8. Ders: JAVA ile Nesne Yönelimli Programlama
File Class (Dosya Sınıfı)

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

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

File Class (Dosya Sınıfı)

File sınıfı, bir dosya veya dizinin (klasör) özelliklerini elde etmek ve bir dosya veya dizini yeniden adlandırmak veya silmek için metotları içerir.

Programda depolanan veriler geçicidir; program sona erdiğinde kaybolurlar.

Bir programda oluşturulan verileri kalıcı olarak depolamak için, bunları bir diskteki veya başka bir kalıcı depolama aygıtındaki bir dosyaya kaydetmeniz gerekir.

Dosya daha sonra başka programlar tarafından taşınabilir ve okunabilir.

File Class (Dosya Sınıfı)

Her dosya, dosya sistemindeki bir dizine yerleştirilir.

Mutlak bir dosya adı (veya tam adı), tam yolu ve sürücü harfiyle birlikte bir dosya adı içerir.

Örneğin, C:\Program Files\Java\jdk-15.0.1\bin\yeni2\Hesapla.java, Windows işletim sistemindeki Hesapla.java dosyasının mutlak dosya adıdır.

Burada C:\Program Files\Java\jdk-15.0.1\bin\yeni2, dosyanın dizin yolu olarak adlandırılır.

Mutlak dosya adları makineye bağlıdır. Windows işletim sisteminde "\", Unix veya Linux işletim sistemlerinde dizin ayırıcı "/" şeklindedir.

File Class (Dosya Sınıfı)

Göreceli bir dosya adı, geçerli çalışma dizini ile ilişkilidir.

Göreceli bir dosya adı için tam dizin yolu atlanmıştır.

Örneğin, Hesapla.java göreceli bir dosya adıdır.

Mevcut çalışma dizini C:\Program Files\Java\jdk-15.0.1\bin\yeni2 ise,

mutlak dosya adı C:\Program Files\Java\jdk-15.0.1\bin\yeni2\Hesapla.java olacaktır.

File Class (Dosya Sınıfı)

File sınıfının, dosyaların ve yol adlarının makineye bağlı karmaşıklıklarının çoğunu makineden bağımsız bir şekilde ele alan bir soyutlama sağlaması amaçlanmıştır.

File sınıfı, Şekil 12.6'da gösterildiği gibi, dosya ve dizin özelliklerini edinme ve dosya ve dizinleri yeniden adlandırma ve silme metotlarını içerir.

Ancak, File sınıfı, dosya içeriklerini okuma ve yazma metotlarını içermez.

File Class (Dosya Sınıfı)

Dosya adı bir dizedir. File sınıfı, dosya adı ve dizin yolu için bir sarmalayıcı sınıfıdır.

Örneğin, new File(" C:\\Program Files\\Java\\jdk-15.0.1\\bin\\yeni2 ")
C:\Program Files\Java\jdk-15.0.1\bin\yeni2 dizini için bir File nesnesi oluşturur. (windows)

new File("C:\\Program Files\\Java\\jdk-15.0.1\\bin\\yeni2\\Hesapla.java")
C:\Program Files\Java\jdk-15.0.1\bin\yeni2\Hesapla.java dosyası için bir File nesnesi oluşturur. (windows)

Unix veya Linux işletim sistemlerinde …

File Class (Dosya Sınıfı)

new File("C:\\Program Files\\Java\\jdk-15.0.1\\bin\\yeni2\\Hesapla.java")
Windows için dizin ayırıcı bir ters eğik çizgidir (\). (backslash)

Ters eğik çizgi, Java'da özel bir karakterdir ve bir dizin değişmezinde \\ olarak yazılmalıdır.

Kaçış kodları	adı	unicode kodu	decimal değeri
\b	backspace	\u0008	8
\t	tab	\u0009	9
\n	linefeed	\u000A	10
\f	formfeed	\u000C	12
\r	carriage return	\u000D	13
\\	backslash	\u005C	92
\"	double quote	\u0022	34

File Class (Dosya Sınıfı)

Nesnenin bir dizini temsil edip etmediğini kontrol etmek için File sınıfının isDirectory() metodunu

ve nesnenin bir dosyayı temsil edip etmediğini kontrol etmek için isFile() metodunu kullanabilirsiniz.

Bir File örneğinin (nesne) oluşturulması, makinede bir dosya oluşturmaz.

