Erciyes Üniversitesi
Bilgisayar Mühendisliği Bölümü

BZ 313 Yazılım Mühendisliği

16. Program Tasarımında Modeller

Program Geliştirme Yaklaşımları

Heavyweight yaklaşım

Program tasarımı ve kodlaması ayrıdır.

Tasarımda, kodlamaya başlamadan önce programın ayrıntılı olarak belirtilmesi için sınıf modelleri ve diğer modeller kullanılır. UML bu amaç için iyi bir modelleme dilidir.

Lightweight yaklaşım

Program tasarımı ve kodlaması iç içe geçmiştir.

Geliştirme, Tümleşik geliştirme ortamı (Integrated development environment, IDE) kullanılarak yinelemelidir.

Karma yaklaşım

Anahat tasarımı, modeller kullanılarak oluşturulur ve ayrıntılar kodlama sırasında yinelemeli olarak işlenir.

Program Tasarımı

Program tasarımının görevi, yazılım mimarisini bir veya daha fazla çalıştırılabilir program olarak uygulanabilecek bir biçimde temsil etmektir.

Verilen bir sistem mimarisindeki program tasarımında şunlar belirtilir:

- programlar, bileşenler, paketler, sınıflar, sınıf hiyerarşileri vb.
- arayüzler, protokoller (sistem mimarisinin bir parçası olmadığı durumlarda)
- algoritmalar, veri yapıları, güvenlik mekanizmaları, operasyonel prosedürler

UML Modelleri

UML modelleri (**diyagramlar** ve **spesifikasyonlar**) program tasarımının hemen hemen tüm yönleri için kullanılabilir.

- **Diyagramlar**, temel unsurları ve birbirleriyle nasıl ilişkili olduklarını gösteren tasarıma genel bir bakış sunar.
- **Spesifikasyonlar,** tasarımın her bir öğesi hakkında ayrıntılı bilgi sağlar.

Heavyweight geliştirme süreçlerinde, spesifikasyonlar, kod yazmak için kullanılabilecek yeterli ayrıntıya sahip olmalıdır. Amaç, kodlama başlamadan önce tüm spesifikasyonu tamamlamaktır.

UML Modelleri

Temelde gereksinimler için kullanılan modeller

- Use-case diyagramı, bir dizi use-case'i ile aktörü ve bunların ilişkilerini gösterir.

Temelde sistem mimarisi için kullanılan modeller

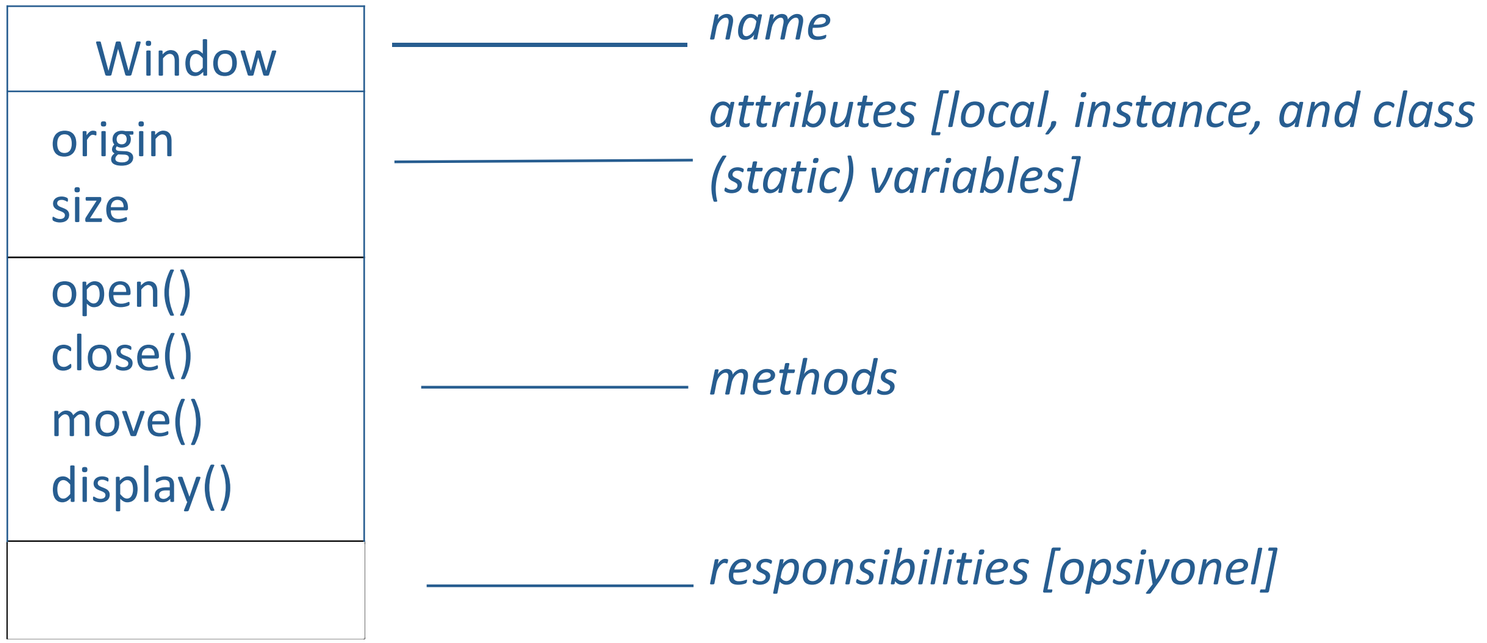
- Bileşen (component) diyagramı, bir bileşen kümesi arasındaki organizasyonu ve bağımlılıkları gösterir.
- Dağıtım (deployment) diyagramı, işlem düğümlerinin yapılandırmasını ve üzerlerinde bulunan bileşenleri gösterir.

Temelde program tasarımı için kullanılan modeller

- Sınıf (class) diyagramı, ilişkileriyle birlikte bir dizi sınıfı, arayüzü ve işbirliğini gösterir.
- Nesne (object) diyagramı veya sıralı (sequence) diyagram, bir nesne kümesini ve bunların ilişkilerini gösterir.

