9. Ders: JAVA ile Nesne Yönelimli Programlama
Abstract Classes (Soyut Sınıflar)

Fırat Üniversitesi Teknoloji Fakültesi Yazılım Mühendisliği Bölümü

YMH112 Algoritma ve Programlama-II

Dr. Öğr. Üyesi Yaman Akbulut

JAVA ile Nesne Yönelimli Programlama

- http://www.kriptarium.com/algoritma.html (Yardımcı kaynak)

- JAVA ile Nesne Yönelimli Programlama

  - Ders 29: Java Polimorfizmi / Yöntem Aşırı Yüklemesi (Method Overloading) (izle)
  - Ders 30: Java Polimorfizmi / Yöntem Geçersiz Kılma (Method Overriding) (izle)
  - Ders 31: super Anahtar Sözcüğünün Kullanımı (izle)
  - Ders 32: final Anahtar Sözcüğünün Kullanımı (izle)
  - Ders 33: Soyut Sınıf (Abstract Class) (izle)
  - Ders 34: Arayüz (Interface) (izle)

Java Keywords

abstract	double	int	super
assert	else	interface	switch
boolean	enum	long	synchronized
break	extends	native	this
byte	final	new	throw
case	finally	package	throws
catch	float	private	transient
char	for	protected	try
class	goto	public	void
const	if	return	volatile
continue	implements	short	while
default	import	static	
do	instanceof	strictfp	

Kaynak: Introduction to JAVA Programming 10th Edition, Y. Daniel Liang

Giriş

Bir üst sınıf, ilgili alt sınıflar için ortak davranışı tanımlar.

Sınıflar için ortak davranışı tanımlamak için bir arayüz (interface) kullanılabilir (ilgisiz sınıflar dahil).

Bir sayı dizisini veya dizeyi (string) sıralamak için java.util.Arrays.sort metodunu kullanabilirsiniz.

Bir geometrik nesne dizisini sıralamak için aynı sıralama (sort) metodunu uygulayabilir misiniz?

Giriş

Bir geometrik nesne dizisini sıralamak için aynı sıralama (sort) metodunu uygulayabilir misiniz?

Bu tür bir kod yazabilmek için arayüzler hakkında bilgi sahibi olmamız gerekir.

Bir arayüz (interface), sınıflar için ortak davranışı tanımlamak içindir (ilgisiz sınıflar dahil).

Arayüzleri tartışmadan önce, yakından ilgili bir konuya bakalım: soyut sınıflar (abstract classes)

Abstract Classes (Soyut Sınıflar)

Nesne oluşturmak için soyut bir sınıf (abstract class) kullanılamaz.

Soyut bir sınıf, somut alt sınıflarda uygulanan soyut metotlar içerebilir.

Kalıtım hiyerarşisinde, sınıflar her yeni alt sınıfla daha spesifik ve somut hale gelir.

Bir alt sınıftan bir üst sınıfa geçerseniz, sınıflar daha genel ve daha az spesifik hale gelir.

Abstract Classes (Soyut Sınıflar)

Sınıf tasarımı, bir üst sınıfın alt sınıflarının ortak özelliklerini içermesini sağlamalıdır.

Bazen bir üst sınıf o kadar soyuttur ki herhangi bir özel örnek oluşturmak için kullanılamaz.

Böyle bir sınıfa soyut sınıf (abstract class) denir.

Abstract Classes (Soyut Sınıflar)

Önceki bölümlerde GeometrikSekil, Cember ve Dikdortgen için süper sınıf olarak tanımlandı.

GeometrikSekil, geometrik nesnelerin ortak özelliklerini modeller.

Hem Cember hem de Dikdortgen, bir çemberin ve bir dikdörtgenin alanını ve çevresini hesaplamak için getAlan() ve getCevre() metotlarını içerir.

Tüm geometrik nesneler için alanları ve çevreleri hesaplayabildiğinizden, getAlan() ve getCevre() metotlarını GeometrikSekil sınıfında tanımlamak daha iyidir.

Abstract Classes (Soyut Sınıflar)

Ancak, bu metotlar GeometrikSekil sınıfında uygulanamaz çünkü bunların uygulanması belirli geometrik nesne türüne bağlıdır.