Var olup olmadığına bakılmaksızın herhangi bir dosya adı için bir File örneği oluşturabilirsiniz.

Dosyanın var olup olmadığını kontrol etmek için bir File örneğinde exists() metodunu çağırabilirsiniz.

java.io.File

java.io.File	
`+File(pathname: String)`	Creates a `File` object for the specified path name. The path name may be a directory or a file.
`+File(parent: String, child: String)`	Creates a `File` object for the child under the directory parent. The child may be a file name or a subdirectory.
`+File(parent: File, child: String)`	Creates a `File` object for the child under the directory parent. The parent is a `File` object. In the preceding constructor, the parent is a string.
`+exists(): boolean`	Returns true if the file or the directory represented by the `File` object exists.
`+canRead(): boolean`	Returns true if the file represented by the `File` object exists and can be read.
`+canWrite(): boolean`	Returns true if the file represented by the `File` object exists and can be written.
`+isDirectory(): boolean`	Returns true if the `File` object represents a directory.
`+isFile(): boolean`	Returns true if the `File` object represents a file.
`+isAbsolute(): boolean`	Returns true if the `File` object is created using an absolute path name.
`+isHidden(): boolean`	Returns true if the file represented in the `File` object is hidden. The exact definition of *hidden* is system-dependent. On Windows, you can mark a file hidden in the File Properties dialog box. On Unix systems, a file is hidden if its name begins with a period(.) character.
`+getAbsolutePath(): String`	Returns the complete absolute file or directory name represented by the `File` object.
`+getCanonicalPath(): String`	Returns the same as `getAbsolutePath()` except that it removes redundant names, such as "." and "..", from the path name, resolves symbolic links (on Unix), and converts drive letters to standard uppercase (on Windows).
`+getName(): String`	Returns the last name of the complete directory and file name represented by the `File` object. For example, new `File("c:\\book\\test.dat").getName()` returns `test.dat`.
`+getPath(): String`	Returns the complete directory and file name represented by the `File` object. For example, new `File("c:\\book\\test.dat").getPath()` returns `c:\book\test.dat`.
`+getParent(): String`	Returns the complete parent directory of the current directory or the file represented by the `File` object. For example, new `File("c:\\book\\test.dat").getParent()` returns `c:\book`.
`+lastModified(): long`	Returns the time that the file was last modified.
`+length(): long`	Returns the size of the file, or 0 if it does not exist or if it is a directory.
`+listFile(): File[]`	Returns the files under the directory for a directory `File` object.
`+delete(): boolean`	Deletes the file or directory represented by this `File` object.The method returns true if the deletion succeeds.
`+renameTo(dest: File): boolean`	Renames the file or directory represented by this `File` object to the specified name represented in dest. The method returns true if the operation succeeds.
`+mkdir(): boolean`	Creates a directory represented in this `File` object. Returns true if the the directory is created successfully.
`+mkdirs(): boolean`	Same as `mkdir()` except that it creates directory along with its parent directories if the parent directories do not exist.

FIGURE 12.6 The `File` class can be used to obtain file and directory properties, to delete and rename files and directories, and to create directories.

java.io.File

java.io.File	
`+File(pathname: String)`	Creates a `File` object for the specified path name. The path name may be a directory or a file.
`+File(parent: String, child: String)`	Creates a `File` object for the child under the directory parent. The child may be a file name or a subdirectory.
`+File(parent: File, child: String)`	Creates a `File` object for the child under the directory parent. The parent is a `File` object. In the preceding constructor, the parent is a string.
`+exists(): boolean`	Returns true if the file or the directory represented by the `File` object exists.
`+canRead(): boolean`	Returns true if the file represented by the `File` object exists and can be read.
`+canWrite(): boolean`	Returns true if the file represented by the `File` object exists and can be written.
`+isDirectory(): boolean`	Returns true if the `File` object represents a directory.
`+isFile(): boolean`	Returns true if the `File` object represents a file.
`+isAbsolute(): boolean`	Returns true if the `File` object is created using an absolute path name.
`+isHidden(): boolean`	Returns true if the file represented in the `File` object is hidden. The exact definition of *hidden* is system-dependent. On Windows, you can mark a file hidden in the File Properties dialog box. On Unix systems, a file is hidden if its name begins with a period(.) character.
`+getAbsolutePath(): String`	Returns the complete absolute file or directory name represented by the `File` object.
`+getCanonicalPath(): String`	Returns the same as `getAbsolutePath()` except that it removes redundant names, such as "." and "..", from the path name, resolves symbolic links (on Unix), and converts drive letters to standard uppercase (on Windows).
`+getName(): String`	Returns the last name of the complete directory and file name represented by the `File` object. For example, new `File("c:\\book\\test.dat").getName()` returns `test.dat`.