Class Diyagramı

Sınıf, aynı nitelikleri, yöntemleri, ilişkileri ve semantiği paylaşan bir nesne kümesinin açıklamasıdır.



Terminoloji hakkında not. Bu derste bir sınıfın desteklediği yöntemler için metot (method) terimi kullanılmaktadır. UML bunun için işlemler (operations) terimini kullanmaktadır.

"Hello, World!" Applet

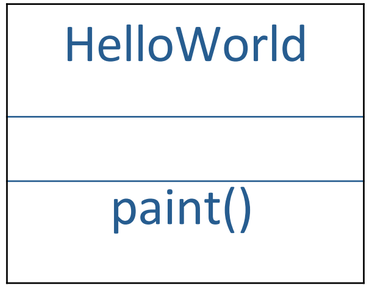
```
import java.awt.Graphics;
class HelloWorld extends java.applet.Applet {
    public void paint (Graphics g) {
        g.drawString ("Hello, World!", 10, 20);
    }
}
```

HelloWorld Class'ı

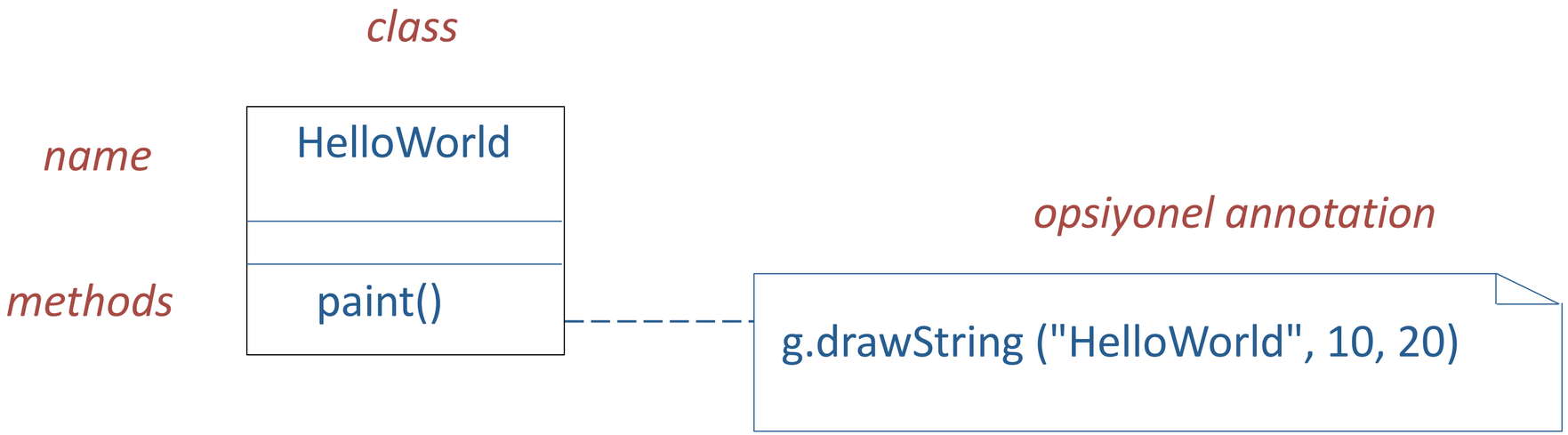
class

name

methods



HelloWorld Class'ı



Gösterim (notation): İlişkiler

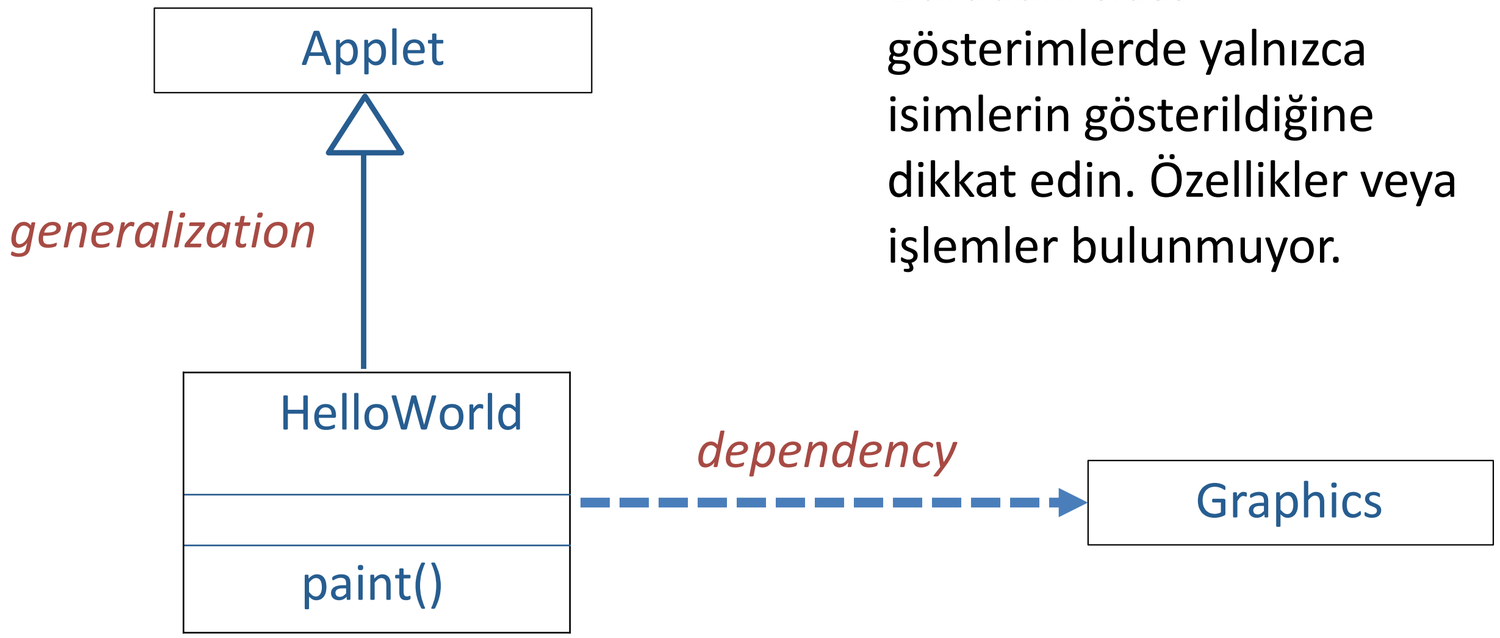
Bağımlılık (dependency), iki eleman arasında var olan ve herhangi bir değişikliğin her ikisini de etkilediği ilişki türüdür.

child  *parent*

Genelleme (generalization) genellikle kalıtımı göstermek için kullanılır. ‘IS-A’ ilişkisini tanımlarken kullanılır. Child nesne parent’tan miras almıştır.

