

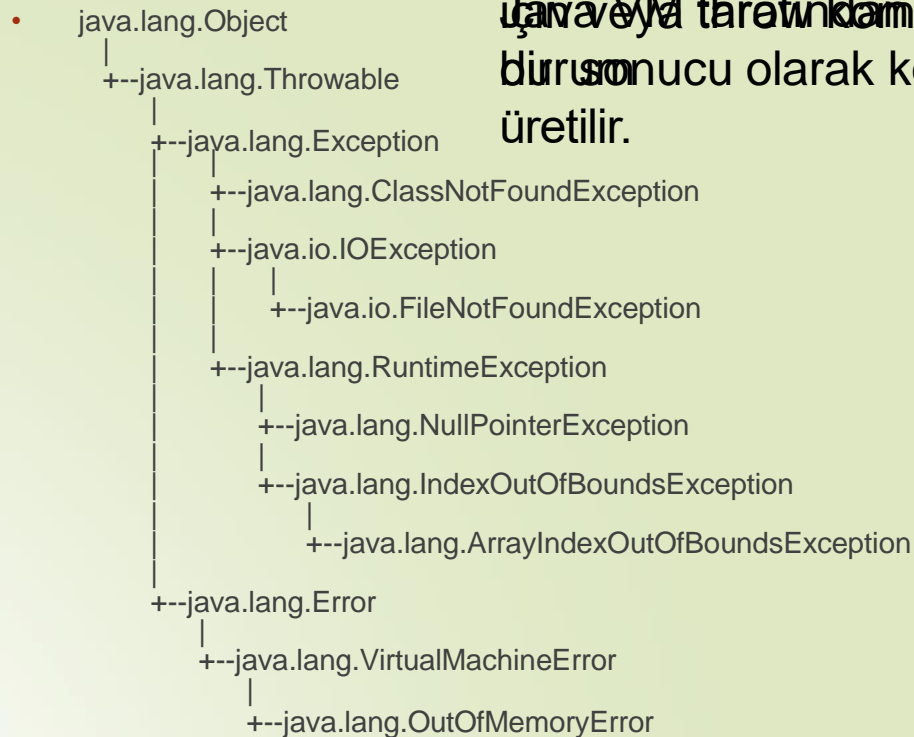
JAVA'DA İSTİSNALAR VE DOSYA İŞLEMLERİ



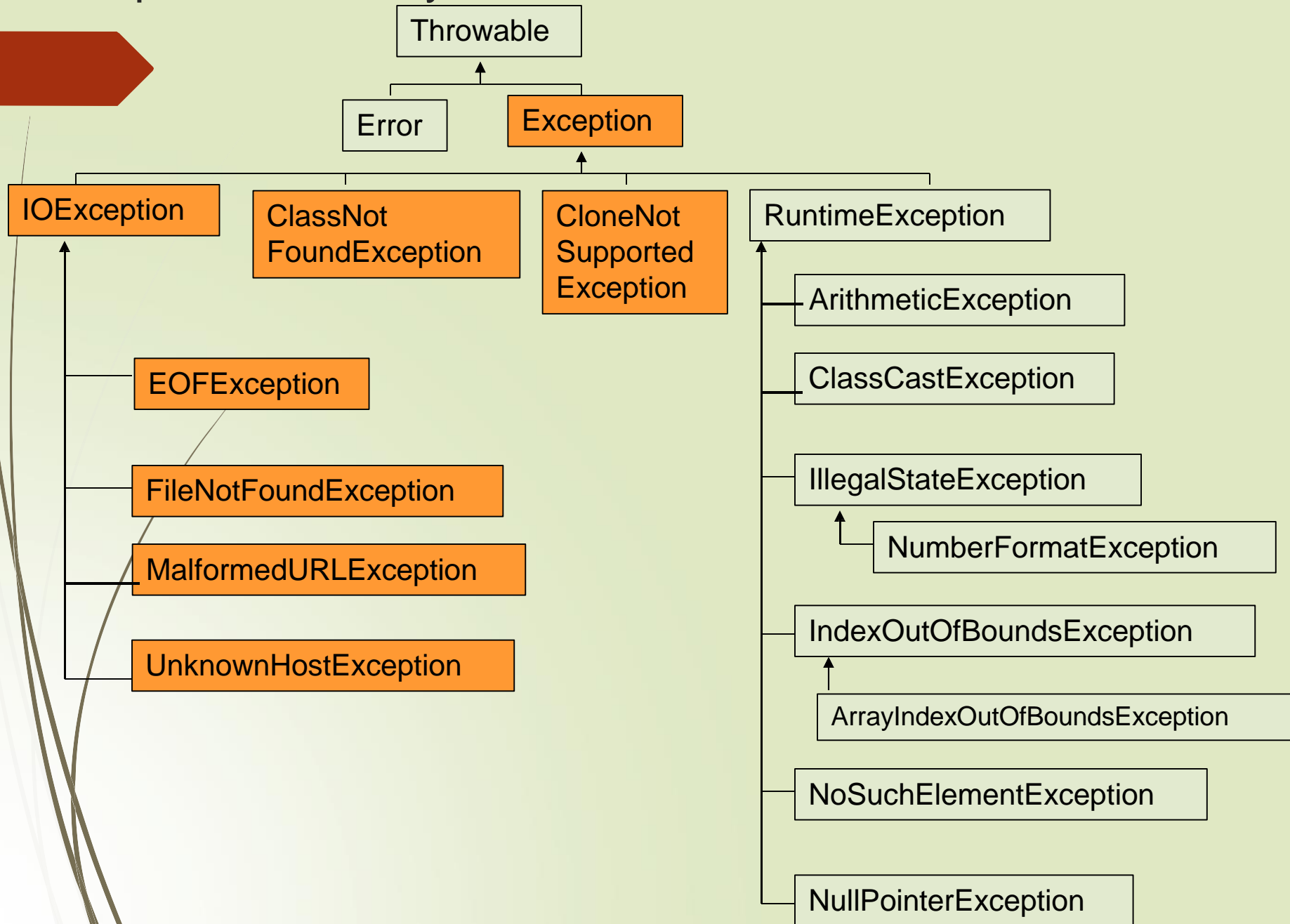
Exceptions - İstisnalar

İstisna çalışma zamanında oluşan bir hatadır.

Java VM tarafından beklenmeyen bir durum olarak kodunuz tarafından üretilir.




Exception Hierarchy



Try-Catch

```
❑ public FileReader(String fileName) throws  
   FileNotFoundException  
  
❑ try {  
    //komutlar  
    }  
    catch (Exception ex)  
    {  
        System.out.println("Hata Bulundu");  
        ex.printStackTrace();  
    }
```



Birden Fazla İstisnanın Yakalanması

```
try {  
    //...  
}  
catch ( FileNotFoundException e ) { System.out.println(  
    e.getMessage());  
}  
catch ( IOException e ) {  
    System.out.println( e + " IO EXCEPTION" );  
}  
catch ( Exception e ) {  
    System.out.println( e + " EXCEPTION" );  
}
```



Dosya G/Ç

- ☐ Giriş/Çıkış (G/Ç) bir programa giren ve çıkan verilerin genel gösterimine denir.
- ☐ Bir programa giriş klavyeden veya dosyadan yapılabilir.
- ☐ Programın çıktısı ise ekrana veya dosyaya yapılabilir.
- ☐ Eğer giriş/çıkış işlemleri dosya ile iletişim gerektiriyorsa Java'da bu işlemlere yönelik hazırlanmış özel sınıf ve nesnelerin kullanılması gerekir.