java.io.File

`+getPath(): String`	Returns the complete directory and file name represented by the `File` object. For example, new `File("c:\\book\\test.dat").getPath()` returns `c:\book\test.dat`.
`+getParent(): String`	Returns the complete parent directory of the current directory or the file represented by the `File` object. For example, new `File("c:\\book\\test.dat").getParent()` returns `c:\book`.
`+lastModified(): long`	Returns the time that the file was last modified.
`+length(): long`	Returns the size of the file, or 0 if it does not exist or if it is a directory.
`+listFile(): File[]`	Returns the files under the directory for a directory `File` object.
`+delete(): boolean`	Deletes the file or directory represented by this `File` object.The method returns true if the deletion succeeds.
`+renameTo(dest: File): boolean`	Renames the file or directory represented by this `File` object to the specified name represented in dest. The method returns true if the operation succeeds.
`+mkdir(): boolean`	Creates a directory represented in this `File` object. Returns true if the the directory is created successfully.
`+mkdirs(): boolean`	Same as `mkdir()` except that it creates directory along with its parent directories if the parent directories do not exist.

FIGURE 12.6 The `File` class can be used to obtain file and directory properties, to delete and rename files and directories, and to create directories.

Örnek 1: java.io.File

```java
public class TestFileClass {
    public static void main(String[] args) {
        java.io.File dosya = new java.io.File("deneme.txt");
        System.out.println("Dosya var mi? " + dosya.exists());
        System.out.println("Dosyanin uzunlugu " + dosya.length() + " bytes");
        System.out.println("Dosya okunabilir mi? " + dosya.canRead());
        System.out.println("Dosya yazilabilir mi? " + dosya.canWrite());
        System.out.println("Bir dizin mi? " + dosya.isDirectory());
        System.out.println("Bir dosya mi? " + dosya.isFile());
        System.out.println("Mutlak mi? " + dosya.isAbsolute());
        System.out.println("Gizli mi? " + dosya.isHidden());
        System.out.println("Mutlak yolu " + dosya.getAbsolutePath());
        System.out.println("Degisiklik tarihi " + new java.util.Date(dosya.lastModified()));
    }
}
```

```
C:\Program Files\Java\jdk-16.0.1\bin\yeni2>java TestFileClass
Dosya var mi? true
Dosyanin uzunlugu 0 bytes
Dosya okunabilir mi? true
Dosya yazilabilir mi? true
Bir dizin mi? false
Bir dosya mi? true
Mutlak mi? false
Gizli mi? false
Mutlak yolu C:\Program Files\Java\jdk-16.0.1\bin\yeni2\deneme.txt
Degisiklik tarihi Mon May 10 00:44:30 TRT 2021
```

Dosya Giriş ve Çıkış G/Ç (File Input and Output I/O)

Bir dosyadan metin verilerini okumak için Scanner sınıfını

ve bir dosyaya metin verilerini yazmak için PrintWriter sınıfını kullanılır.

Bir File nesnesi, bir dosyanın veya yolun özelliklerini kapsüller (içerir),

ancak bir dosya oluşturma veya bir dosyaya veri yazma veya dosyadan veri okuma metotlarını içermez (G/Ç (I/O)).

G/Ç gerçekleştirmek için, uygun Java G/Ç sınıflarını kullanarak nesneler oluşturmamız gerekir.

Dosya Giriş ve Çıkış G/Ç (File Input and Output I/O)

Nesneler, bir dosyadan/dosyaya veri okuma/yazma yöntemlerini içerir.

İki tür dosya vardır: metin ve ikili.

Metin dosyaları esasen diskteki karakterlerdir.

Scanner ve PrintWriter sınıfları kullanılarak bir metin dosyasından stringlerin ve sayısal değerlerin nasıl okunacağı veya bir metin dosyasına stringlerin ve sayısal değerlerin nasıl yazılacağı gösterilecektir.