Gerçekleştirme (realization, implementation) ile interface ve sınıflar arasındaki ilişkiyi modellemek için kullanılır.

HelloWorld Sınıfı

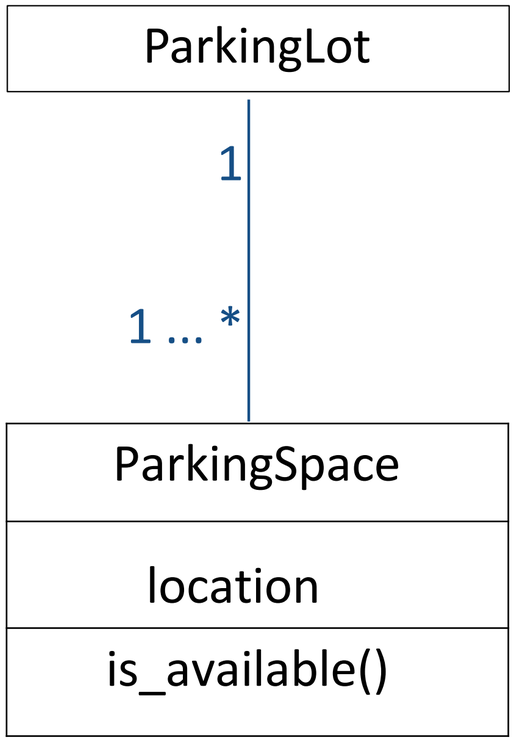


Buradaki class gösterimlerde yalnızca isimlerin gösterildiğine dikkat edin. Özellikler veya işlemler bulunmuyor.

Notation: Association

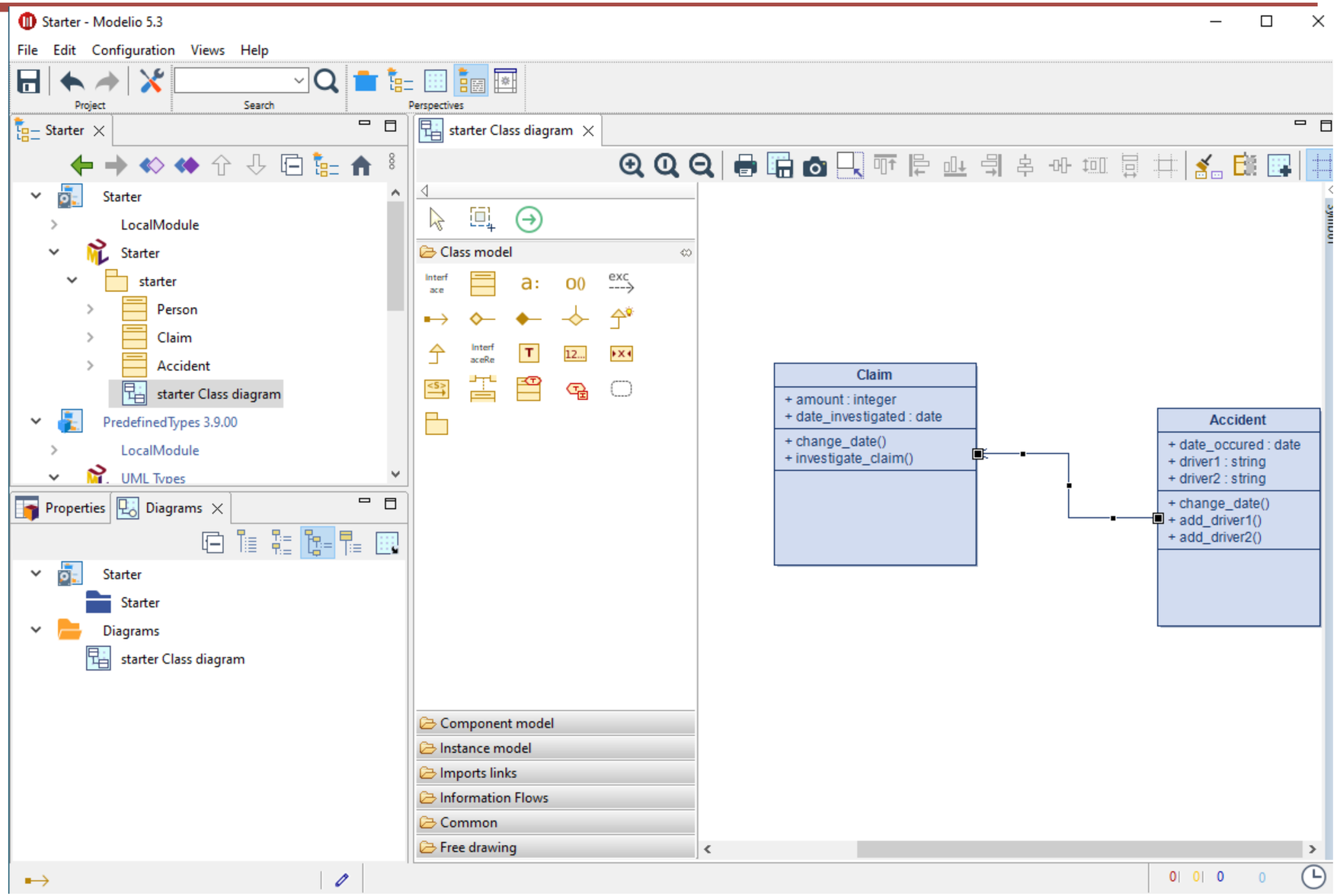
Association (ilişkilendirme) bir dizi bağlantıyı tanımlayan yapısal bir ilişkidir; bağlantı, nesneler arasındaki bağlantıdır.