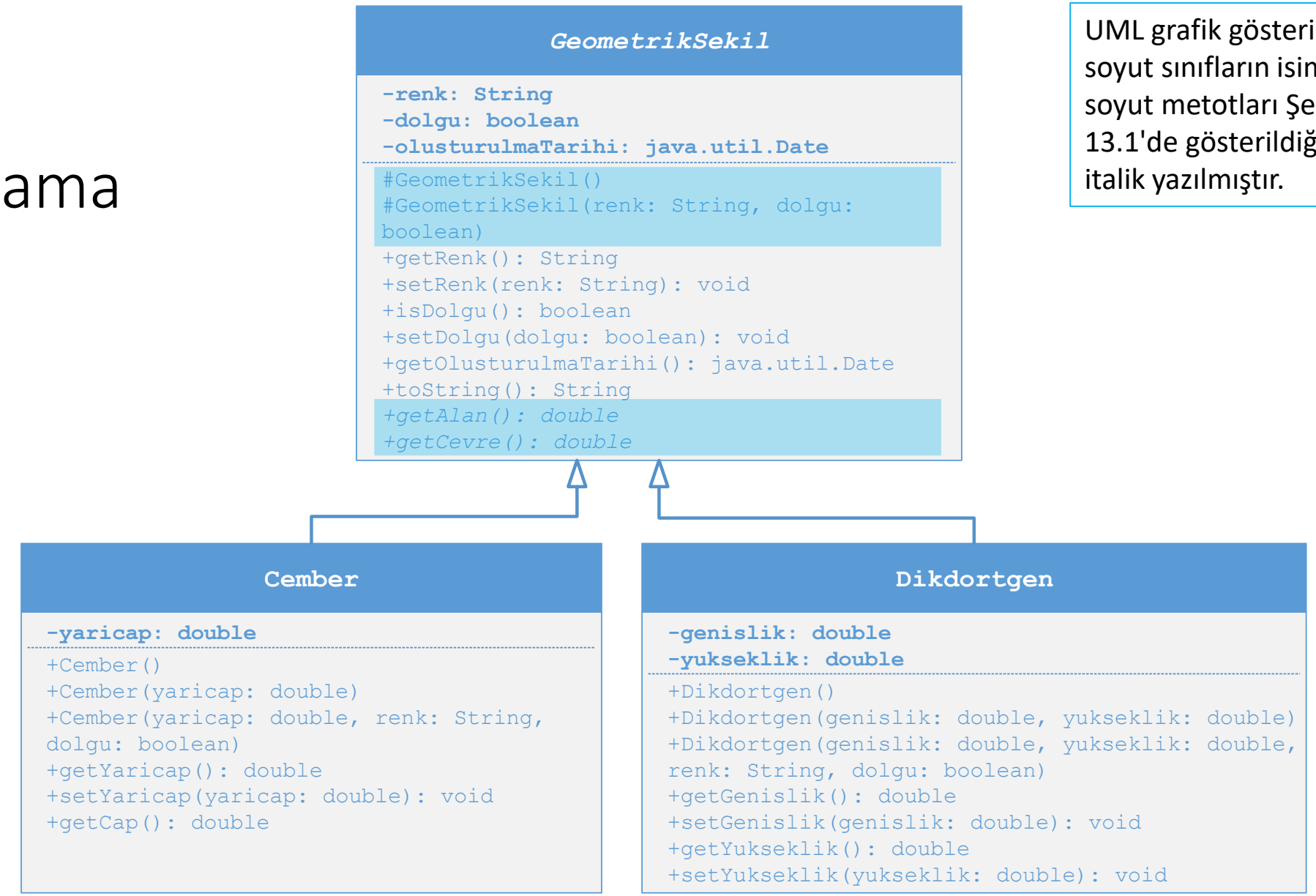
Bu tür metotlara soyut (abstract) metotlar olarak atıfta bulunulur ve metot başlığındaki abstract belirleyici kullanılarak belirtilir.

GeometrikSekil'de metotları tanımladıktan sonra, sınıf soyut bir sınıf haline gelir.

Soyut sınıflar, sınıf başlığındaki abstract belirleyici kullanılarak gösterilir.

UML grafik gösteriminde, soyut sınıfların isimleri ve soyut metotları Şekil 13.1'de gösterildiği gibi italik yazılmıştır.

Soyut metot tanımlama



UML grafik gösteriminde, soyut sınıfların isimleri ve soyut metotları Şekil 13.1'de gösterildiği gibi italik yazılmıştır.

Şekil 13.1. Yeni GeometrikSekil sınıfı soyut metotlar içermektedir.