PrintWriter Kullanarak Veri Yazma

java.io.PrintWriter sınıfı, bir dosya oluşturmak ve bir metin dosyasına veri yazmak için kullanılır.
Öncelikle, aşağıdaki gibi bir metin dosyası için bir PrintWriter nesnesi oluşturmanız gerekir:

PrintWriter cikis = new PrintWriter(dosya adı);

Ardından, bir dosyaya veri yazmak için PrintWriter nesnesindeki print, println ve printf metotlarını çağırabiliriz.

java.io.PrintWriter

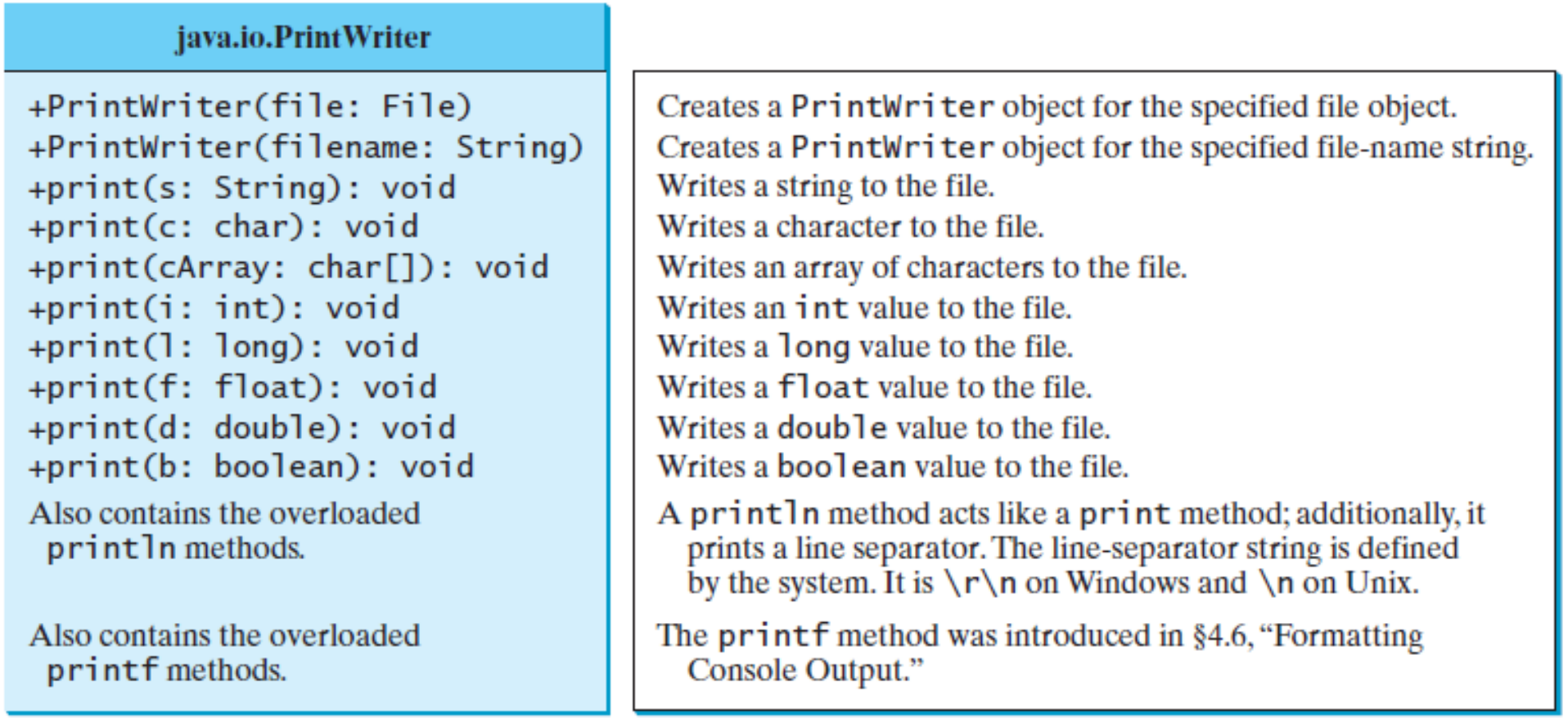
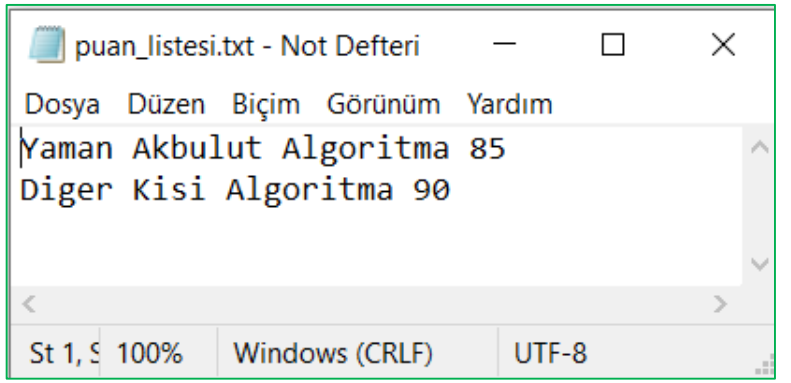