Association



Modelio: Tipik Bir Diyagram

https://github.com/ModelioOpenSource



Hangi Sınıfların Kullanılacağına Karar Verme

Gerçek hayattaki bir sistem göz önüne alındığında, hangi sınıfları kullanacağınıza nasıl karar verirsiniz?

Adım 1. Sistem tasarımını temsil eden bir dizi aday sınıf belirleyin.

- Kullanıcılar ve uygulayıcılar sistemi tanımlamak için hangi terimleri kullanıyor? Bu terimler sınıflar için adaylardır.
- Her aday sınıf net bir şekilde tanımlanmış mı?
- Her sınıf için sorumlulukları nelerdir? Sorumluluklar sınıflar arasında eşit olarak dengeleniyor mu?
- Her sınıfın sorumluluklarını yerine getirmek için hangi niteliklere ve yöntemlere ihtiyacı vardır?

Hangi Sınıfların Kullanılacağına Karar Verme

Adım 2. Sınıf kümesini düzenleme

Hedefler:

- Tasarımın netliğini artırın

  Her sınıfın amacı açıksa, kolay anlaşılır yöntemler ve ilişkilerle, geliştiricilerin gelecekteki geliştiricilerin anlayabileceği ve değiştirebileceği basit kod yazması muhtemeldir.

- Sınıflar arasındaki tutarlılığı artırın ve sınıflar arasındaki bağlantıyı (coupling) azaltın.

Sınıflar arasında yüksek uyumu (cohesion) ve aralarında zayıf bağlantıyı (coupling) hedefleyin.

Uygulama Sınıfları ve Çözüm Sınıfları

İyi bir tasarım genellike uygulama (application) sınıfları ile çözüm (solution) sınıflarının birleşimidir.

- **Uygulama sınıfları,** uygulama kavramlarını temsil eder.

  İsim tanımlama, aday uygulama sınıfları oluşturmak için etkili bir tekniktir.

- **Çözüm sınıfları,** kullanıcı arayüzleri nesneleri, veritabanları vb. gibi sistem kavramlarını temsil eder.

İsim Tanımlama: Kütüphane Örneği

Kütüphanede kitaplar ve dergiler bulunmaktadır. Belirli bir kitabın birkaç kopyası olabilir. Kitaplardan bazıları yalnızca kısa vadeli ödünç verme için ayrılmıştır. Diğerleri herhangi bir kütüphane üyesi tarafından üç hafta süreyle ödünç alınabilir.

Kütüphane üyeleri bir seferde en fazla altı materyal ödünç alabilir ancak personel üyeleri bir seferde en fazla 12 materyal ödünç alabilir. Dergileri sadece personel ödünç alabilir.

Sistem, kitapların ve dergilerin ne zaman ödünç alındığını ve iade edildiğini takip etmeli ve kuralları uygulamalıdır.

İsim Tanımlama: Kütüphane Örneği

Kütüphanede kitaplar ve dergiler bulunmaktadır. Belirli bir kitabın birkaç kopyası olabilir. Kitaplardan bazıları yalnızca kısa vadeli ödünç verme için ayrılmıştır. Diğerleri herhangi bir kütüphane üyesi tarafından üç hafta süreyle ödünç alınabilir.

Kütüphane üyeleri bir seferde en fazla altı materyal ödünç alabilir ancak personel üyeleri bir seferde en fazla 12 materyal ödünç alabilir. Dergileri sadece personel ödünç alabilir.

Sistem, kitapların ve dergilerin ne zaman ödünç alındığını ve iade edildiğini takip etmeli ve kuralları uygulamalıdır.

Aday Sınıflar

İsim	Yorum	Aday
Kütüphane	Sistemin adı	Hayır
Kitap		Evet
Dergi		Evet
Kopya		Evet
KısaVadeliÖdünç	Eylem	Hayır (?)
Kütüphane Üyesi		Evet
Hafta	Ölçüm	Hayır
Kütüphane Üyeleri	Kütüphane Üyesi tekrarı	Hayır
Materyal	Kitap veya Dergi	Evet (?)
Personel		Evet
Sistem	Genel terim	Hayır
Kurallar	Genel terim	Hayır

Sınıflar Arası İlişkiler

Kitap	is an	Materyal
Dergi	is an	Materyal
Kopya	is a copy of a	Kitap
KütaphaneÜyesi		
Materyal		
Personel	is a	KütüphaneÜyesi