Dosya G/Ç işleminin gerekliliği

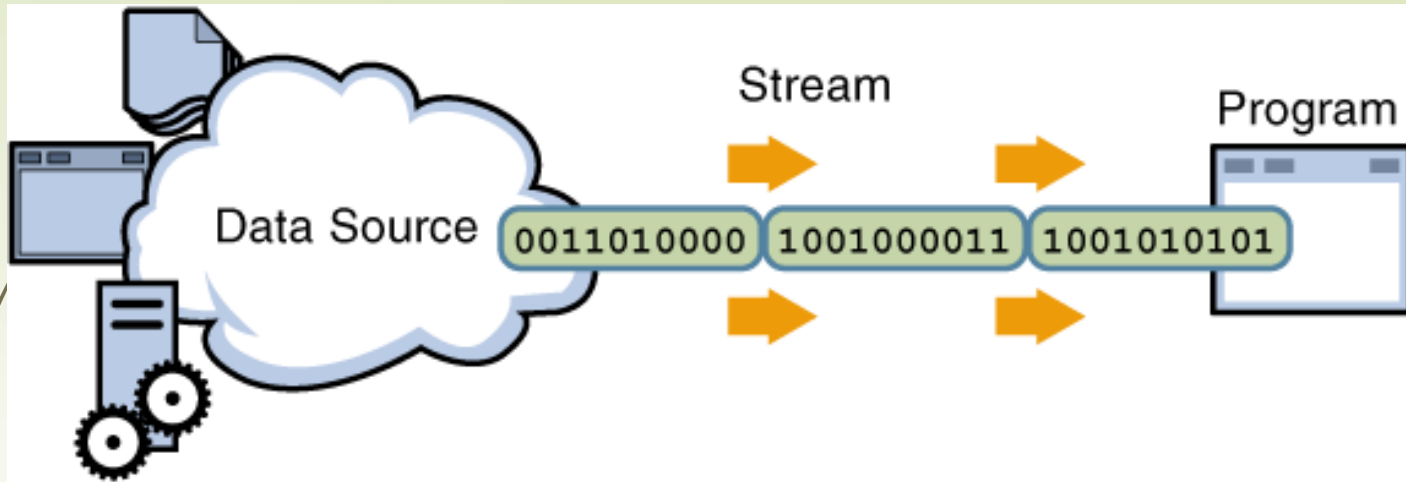
- ☐ Program sona erdiğinde kullanılan veriler kaybolur.
- ☐ Verileri kaybetmemek için dosyada saklanması gereklidir.
- ☐ Aynı şekilde klavyeden girilen verilerin de program çalıştırıldığında tekrar tekrar girilmesi yerine kaydedilip tekrar çalıştırıldığında okunarak elde edilmesi gerekir.

Metin Dosyaları Giriş/Çıkış İşlemleri

- ❑ Java'da metin dosyası çıkış işlemleri Prinwriter ve FileOutputStream sınıfları kullanılarak yapılır.
- ❑ Dosya çıkış işlemleri için bu sınıflar ile bir output stream oluşturulur.
- ❑ Giriş işlemleri ise BufferedReader ve FileReader sınıfları kullanılarak yapılır.
- ❑ Dosya giriş işlemleri için bu sınıflar ile bir input stream oluşturulur.

Stream (Akış)

- Bir akış byte ve bitler için programınız ile harici bir kaynak veya hedef arasında bir bağlantı olarak tanımlanır.
- Akış standart giriş/çıkış, dosya veya ağ bağlantısı olabilir



Dosya İşlemleri

- `import java.io.File;`
- `File dosya = new File(dosyaAdi);`
 - `dosya.getAbsolutePath()`
 - `dosya.getPath()`
 - `dosya.getName()`
 - `dosya.getParent()`
 - `dosya.exists()`
 - `dosya.canRead()`
 - `dosya.canWrite()`
 - `dosya.isDirectory()`
 - `dosya.isFile()`
 - `dosya.lastModified()`
 - `dosya.length()`

Yeni Dosya Oluşturma

```
File f = new File(dosyaAdi); // Dosya nesnesi
if(!f.exists()){           //Dosya zaten var mı
    f.createNewFile(); //Dosyayı oluştur
}
```

```
File dosya = new File("ornek.dat");
```

Bulunulan klasördeki
ornek.dat dosyasını açar

```
File dosya = new File
    ("C:/OrnekProgram/test.dat");
```

C:\OrnekProgram
klasöründeki **test.dat**
dosyasını açar.
Dosyanın adresi / ayracı
ile verilir.

Dosya İşlemleri

```
if ( dosya.exists( ) ) {
```


dosya değişkeni gerçekten var olan bir dosyayı mı gösteriyor.

```
if ( dosya.isFile() ) {
```

dosya bir dosya mı yoksa bir klasör mü.

```
File klasor = new  
    File("C:/Programlarım/java");  
  
String dosyalar[] = klasor.list();  
  
for (int i = 0; i < dosyalar.length; i++) {  
    System.out.println(dosyalar[i]);  
}
```

C:\Programlarım\java
verilen klasördeki bütün dosyaları listeler.