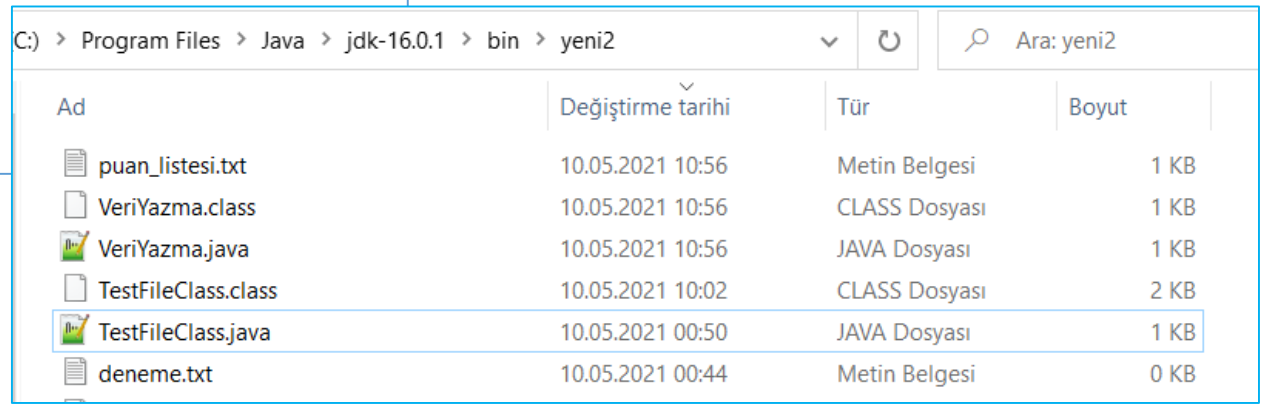
FIGURE 12.8 The `PrintWriter` class contains the methods for writing data to a text file.

Örnek 2: java.io.PrintWriter

```
1  import java.io.*;
2
3  public class VeriYazma {
4      public static void main(String[] args) throws IOException {
5          File dosya = new File("puan_listesi.txt");
6              if (dosya.exists()) {
7              System.out.println("Dosya mevcut");
8              System.exit(1);
9              }
10
11         // Dosya olusturma
12         PrintWriter cikis = new PrintWriter(dosya);
13
14         // Dosyaya yazma
15         cikis.print("Yaman Akbulut Algoritma ");
16         cikis.println(85);
17         cikis.print("Diger Kisi Algoritma ");
18         cikis.println(90);
19
20         // Dosyayı kapatma
21         cikis.close();
22     }
23 }
```

```
C:\Program Files\Java\jdk-16.0.1\bin\
yeni2>javac VeriYazma.java

C:\Program Files\Java\jdk-16.0.1\bin\
yeni2>java VeriYazma

C:\Program Files\Java\jdk-16.0.1\bin\
yeni2>
```