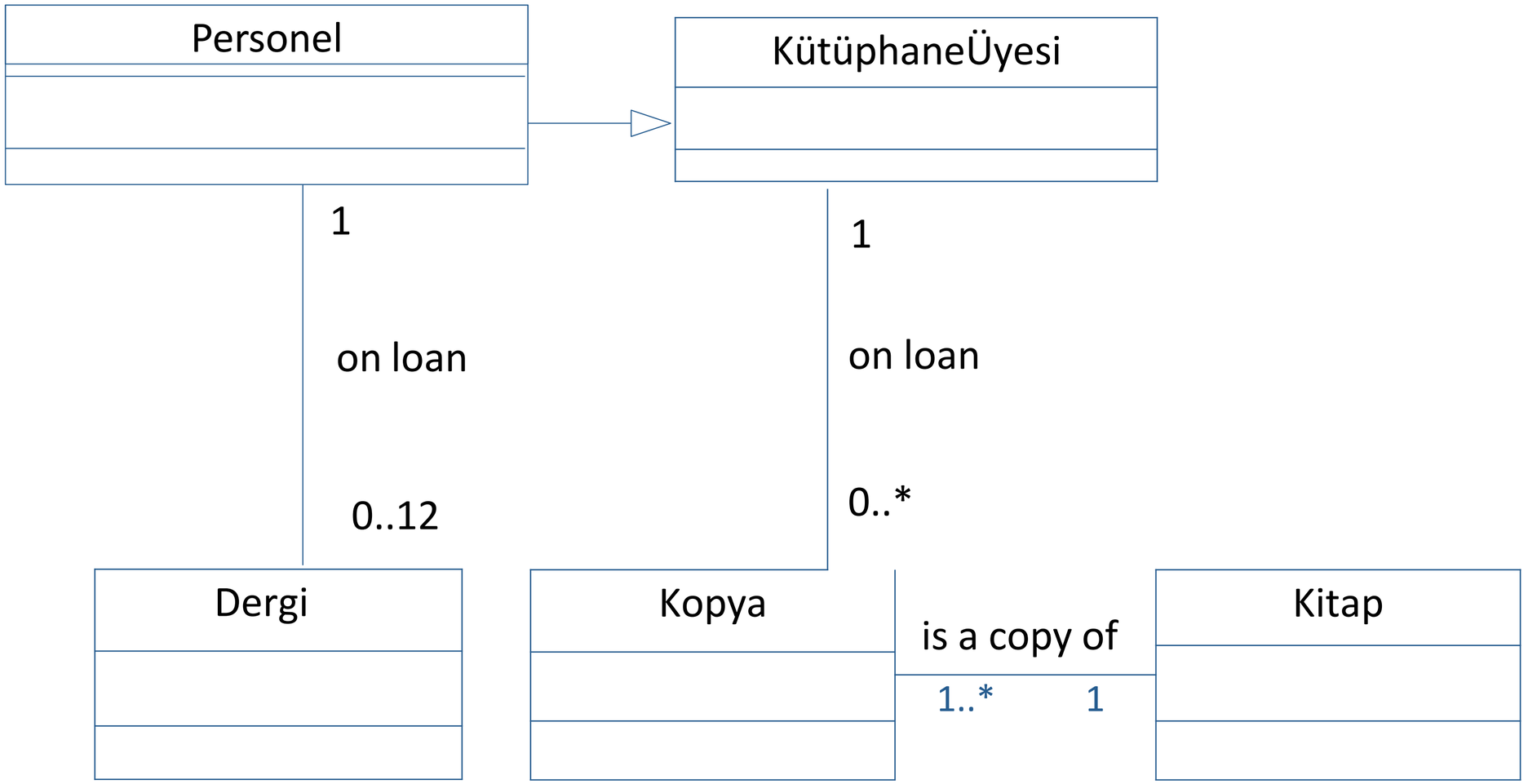
Materyal gerekli mi?

Metotlar

KütüphaneÜyesi	Ödünç alır	Kopya
KütüphaneÜyesi	İade eder	Kopya
Personel	Ödünç alır	Dergi
Personel	İade eder	Dergi

Materyal hala gerekli değil.

Olası Bir Sınıf Diyagramı



Aday Sınıflardan Tamamlanmış Tasarıma

Nihai tasarıma geçmek için kullanılan yöntemler

Yeniden kullanın: Mümkün olan her yerde varolan bileşenleri veya sınıf kütüphanelerini kullanın. Uzantılara ihtiyaç duyabilirler.

Yeniden yapılandırma: Anlaşılabilirliği, sürdürülebilirliği vb. iyileştirmek için tasarımı değiştirin. Teknikler arasında benzer sınıfların birleştirilmesi, karmaşık sınıfların bölünmesi vb. yer alır.

Optimizasyon: Sistemin, örneğin değiştirilen algoritmalar veya yeniden yapılanma gibi beklenen performans gereksinimlerini karşıladığından emin olun.

Tamamlama: Tüm boşlukları doldurun, arayüzleri belirtin, vs.

Tasarım yinelemelidir

Süreç ön tasarımdan spesifikasyona, uygulamaya ve teste doğru ilerledikçe, program tasarımında zayıflıklar bulmak yaygındır. Büyük değişiklikler yapmaya hazır olun.

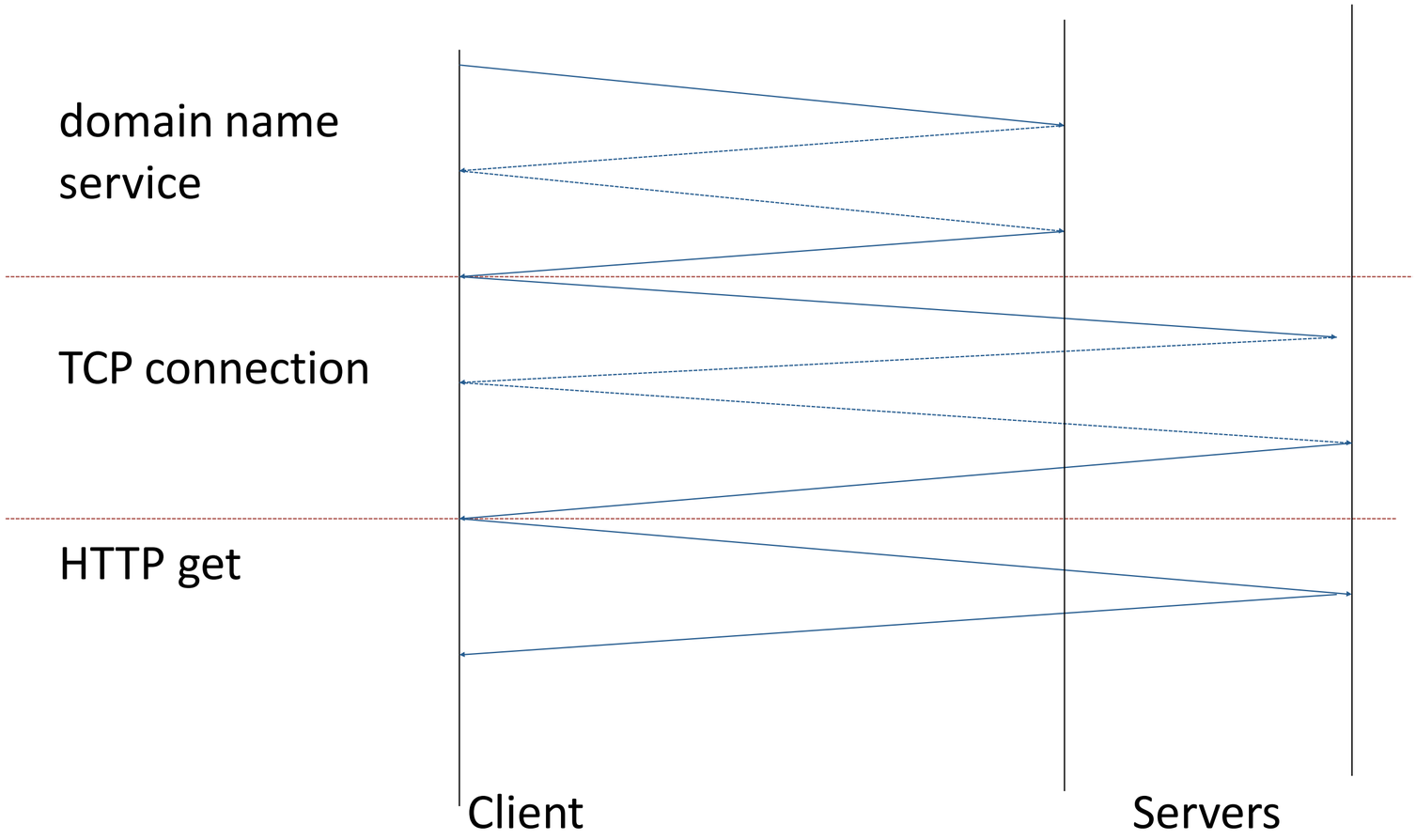
Sistemlerin Dinamik Yönlerinin Modellenmesi