Dosya Silme

```
File f = new File(dosyaAdi); //Dosya Nesnesi
if(f.exists()){           //Dosya var mı
    f.delete();           //Dosyayı sil
}
```

Örnek: Dosya oluşturma

```
import java.io.*; public
class Dosyalar {
    public static void main(String[] args) {
        String path = "d:\\deneme.txt";
        File dosya = new File(path);
        try {
            if (!dosya.exists()) {
                dosya.createNewFile();
            } else {
                System.out.println("Dosya mevcuttur");
            }
        } catch (IOException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

Örnek: Bir klasördeki dosyaları listelemek

```
import java.io.*;

public class Dosyalar {
    public static void main(String[] args) {
        String path = "D:\\";

        File dosya = new File(path);

        if (dosya.isDirectory()) {
            File[] tumDosyalar = dosya.listFiles();
            for(int i=0;i<tumDosyalar.length;i++)
                System.out.println(tumDosyalar[i].getName());

        } else {
            System.out.println("Klasör veya dizin değil.");
        }
    }
}
```

Örnek: Dosya silme

```
import java.io.*; public
class Dosyalar {

    public static void main(String[] args) {
        String path = "d:\\example.txt";
        File dosya = new File(path);
        if(dosya.delete()){
            System.out.println("Dosya silinmiştir.");
        }
        else{
            System.out.println("Dosya silinemiştir.");
        }
    }
}
```


PrintWriter ile Dosyaya veri yazma

- Java'da bir metin dosyasına çıktı gönderebilmek için java.io kütüphanesinin içinde yer alan PrintWriter sınıfına ait println metodu kullanılır.
- Java'da PrintWriter sınıfını programda kullanabilmek için import java.io.* ile import edilmesi gereklidir.

java.io.PrintWriter

```
+PrintWriter(file: File)
+PrintWriter(filename: String)
+print(s: String): void
+print(c: char): void
+print(cArray: char[]): void
+print(i: int): void
+print(l: long): void
+print(f: float): void
+print(d: double): void
+print(b: boolean): void
```

- ☐ Belirlenen file nesnesi için PrintWriter nesnesi oluşturma
- ☐ Belirtilen dosya adı için PrintWriter nesnesi oluşturma
- ☐ String bilgiyi dosyaya yazdırma
- ☐ Karakteri dosyaya yazdırma
- ☐ Tam Sayıyı dosyaya yazdırma
- ☐ Long tipinde veriyi dosyaya yazdırma
- ☐ Float tipinde veriyi dosyaya yazdırma
- ☐ Double türünde veriyi dosyaya yazdırma
- ☐ Boolean veriyi dosyaya yazdırma

Örnek:

// Metin dosyasi olusturulmasi örnek programi

```
import java.io.*;

import java.util.*;

public class OrnekCikisMetinDosyasi
{
    public static void main(String[] args)
    {
        PrintWriter ciktiAkimi = null;
        String dosya = "ornek.txt";
        try
        {
            ciktiAkimi = new PrintWriter (new FileOutputStream(dosya));
        }
    }
}
```


PrintWriter Sınıfı


```
catch (FileNotFoundException hata)
{
    System.out.println("ornek.txt dosyasi olustururken hata oldu");
    System.exit(0); // Programdan normal çıkış.
}