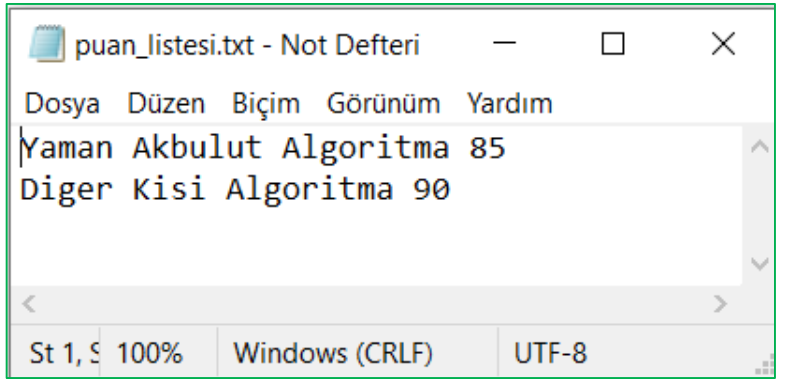


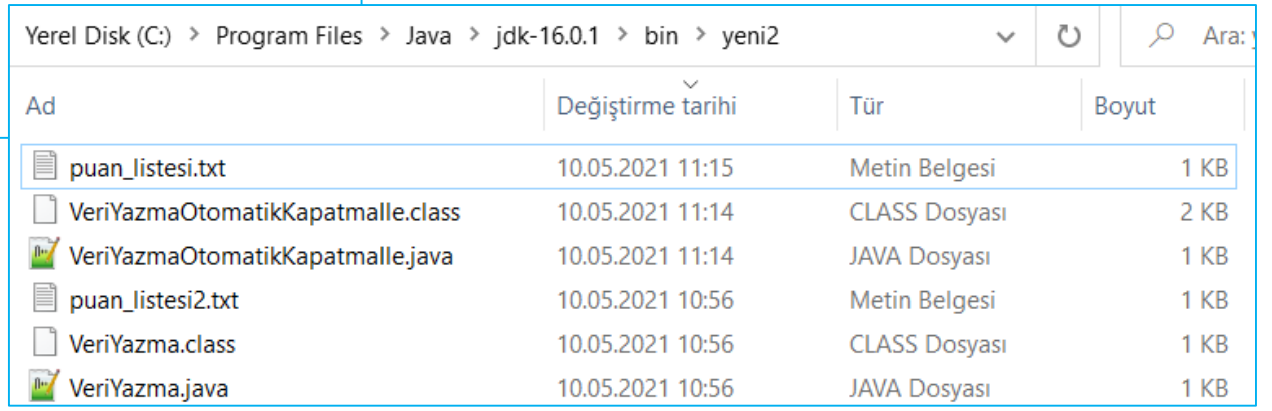
Dosyayı kapatmak için close() metodu kullanılmalıdır (satır 21). Bu metot kullanılmazsa, veriler dosyaya düzgün kaydedilemeyebilir.

Örnek 3: java.io.PrintWriter ve try-with-resources

```java
import java.io.*;

public class VeriYazmaOtomatikKapatmaIle {
    public static void main(String[] args) throws Exception {
        File dosya = new File("puan_listesi.txt");
            if (dosya.exists()) {
            System.out.println("Dosya mevcut");
            System.exit(0);
            }

        try(
            // Dosya olusturma
            PrintWriter cikis = new PrintWriter(dosya);
        ){
            // Dosyaya yazma
            cikis.print("Yaman Akbulut Algoritma ");
            cikis.println(85);
            cikis.print("Diger Kisi Algoritma ");
            cikis.println(90);
        }
    }
}
```

```
C:\Program Files\Java\jdk-16.0.1\bin\yeni2>
javac VeriYazmaOtomatikKapatmaIle.java

C:\Program Files\Java\jdk-16.0.1\bin\yeni2>
java VeriYazmaOtomatikKapatmaIle
Dosya mevcut

C:\Program Files\Java\jdk-16.0.1\bin\yeni2>
java VeriYazmaOtomatikKapatmaIle
```






```
try(kaynakların tanımlanması ve oluşturulması){
   kaynakların kullanımı…
   //try-with-resources
  }
```

Scanner Kullanarak Veri Okuma

java.util.Scanner sınıfı, konsoldan dizeleri (string) ve ilkel (primitive) değerleri okumak için kullandık.

Bir Scanner, girdisini boşluk karakterleriyle sınırlandırılmış belirteçlere (token) böler.

Klavyeden okumak için, aşağıdaki gibi System.in için bir Scanner oluşturulur:

```
Scanner giris = new Scanner(System.in);
```