Etkileşim diyagramı: Aralarında gönderilebilecek iletiler de dahil olmak üzere nesne kümesini ve ilişkilerini gösterir.

- Sıralı diyagramlar***:*** Mesajların zaman sıralaması

Etkileşimli: Informal Bouncing Ball Diagramlar

Örnek: bir HTTP get komutunun yürütülmesi.,
ör: http://bm.erciyes.edu.tr/



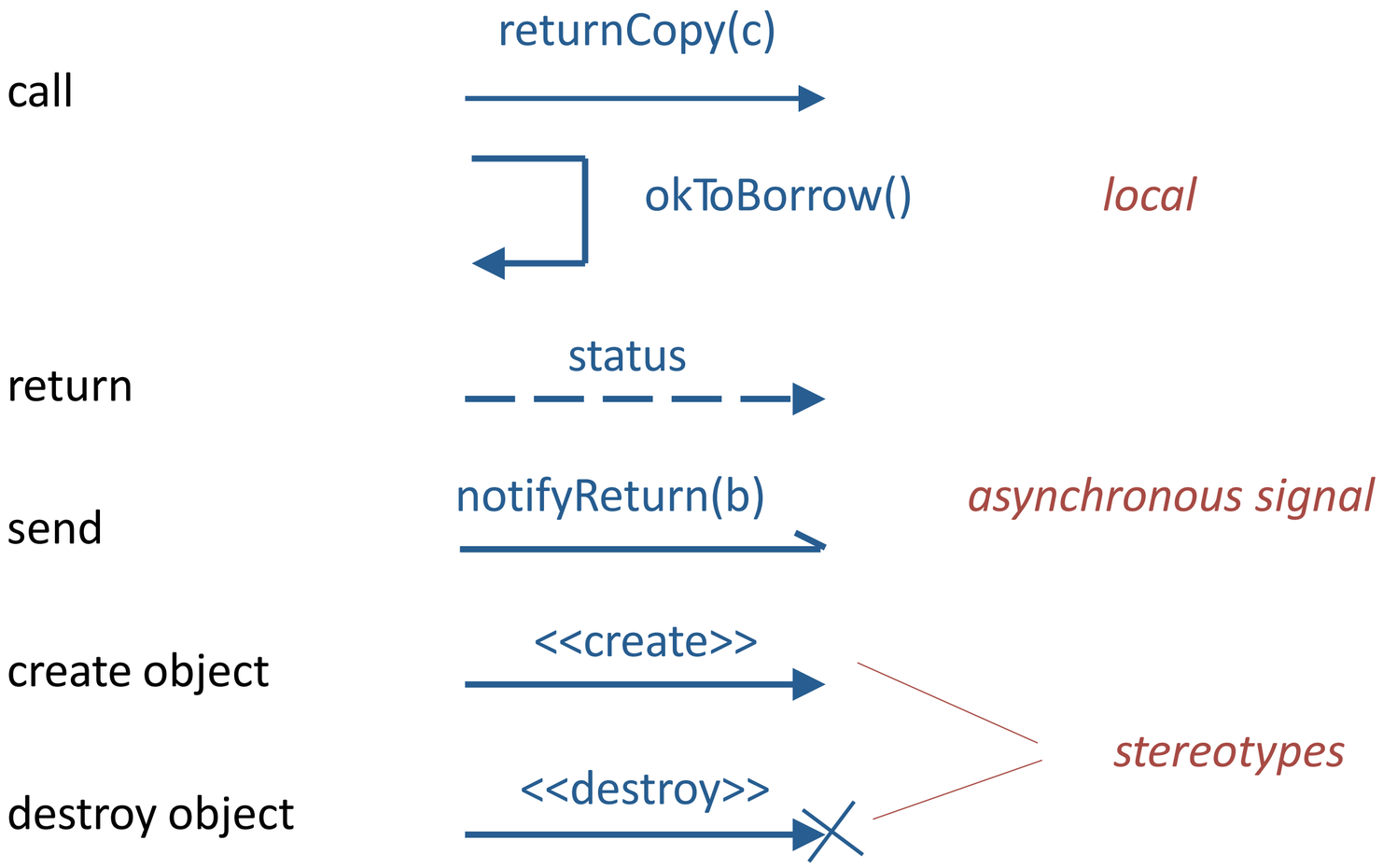
Sınıflar ve Nesneler için UML Gösterimi

Classes	Objects
AnyClass attribute1 attribute2 method1() method2()	anObject:AnyClass
veya	*veya*
	:AnyClass
	veya
	anObject
AnyClass	Nesnelerin adlarının altı çizilir.