System.out.print("Bir ornek cumle giriniz:");
Scanner klavye = new Scanner(System.in);
String cumle = klavye.nextLine();
ciktiAkimi.println("Klavyeden girdiginiz cumle: " + cumle);
ciktiAkimi.println("Dosyayi kapatabiliriz.");
ciktiAkimi.close();
System.out.println("Girilen cumle ornek.txt dosyasina yazildi.");
}
```

PrintWriter Sınıfı

- ❑ Java'da bir dosya açıldığında, bu dosyayla ilgili işlemler bitince dosyanın `close();` ile kapatılması gerekir.
- ❑ `ciktiAkimi.close();`
- ❑ Java'da varolan bir dosya metin dosyası üzerinde değişiklik yapmak istenirse, `PrintWriter` sınıfına overload yapılmış başka bir kurucuyu kullanmak gerekir.
- ❑ `ciktiAkimi = new PrintWriter(new FileOutputStream(dosya, true));`
- ❑ İkinci parametre dosyaya ekleme yapılıp yapılmayacağını belirler. `true` ise dosyaya ekleme yapılacağı bildirilir.
- ❑ Bir dosya açılmadan önce var olup olmadığı `File` sınıfına ait `exists()` metoduyla kontrol edilir.
- ❑ Bir nesneye ait bilgiler `toString()` ile yazdırılır.

- 
- ❑ Örnekte dosyaya çıktı göndermek için PrintWriter sınıfına ait ciktAkimi nesnesi oluşturulmuştur.
 - ❑ Program bu satırı çalıştırırken eğer ornek.txt dosyası varsa, varolan dosyanın içeriği silinip yeni içerik olarak gönderilen metin yazılır.
 - ❑ Eğer ornek.txt dosyası yoksa, yeni bir dosya oluşturulup gönderilen metin yazılır.
 - ❑ Örnekte PrintWriter nesne kurucusuna FileOutputStream sınıfına ait bir nesne gönderilmiştir.
 - ❑ FileOutputStream nesne kurucusu ise parametre olarak String türünde bir değer almaktadır ve dosyanın adını göstermektedir.
 - ❑ Dosya isimleri seçerken Java'da kullanılan isimlendirme kuralları değil işletim sistemleri kuralları geçerlidir.

- 
- ❑ Java'da dosya işlemleri sırasında FileNotFoundException kural dışı durumu oluşabilir.
 - ❑ Kural dışı durumu yakalamak için try-catch bloğu kullanılmalıdır.
 - ❑ Buradaki kural dışı durum sadece dosya açma işlemi yapan FileOutputStream nesne kurucusundan kaynaklanabilir. Bu yüzden try bloğu içerisine alınmalıdır.
 - ❑ FileNotFoundException tipi kural dışı durum, IOException sınıfına ait özel bir kural dışı durumdur.
 - ❑ Aşağıdaki satır ile PrintWriter sınıfına ait println() metoduyla dosyaya yazma yapılmaktadır.

```
ciktiAkimi.println("Klavyeden girdiginiz cumle: " + cumle);
```

Örnek: Dosya oluşturma ve veri yazma

```
import java.io.*; public
class Dosyalar {
    public static void main(String[] args) throws FileNotFoundException {
        String path = "d:\\example.txt";
        File dosya = new File(path);
        if(dosya.exists()) {
            System.out.println("Dosya zaten var");
            System.exit(0);
        }
        else {
            PrintWriter dosyayaz=new PrintWriter(dosya);
            dosyayaz.print("Ali BAL"); dosyayaz.println(90);
            dosyayaz.print("Ayşe DÖNMEZ");
            dosyayaz.println(60);
            dosyayaz.close();
        }
    }
}
```

Scanner ile okuma

- ❑ **java.util.Scanner** sınıfı konsoldan stringleri ve ilkel veri türlerini okumak için kullanıldı.
- ❑ Klavyeden veri okumak için Scanner sınıfı için aşağıdaki şekilde bir tanımlama yapmak gerekir.
 - ❑ `Scanner input = new Scanner(System.in);`
- ❑ Dosyadan okuma için, Scanner sınıfı aşağıdaki biçimde tanımlanır:
 - ❑ `Scanner input = new Scanner(new File(filename));`


```
public static void main(String[] args) {  
    File dosya=new File("ogrenci.txt");  
    PrintWriter cikti=null;  
    Scanner klavye=new Scanner(System.in);  
    try {  
        if(!dosya.exists()) dosya.createNewFile();  
        else{  
            cikti=new PrintWriter(dosya);  
            for (int j = 0; j < 3; j++) {  
                System.out.println(j+" . Ogrenci adi soyadi ve notu");  
                String ad=klavye.next();  
                String soyad=klavye.next();  
                int not=klavye.nextInt();  
                cikti.println(ad+" "+soyad+" "+not);  
            }  
            cikti.close();  
        }  
    }  
}
```

Örnek: Scanner ile dosyadan okuma

```
import java.io.*;
import java.util.Scanner;

public class Dosyalar {
    public static void main(String[] args) throws FileNotFoundException {
        String path = "d:\\example.txt";
        File dosya = new File(path);
        Scanner giris=new Scanner(dosya);
        while (giris.hasNext()) {
            String isim = giris.next();
            String soyad = giris.next();
            int notu = giris.nextInt();
            System.out.println(isim + " " + " " + soyad + " " + notu);
        }
        giris.close();
    }
}
```

Example.txt dosyasının içeriği

Ali BAL 90


Ayşe DÖNMEZ 60

Scanner ile Dosyadan okuma

- ☐ Belirlenen dosyadan veri taramak için Scanner sınıfı oluşturma
- ☐ Belirlenen string'ten veri taramak için Scanner sınıfı oluşturma.
- ☐ Scanner'ı kapatma.
- ☐ Scanner okunacak veriye sahip ise true gönderir.
- ☐ Scanner'dan bir string olarak sonraki bilgiyi alır
- ☐ Scanner'dan satır ayracı ile sonlanan bir satır okur.
- ☐ Byte okuma.
- ☐ Short veri okuma
- ☐ Int veri okuma.
- ☐ Long veri okuma
- ☐ Float veri okuma
- ☐ Double veri okuma
- ☐ Özel karakter ile ayrılan veri okuma


java.util.Scanner

