator
{
    public Pdf create()
    {
        PdfCreatorService service = getService();
        ...
    }

    protected PdfCreatorService getService()
    {
        return PdfCreatorService.getInstance();
    }
}
```

Test Edilebilir Tasarım

```
public class PdfCreatorTest extends TestCase
{
    public void testCreatePdf()
    {
        PdfCreator creator = new PdfCreator()
        {
            protected PdfCreatorService getService()
            {
                return new DummyPdfCreatorService();
            }
        };

        assertNotNull(creator.create());
    }
}
```

Test Edilebilir Tasarım

4

Bağımlılıkların enjekte (injection) edilmesi, testleri kolaylaştırır.

Test Edilebilir Tasarım

```
public class PdfCreator
{
    private PdfCreatorService service;

    public Pdf create()
    {
        PdfCreatorService service = getService();
        ...
    }

    public PdfCreatorService getService()
    {
        return service;
    }

    public void setService(PdfCreatorService service)
    {
        this.service = service;
    }
}
```

Test Edilebilir Tasarım

```
public class PdfCreatorTest extends TestCase
{
    public void testCreatePdf()
    {
        PdfCreator creator = new PdfCreator();
        DummyPdfCreatorService service =
            new DummyPdfCreatorService();
        creator.setService(service);
        assertNotNull(creator.create());
    }
}
```

Kaynak: KurumsalJava.com



The screenshot shows a web browser window displaying the KurumsalJava.com website. The page title is "Extreme Programming | Kurumsal Java Yazılımı - Windows Internet Explorer". The address bar shows "http://www.kurumsaljava.com/category/xp/". The website header features the logo "Kurumsal Java" with the tagline "Java Enterprise Architecture". Below the logo is a navigation menu with links: ANA SAYFA, FORUM, HAKKIMIZDA, İLETİŞİM, KİTAPLARIMIZ, MAKALELER, OKUYUCUDAN, PROJELER, SUNUMLAR, TAVSİYE. The main content area is titled "Extreme Programming" and contains a blog post titled "Çevik Sürece Geciş Nasıl Olmalı?". The post is dated "Şub 13th, 2009" and is by "acar". The text discusses the popularity of Agile processes like Scrum and Extreme Programming, comparing them to the Waterfall model. It mentions that Agile processes are becoming more popular due to the need for faster development and delivery. The post also includes tags for "Extreme Programming" and a link to "No Comments". Below the post is another article titled "Türkiye'nin İlk Extreme Programming Konulu Kitabı" dated "Şub 8th, 2009" by "acar". The text mentions that a new book on Extreme Programming is being published. The right sidebar contains sections for "ONLINE ÜYELER" (2 kullanıcı Online, acar, 1 ziyaretçi), "KURUMSAL JAVA YAZIŞMA GRUBU" (Google Grup adresi: http://groups.google.com/group/kurumsaljava), "E-POSTA ABONELİĞİ" (Abonelik için email adresinizi giriniz), and "KATEGORİLER" (Extreme Programming (9), Haberler (13)).

Kaynak: KurumsalJava.com

Tasarım Prensipleri:

<http://www.kurumsaljava.com/download/7/>

Test Edilebilir Tasarım:

<http://www.kurumsaljava.com/download/11/>

KurumsalJavaAkademisi.com

Kurumsal Java Akademisi » Kurumsal Java Akademisi - Windows Internet Explorer

http://www.kurumsaljavaakademisi.com/

Kurumsal Java Akademisi » Kurumsal Java Akademisi

Kurumsal Java Akademisi

Java Enterprise Architecture

[Ana Sayfa](#) [Eğitim](#) [Eğitmenlerimiz](#) [Hakkımızda](#) [İletişim](#) [Kitaplarımız](#) [KurumsalJava.com](#)

[Düzenlediğimiz Seminerler](#) [Yeni Seminerler](#)

DÜZENLEDİĞİMİZ SEMİNERLER

Çankaya Üniversitesi Wicket Semineri



Kurumsal Java Akademisi eğitmenlerinden Özcan Acar 3 ocak 2009 tarihinde Ankara Çankaya üniversitesinde CETURK tarafından düzenlenen Java teknolojileri seminerine konuşmacı olarak katıldı ve Apache Wicket framework hakkında bir sunum yaptı. Sunum dosyalarını aşağıdaki linkler üzerinden edinebilirsiniz.

[Devamı...]

Son

İlginiz için teşekkür ederim.