GeometrikSekil sınıfı

```java
public abstract class GeometrikSekil {
    private String renk = "beyaz";
    private boolean dolgu;
    private java.util.Date olusturulmaTarihi;

    /** Varsayilan bir geometrik nesne olustur */
    protected GeometrikSekil() {
        olusturulmaTarihi = new java.util.Date();
    }

    /** Belirtilen renk ve dolgu degeri ile
    * varsayilan bir geometrik nesne olustur */
    protected GeometrikSekil(String renk, boolean dolgu) {
        olusturulmaTarihi = new java.util.Date();
        this.renk = renk;
        this.dolgu = dolgu;
    }

    /** Renk dondur */
    public String getRenk() {
        return renk;
    }

    /** Yeni renk ata */
    public void setRenk(String renk) {
        this.renk = renk;
    }

    /** Dolgu dondur. dolgu boolean oldugundan,
    get metodunu is..... seklinde adlandirildi */
    public boolean isDolgu() {
        return dolgu;
    }

    /** Dolgu ata */
    public void setDolgu(boolean dolgu) {
        this.dolgu = dolgu;
    }

    /** OlusturulmaTarihi dondur */
    public java.util.Date getOlusturulmaTarihi() {
        return olusturulmaTarihi;
    }

    /** Bu nesnenin string sunumunu dondur */
    @Override
    public String toString() {
        return "Olusturulma Tarihi: " + olusturulmaTarihi
        + "\nrenk: " + renk + " ve dolgu: " + dolgu;
    }

    /** Soyut metot getAlan */
    public abstract double getAlan();

    /** Soyut metot getCevre */
    public abstract double getCevre();
}
```

Cember sınıfı

```java
public class Cember extends GeometrikSekil{
    private double yaricap;

    public Cember() {
    }

    public Cember(double yaricap) {
        this.yaricap = yaricap;
    }

    public Cember(double yaricap, String renk, boolean dolgu) {
        this.yaricap = yaricap;
        setRenk(renk);
        setDolgu(dolgu);
    }

    /** Yaricap dondur */
    public double getYaricap() {
        return yaricap;
    }

    /** Yeni bir yaricap ata */
    public void setYaricap(double yaricap) {
        this.yaricap = yaricap;
    }

    /** Alan dondur */
    public double getAlan() {
        return yaricap * yaricap * Math.PI;
    }

    /** Cap dondur */
    public double getCap() {
        return 2 * yaricap;
    }

    /** Cevre dondur */
    public double getCevre() {
        return 2 * yaricap * Math.PI;
    }

    /** Cember bilgisini yazdir */
    public void yazdirCember() {
        System.out.println("Cember, " + getOlusturulmaTarihi() +
        " tarihinde olusturuldu ve yaricapi " + yaricap + " dir.");
    }

    // Üst sınıfta tanımlanan toString metodunu gecersiz kılma
    public String toString() {
        return super.toString() + "\n yari cap: " + yaricap;
    }
}
```

Dikdortgen sınıfı

```java
public class Dikdortgen extends GeometrikSekil {
    private double genislik;
    private double yukseklik;

    public Dikdortgen() {
    }

    public Dikdortgen(double genislik, double yukseklik) {
        this.genislik = genislik;
        this.yukseklik = yukseklik;
    }

    public Dikdortgen(double genislik, double yukseklik,
                      String renk, boolean dolgu) {
        this.genislik = genislik;
        this.yukseklik = yukseklik;
        setRenk(renk);
        setDolgu(dolgu);
    }

    /** Genislik dondur */
    public double getGenislik() {
        return genislik;
    }

    /** Yeni bir genislik ata */
    public void setGenislik(double genislik) {
        this.genislik = genislik;
    }

    /** Yukseklik dondur */
    public double getYukseklik() {
        return yukseklik;
    }

    /** Yeni bir yukseklik ata */
    public void setYukseklik(double yukseklik) {
        this.yukseklik = yukseklik;
    }

    /** Alan dondur */
    public double getAlan() {
        return genislik * yukseklik;
    }

    /** Cevre dondur */
    public double getCevre() {
        return 2 * (genislik + yukseklik);
    }
}
```

Abstract Classes (Soyut Sınıflar)

Soyut sınıflar normal sınıflar gibidir,

ancak new operatörünü kullanarak soyut sınıfların örneklerini oluşturamazsınız.

Soyut bir metot, uygulama (implementation) olmadan tanımlanır.

Uygulanması (implementation) alt sınıflar tarafından sağlanır.

Soyut (abstract) metotlar içeren bir sınıf, soyut (abstract) olarak tanımlanmalıdır.

Abstract Classes (Soyut Sınıflar)

Soyut sınıftaki yapıcı, yalnızca alt sınıflar tarafından kullanıldığından korumalı (protected) olarak tanımlanır.

Somut bir alt sınıfın bir örneğini oluşturduğunuzda, üst sınıfın yapıcısı, üst sınıfta tanımlanan veri alanlarını başlatmak için çağrılır.