Bir dosyadan okumak için aşağıdaki gibi bir Scanner oluşturun:

```
Scanner giris = new Scanner (new File(dosya adı));
```

java.util.Scanner

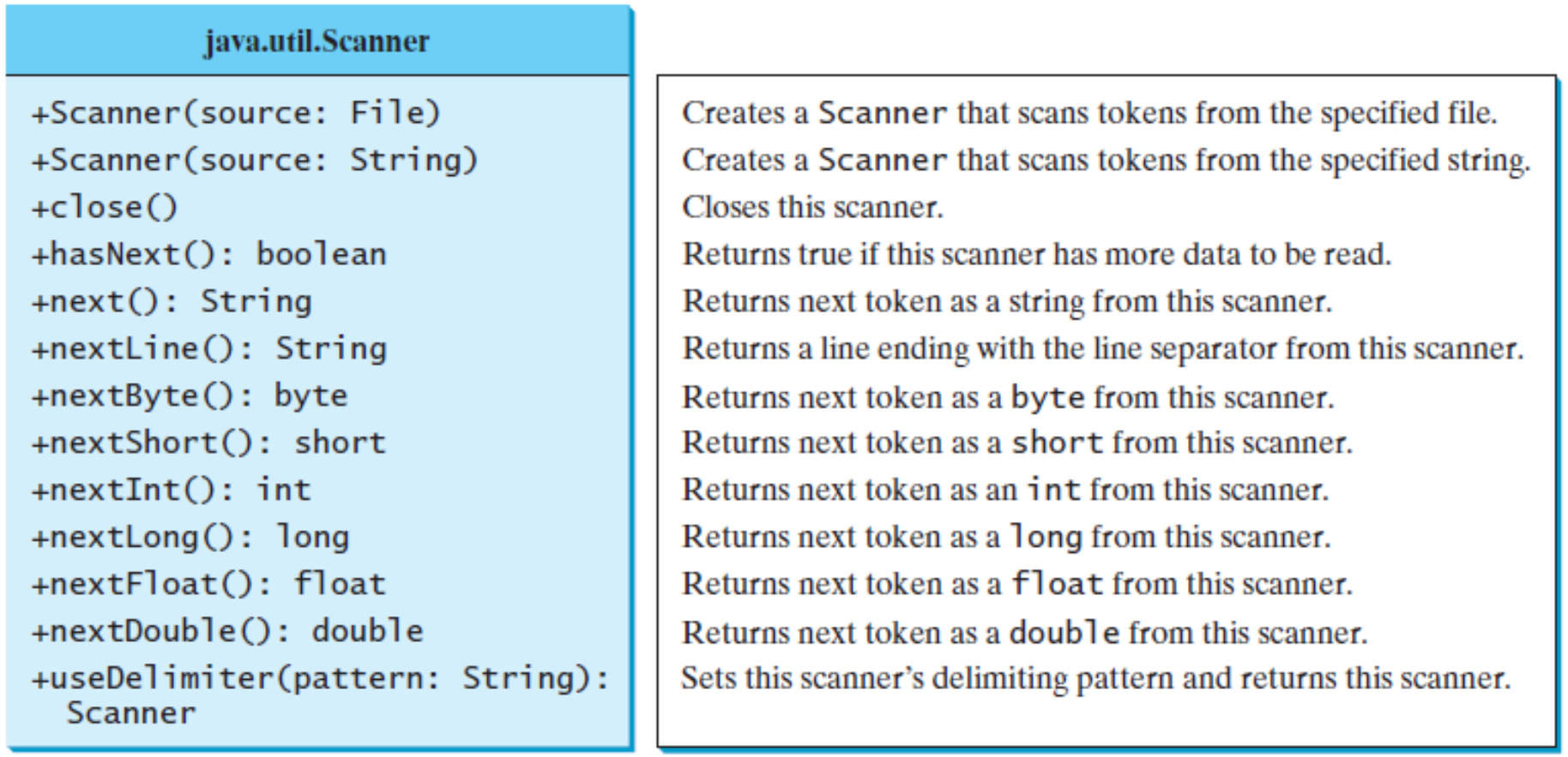


FIGURE 12.9 The Scanner class contains the methods for scanning data.

Örnek 4: java.util.Scanner

```java
import java.util.Scanner;

public class VeriOkuma {
    public static void main(String[] args) throws Exception {
        // File örneği oluşturma
        java.io.File dosya = new java.io.File("puan_listesi.txt");

        // Dosya için bir Scanner oluşturma
        Scanner giris = new Scanner(dosya);

        // Dosyadan veri okuma
        while (giris.hasNext()) {
        String isim = giris.next();
        String soyIsim = giris.next();
        String dersAdi = giris.next();
        int notDegeri = giris.nextInt();
        System.out.println(isim + " " + soyIsim + " " + dersAdi + " " + notDegeri);
        }

        // Dosyayı kapatma
        giris.close();
    }
}
```

```
C:\Program Files\Java\jdk-16.0.1\bin\yeni2>javac VeriOkuma.java

C:\Program Files\Java\jdk-16.0.1\bin\yeni2>java VeriOkuma
Yaman Akbulut Algoritma 85
Diger Kisi Algoritma 90
```