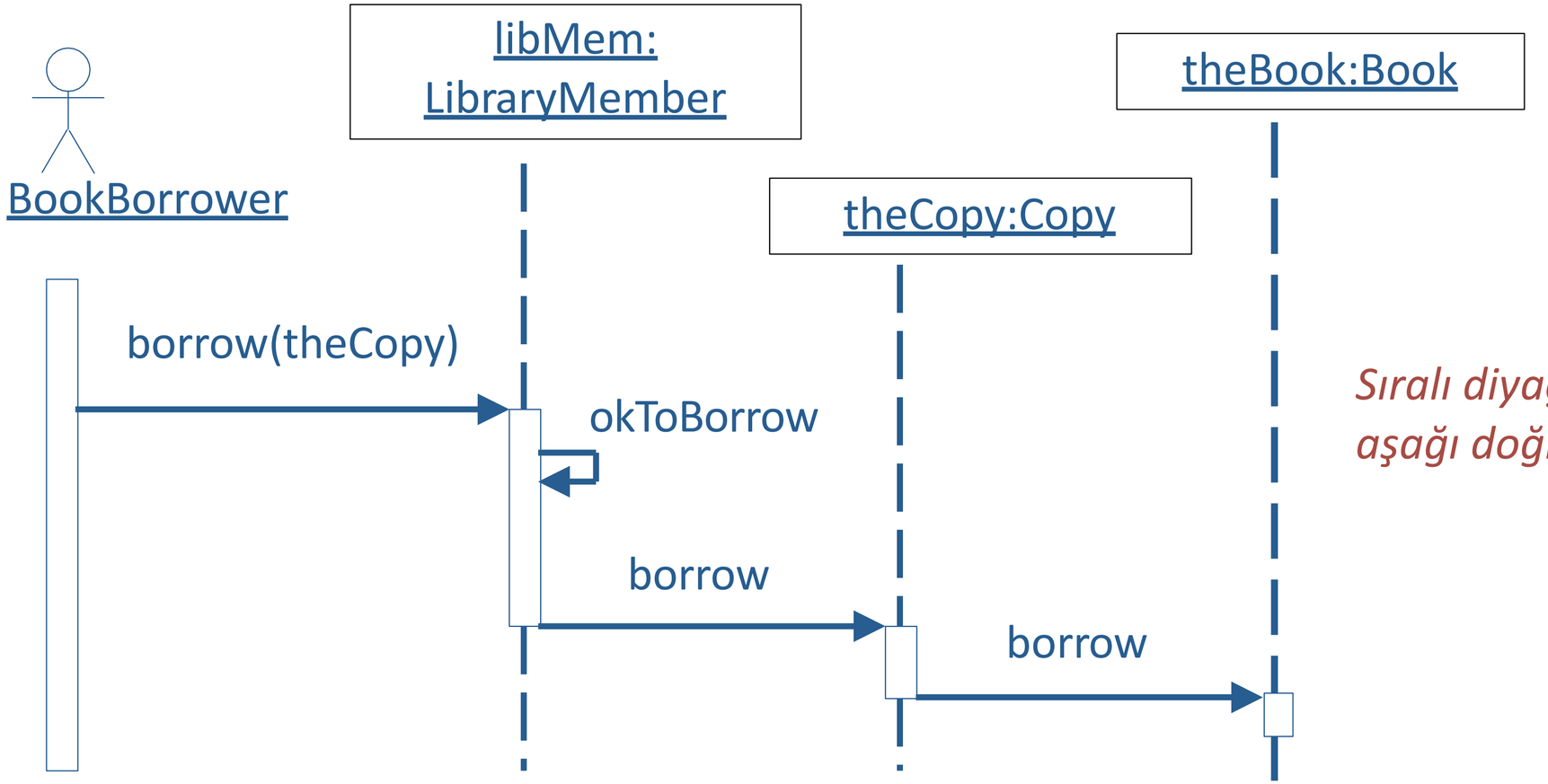
Notation: Interaction

Etkileşim, belirli bir amacı gerçekleştirmek için belirli bir bağlam içindeki bir dizi nesne arasında alınıp verilen bir dizi **mesajı** içeren bir davranıştır.