```
+Scanner(source: File)
+Scanner(source: String)
+close()
+hasNext(): boolean
+next(): String
+nextLine(): String
+nextByte(): byte
+nextShort(): short
+nextInt(): int
+nextLong(): long
+nextFloat(): float
+nextDouble(): double
+useDelimiter(pattern: String):
Scanner
```



Scanner ile Dosya Okuma 1

```
try {  
    Scanner s = new Scanner( new  
    File(dosyaAdi));  
    String dosyalcerigi =  
    s.useDelimiter("\\A").next();  
    System.out.println(dosyalcerigi);  
    s.close();  
} catch (Exception e) {  
    e.printStackTrace();  
}
```



```
import java.io.*;

class TestScanner {

    public static void main (String[] args) throws IOException {

        //Scanner nesnesi olustur
        Scanner scanner = new Scanner(new File("ornek.data"));

        //integer oku
        int i = scanner.nextInt();

        //diger veri turleri de benzer sekilde okunur


        scanner.close();

    }
}
```



Scanner ile Dosya Okuma 2

```
try {  
    Scanner s = new Scanner( new File("test.txt"));  
    while(s.hasNext()){  
        String satir = s.nextLine();  
        System.out.println(satir);  
    }  
    scanner.close();  
} catch (Exception e) {  
    e.printStackTrace();  
}
```



```
import java.util.Scanner;
import java.io.File;
import java.io.IOException;

public class RakamlariOku
{
    public static void main(String[] args)
    {
        try
        {
            Scanner s = new Scanner( new File("rakamlar.dat") );
            while( s.hasNextInt() )
            {
                System.out.println( s.nextInt() );
            }

            s.close();
        }
        catch(IOException e)
        {
            System.out.println( e );
        }
    }
}
```





```
//Bir dosyaya 100 tane rastgele int yazan program
```

```
import java.io.PrintStream;  
import java.io.IOException;  
import java.io.File;
```

```
import java.util.Random;
```

```
public class DosyayaYaz  
{ public static void main(String[] args)  
  { try  
    { PrintStream writer = new PrintStream( new  
      File("sayilar.txt"));  
      Random r = new Random();  
      final int LIMIT = 100;  
  
      for(int i = 0; i < LIMIT; i++)  
      { writer.println( r.nextInt() );  
      }  
      writer.close();  
    }  
    catch(IOException e)  
    { System.out.println("Bir hata olustu");  
    }  
  }  
}
```


- 
- 
- Scanner kullanarak sadece kelimeleri okumak için:
 - `Scanner s = new Scanner(new File("test.txt")).useDelimiter("\\W");`



BufferedReader Sınıfı

- Java'da bir metin dosyasından okuma yapmak için java.io kütüphanesinin içinde yer alan BufferedReader sınıfı kullanılır.
- BufferedReader sınıfına ait bir nesne oluştururken metin dosyasının adı doğrudan girilmez. FileReader adlı bir sınıftan oluşturulan nesne kullanılır.
- Metin dosyasından bir satır okumak için readLine() metodu kullanılır.

Örnek

```
import java.io.*;

public class OrnekGirisMetinDosyasi
{
    public static void main(String[] args)
    {
        try {
            BufferedReader girisAkimi = new BufferedReader(new
            FileReader("ornek.txt"));
            String satir = null;
            int sayac = 0;
            satir = girisAkimi.readLine( );
            while (satir != null) {
                sayac++;
                System.out.print("ornek.txt dosyasinin " + sayac);
                System.out.print("satirinda:" + sayac);
                System.out.println("\n" + satir + "\" yazmaktadır. ");
                satir = girisAkimi.readLine( );
            }
        }
    }
}
```

Örneğin devamı