Örnek 4: java.util.Scanner

```java
import java.util.Scanner;

public class VeriOkuma {
    public static void main(String[] args) throws Exception {
        // File örneği oluşturma
        java.io.File dosya = new java.io.File("puan_listesi.txt");

        // Dosya için bir Scanner oluşturma
        Scanner giris = new Scanner(dosya);

        // Dosyadan veri okuma
        while (giris.hasNext()) {
        String isim = giris.next();
        String soyIsim = giris.next();
        String dersAdi = giris.next();
        int notDegeri = giris.nextInt();
        System.out.println(isim + " " + soyIsim + " " + dersAdi + " " + notDegeri);
        }

        // Dosyayı kapatma
        giris.close();
    }
}
```

```
C:\Program Files\Java\jdk-16.0.1\bin\yeni2>java VeriOkuma
Exception in thread "main" java.io.FileNotFoundException: puan_listesi.txt (Sistem belirtilen dosyayı bulamıyor)
        at java.base/java.io.FileInputStream.open0(Native Method)
        at java.base/java.io.FileInputStream.open(FileInputStream.java:211)
        at java.base/java.io.FileInputStream.<init>(FileInputStream.java:153)
        at java.base/java.util.Scanner.<init>(Scanner.java:639)
        at VeriOkuma.main(VeriOkuma.java:9)
```

Scanner nasıl çalışır?

nextByte(), nextShort(), nextInt(), nextLong(), nextFloat(), nextDouble() ve next() metotları, belirteç (token) okuma metotları olarak bilinir,

çünkü bunlar sınırlayıcılarla (delimiter) ayrılmış belirteçleri (token) okurlar.

Varsayılan olarak, sınırlayıcılar boşluk (whitespace) karakterleridir.

Sınırlayıcılar (delimiter) için yeni bir desen ayarlamak için useDelimiter(String regex) metodunu kullanabiliriz.

whitespace karakterler: ' ', \t, \f, \r, \n

Örnek 5: Metin değişme

```java
import java.io.*;
import java.util.*;
public class MetinDegisme {
    public static void main(String[] args) throws Exception {
        // Komut satırı parametre kullanımını kontrol etme
        if (args.length != 4) {
            System.out.println(
            "Kullanim: java MetinDegisme kaynakDosya hedefDosya eskiMetin yeniMetin");
            System.exit(1);
        }
        // Kaynak dosya mevcut mu?
        File kaynakDosya = new File(args[0]);
        if (!kaynakDosya.exists()) {
            System.out.println("Kaynak dosya " + args[0] + " mevcut degil");
            System.exit(2);
        }
        // Hedef dosya mevcut mu?
        File hedefDosya = new File(args[1]);
        if (hedefDosya.exists()) {
            System.out.println("Hedef dosya " + args[1] + " zaten mevcut");
            System.exit(3);
        }
        try (
            // giris ve cikis dosyalarını oluşturma
            Scanner giris = new Scanner(kaynakDosya);
            PrintWriter cikis = new PrintWriter(hedefDosya);
        ) {
            while (giris.hasNext()) {
            String s1 = giris.nextLine();
            String s2 = s1.replaceAll(args[2], args[3]);
            cikis.println(s2);
            }
        }
    }
}
```

Örnek 5: Metin değişme

```
java MetinDegisme kaynakDosya hedefDosya eskiMetin yeniMetin
```

```
C:\Program Files\Java\jdk-16.0.1\bin\yeni2>javac MetinDegisme.java

C:\Program Files\Java\jdk-16.0.1\bin\yeni2>java MetinDegisme Bolumler.txt Bolumler_v1.txt Bilgisayar Yazilim
```

```
C:\Program Files\Java\jdk-16.0.1\bin\yeni2>java MetinDegisme Bolumler.txt Bolumler_v1.txt Bilgisayar Yazilim
Kaynak dosya Bolumler.txt mevcut degil

C:\Program Files\Java\jdk-16.0.1\bin\yeni2>java MetinDegisme Bolumler.txt Bolumler_v1.txt Bilgisayar Yazilim
Hedef dosya Bolumler_v1.txt zaten mevcut
```