Nesneler Üzerindeki Eylemler

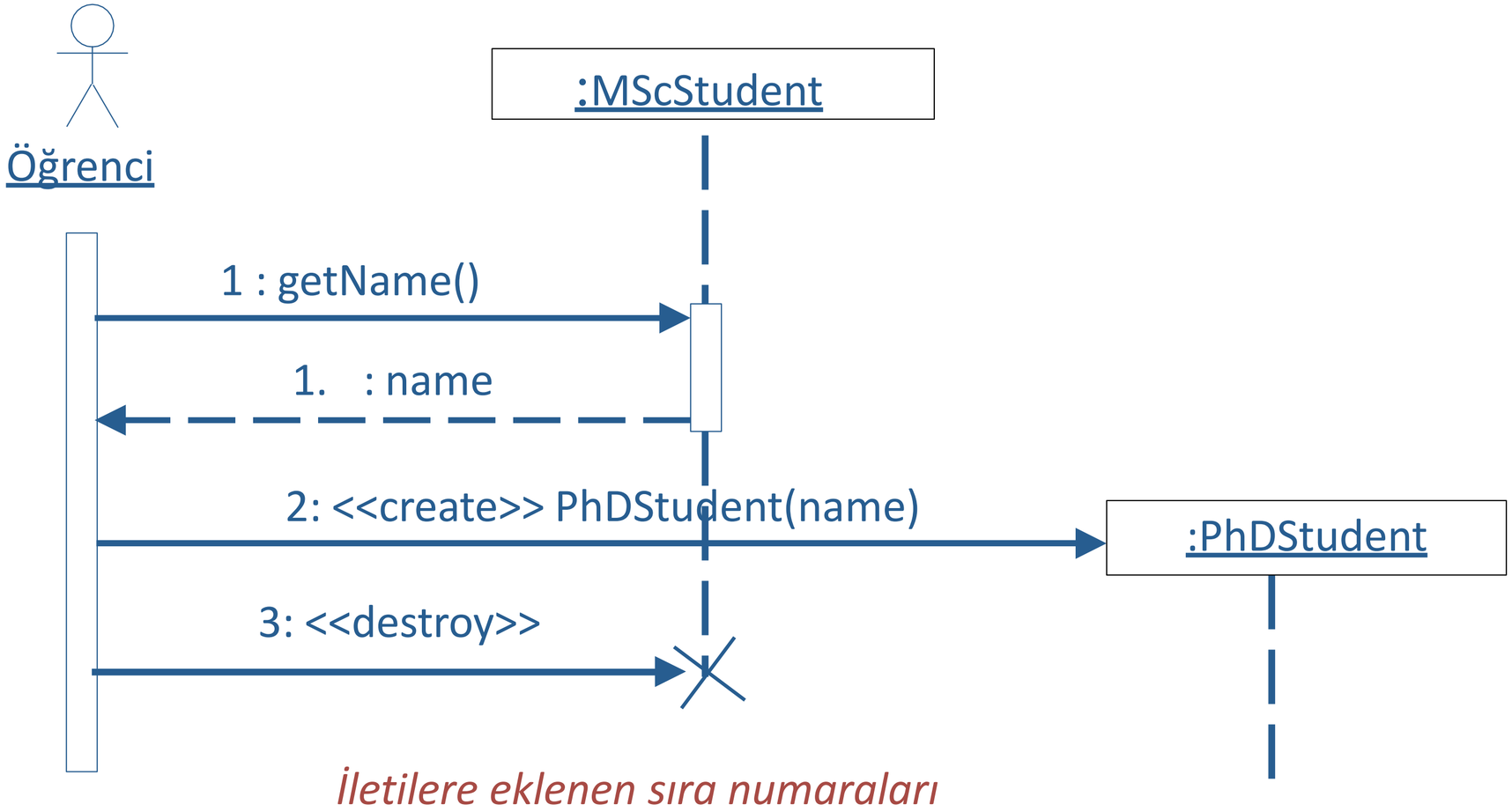


Sequence Diagram: Kitabın kopyasını ödünç alma



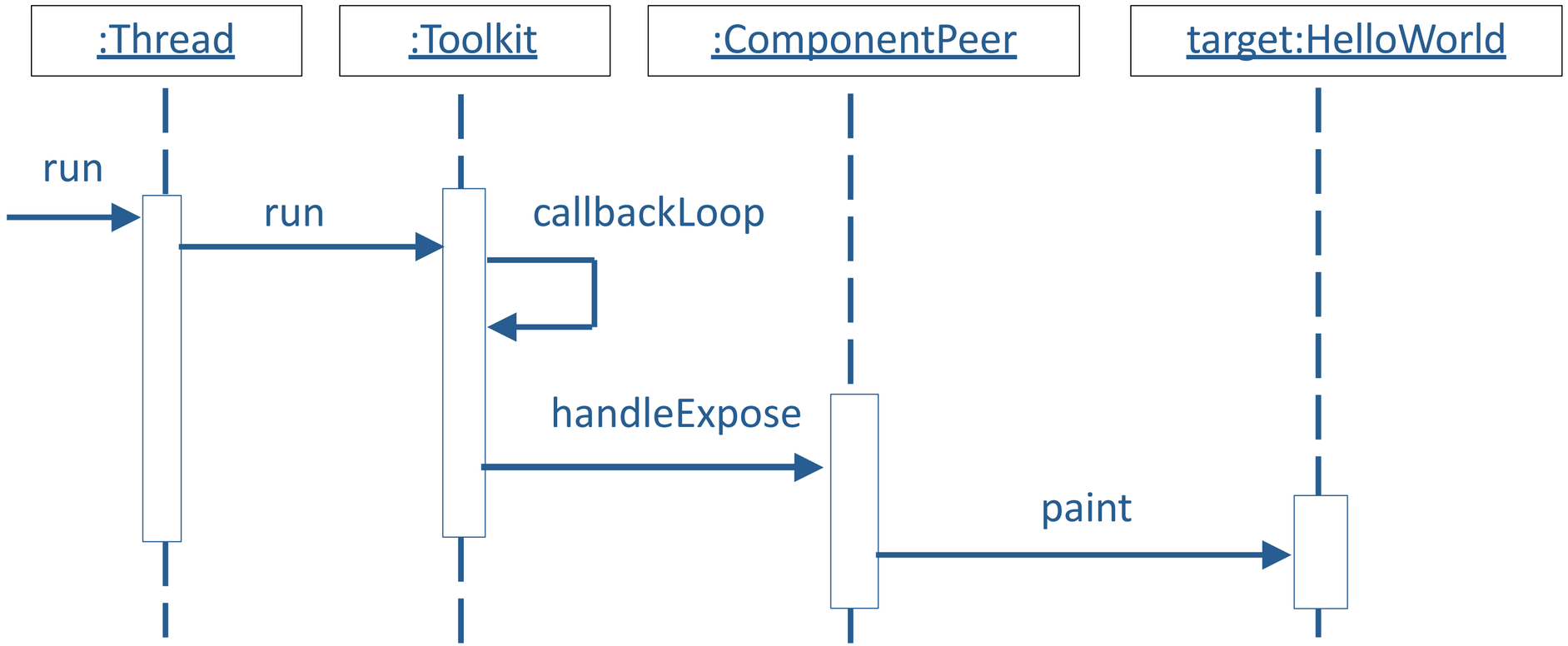
Sıralı diyagramda zaman aşağı doğru akar

Sequence Diagram: Erciyes Programında Değişiklik



İletilere eklenen sıra numaraları

Sequence Diagram: Boyama Mekanizması



BZ 313 Yazılım Mühendisliği

16. Program Tasarımında Modeller

Ders Sonu