```
girisAkimi.close( );  
}  
catch(FileNotFoundException e)  
{  
    System.out.print("ornek.txt dosyasi bulunamadi");  
    System.out.println("veya acilamadi.");  
}  
catch(IOException e)  
{  
    System.out.print("ornek.txt dosyasindan veri girisinde");  
    System.out.println(" hata olustu.");  
}  
}  
}
```



Dosya Okuma 1

```
try {  
    FileInputStream fis = new FileInputStream(dosyaAdi);  
  
    int ch = 0;  
    while (ch != -1) {  
        ch = fis.read();  
        char karakter = (char)ch;  
        System.out.print(karakter);  
    }  
    fis.close();  
} catch (Exception e) {  
    e.printStackTrace();  
}
```

```
//dosya ve stream olustur
File dosya = new File("ornek.data");
FileInputStream girisStream = new FileInputStream(dosya);

//verileri okumak icin bir dizi olustur
int dosyaBoyutu = (int)dosya.length();
byte[] byteDizisi = new byte[dosyaBoyutu];

//veriyi oku ve goster
girisStream.read(byteDizisi);
for (int i = 0; i < dosyaBoyutu; i++) {
    System.out.println(byteDizisi[i]);
}

//okuma bitti stream'I kapat
girisStream.close();
```




Dosya Okuma 2

```
try {  
    FileReader fr = new FileReader(dosyaAdi);  
    BufferedReader br = new BufferedReader(fr);  
    while(br.ready()){  
        String satir = br.readLine();  
        System.out.println(satir);  
    }  
    fr.close();  
    br.close();  
} catch (Exception e) {  
    e.printStackTrace();  
}
```



Dosya Yazma 1

```
try {  
    FileOutputStream fos = new  
        FileOutputStream(dosyaAdi);  
    String yazi = "Bu satir dosyaya yazilacak\naltina da  
        bu satir yazilacak.";  
    fos.write(yazi.getBytes());  
    fos.flush();  
    fos.close();  
} catch (Exception e) {  
    e.printStackTrace();  
}
```

```
//Yazilacak dosyayi olustur
File cikisDosyasi = new File("sample1.data");

FileOutputStream
    cikisStream = new FileOutputStream( cikisDosyasi );

//kaydedilecek veri
byte[] byteDizisi = {10, 20, 30, 40,
                    50, 60, 70, 80};

//verileri stream'e yaz
cikisStream.write( byteDizisi );

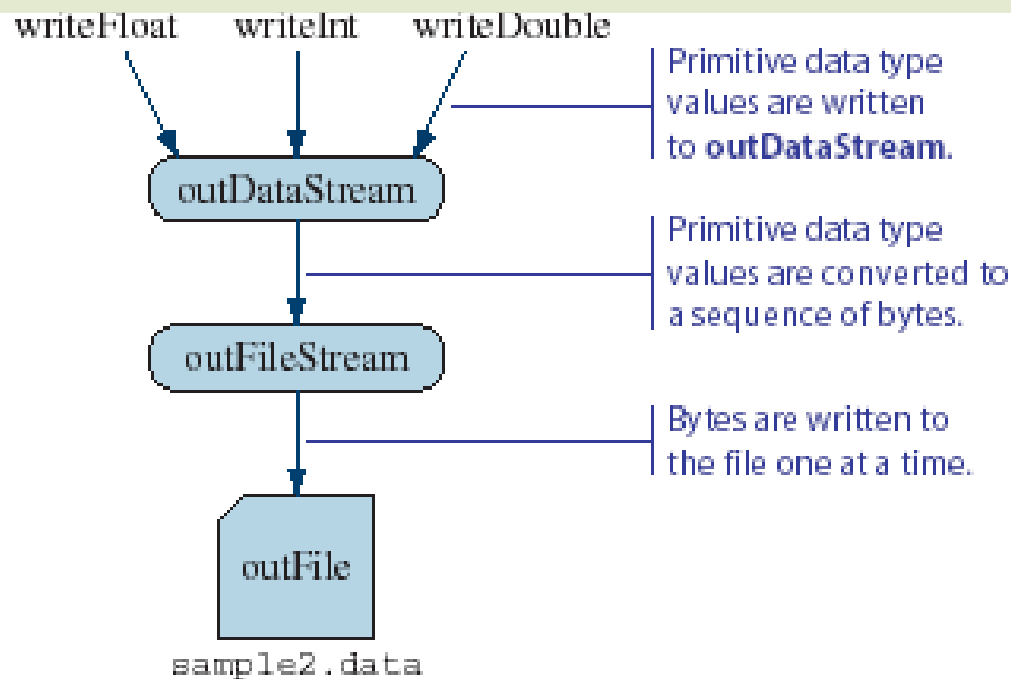
//stream kapat
cikisStream.close();
```

DataOutputStream

42

Typical sequence:

```
File cikisDosyasi = new File( "ornek.data" );  
FileOutputStream cikisDosyasiStream = new  
FileOutputStream(cikisDosyasi);  
DataOutputStream outDataStream = new  
DataOutputStream(cikisDosyasiStream);
```

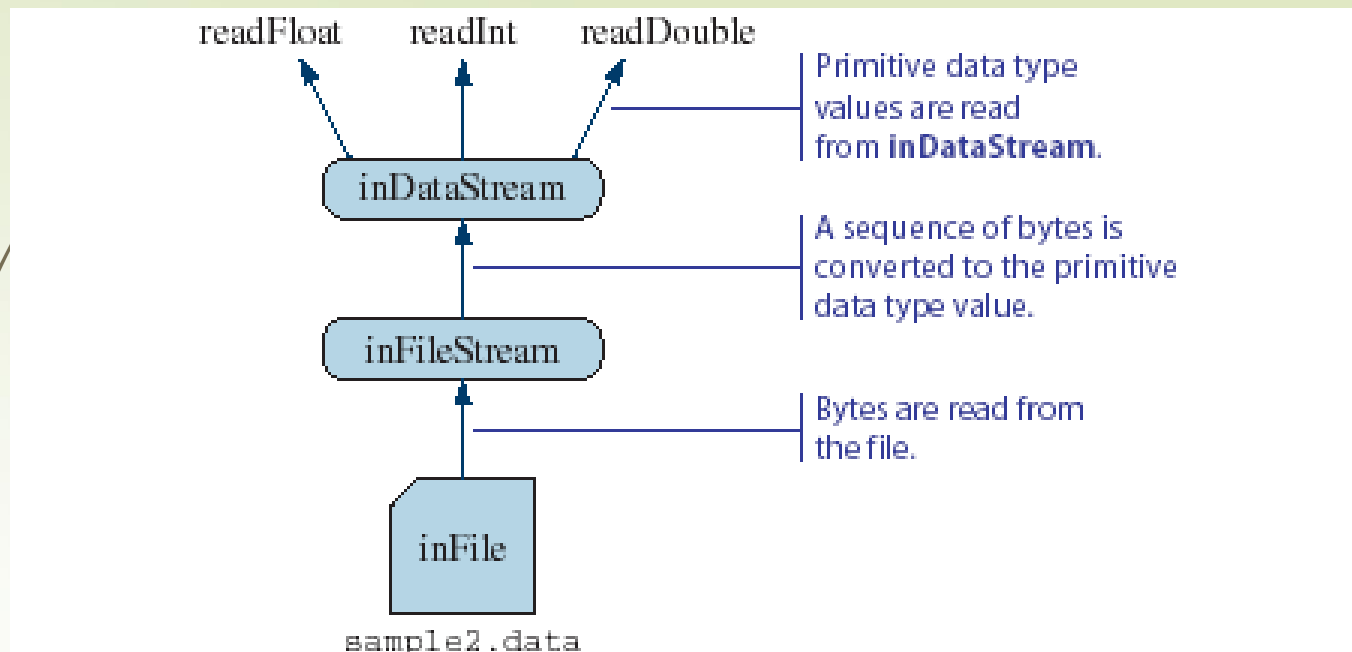


DataInputStream

43

□ Typical sequence:

```
File okunacakDosya = new File( "sample2.data" );  
FileInputStream okuDosyaStream = new FileInputStream(inFile);  
DataInputStream inDataStream = new DataInputStream(okuDosyaStream);
```





Dosya Yazma 2

```
try {  
    FileWriter fw = new FileWriter(dosyaAdi);  
    BufferedWriter bw = new BufferedWriter(fw);  
    bw.write("Bu satiri yaz\nnyeni satira gec.");  
    bw.flush();  
    bw.close();  
} catch (Exception e) {  
    e.printStackTrace();  
}
```