Abstract Classes (Soyut Sınıflar)

GeometrikSekil soyut sınıfı, geometrik nesneler için ortak özellikleri (veriler ve metotlar) tanımlar ve uygun yapıcılar sağlar.

Geometrik nesnelerin alanlarını ve çevresini nasıl hesaplanacağı o an için bilinmediğinden getAlan() ve getCevre() soyut metotlar olarak tanımlanır.

Bu metotlar alt sınıflarda uygulanmaktadır.

Cember ve Dikdortgen'in uygulaması, bu bölümde tanımlanan GeometrikSekil sınıfından türetilmeleri (genişletilmeleri) dışında önceki bölümlerdeki kod ile aynıdır.

Abstract Classes (Soyut Sınıflar)

GeometrikSekil sınıfında getAlan() ve getCevre() metotlarını soyut (abstract) olarak tanımlayarak hangi avantajın elde edildiğini merak ediyor olabilirsiniz.

Bunları, GeometrikSekil sınıfında tanımlamanın faydaları Örnek 1'de gösterilmiştir.

Program, çember ve dikdörtgen olmak üzere iki geometrik nesne oluşturur, eşit alanlara sahip olup olmadıklarını kontrol etmek için esitAlan metodunu çağırır

ve bunları görüntülemek için gosterGeometrikSekil metodunu çağırır.

Örnek 1:

```
C:\Program Files\Java\jdk-16.0.1\bin\yeni2>
java TestGeometrikSekil
Iki nesne esit alana mi sahip? false

Alan: 78.53981633974483
Cevre: 31.41592653589793

Alan: 15.0
Cevre: 16.0
```

```java
public class TestGeometrikSekil {
    /** Main metot */
    public static void main(String[] args) {
        // İki geometrik nesne oluştur
        GeometrikSekil geoNesne1 = new Cember(5);
        GeometrikSekil geoNesne2 = new Dikdortgen(5, 3);

        System.out.println("Iki nesne esit alana mi sahip? " + esitAlan(geoNesne1, geoNesne2));

        // Cemberi göster
        gosterGeometrikSekil(geoNesne1);

        // Dikdörtgeni göster
        gosterGeometrikSekil(geoNesne2);
    }

    /** İki geometrik nesnenin alanını karşılaştıran metot */
    public static boolean esitAlan(GeometrikSekil nesne1, GeometrikSekil nesne2) {
        return nesne1.getAlan() == nesne2.getAlan();
    }

    /** Geometrik nesneyi gösteren metot */
    public static void gosterGeometrikSekil(GeometrikSekil nesne) {
        System.out.println();
        System.out.println("Alan: " + nesne.getAlan());
        System.out.println("Cevre: " + nesne.getCevre());
    }
}
```

Örnek 1:

GeometrikSekil sınıfında tanımlanan getAlan() ve getCevre() metotları, Cember sınıfında ve Dikdortgen sınıfında geçersiz (override) kılınır.

Örnek 1, ifadeler (5-6. Satırlar)

```
5        GeometrikSekil geoNesne1 = new Cember(5);
6        GeometrikSekil geoNesne2 = new Dikdortgen(5, 3);
```

yeni bir çember ve dikdörtgen oluşturun ve bunları geoNesne1 ve geoNesne2 değişkenlerine atayın.

Bu iki değişken GeometrikSekil tipindedir.

Örnek 1:

esitAlan (geoNesne1, geoNesne2) (satır 8) çağrılırken, geoNesne1 bir cember olduğundan Cember sınıfında tanımlanan getAlan() metodu nesne1.getAlan() için kullanılır.

geoNesne2 bir dikdörtgen olduğundan Dikdortgen sınıfında tanımlanan getAlan() metodu nesne2.getAlan() için kullanılır.

Benzer şekilde, gosterGeometrikSekil(geoNesne1) (satır 11) çağrılırken, Cember sınıfında tanımlanan getAlan() ve getCevre() metotları kullanılır

ve gosterGeometrikSekil(geoNesne2) (satır 14) çağrılırken Dikdortgen sınıfındaki getAlan ve getCevre metotları kullanılır.

Örnek 1:

JVM, metodu çağıran gerçek nesneye bağlı olarak bu metotlardan hangisinin çalışma zamanında çağrılacağını dinamik olarak belirler.

getAlan metodu GeometrikSekil'de tanımlanmasaydı, iki geometrik nesnenin aynı alana sahip olup olmadığını karşılaştırmak için esitAlan metodunu tanımlayamayacağımızı unutmayınız.

Artık GeometrikSekil'de soyut metotları tanımlamanın faydalarını görmüş olduk.