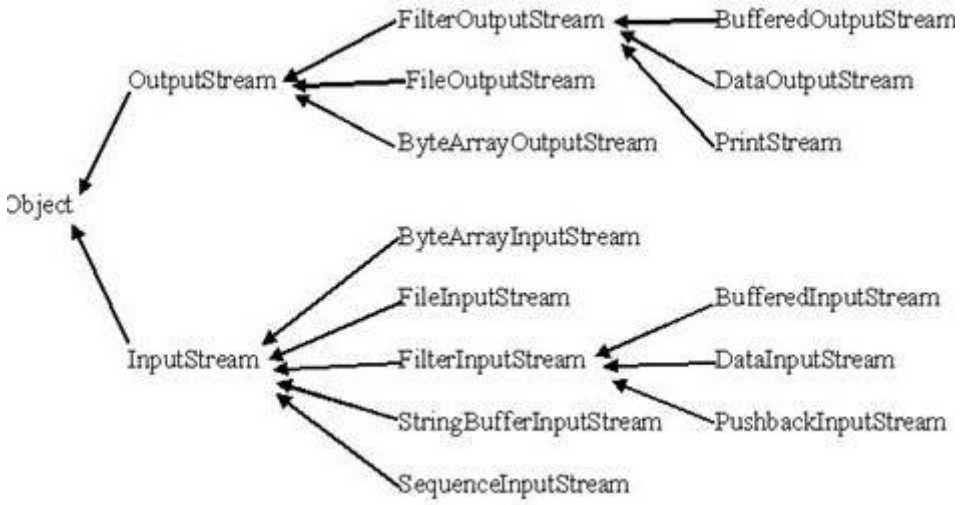
PART 15

15.1 Dosyaları Okuma ve Yazma

Stream, bir veri dizisi olarak tanımlanabilir. InputStream, bir kaynak dosyadan veri okumak için ve OutputStream bir hedef dosyaya veri yazmak için kullanılır.

Burada, Input ve Output stream'ler ile ilgili sınıfların hiyerarşisi gösterilmektedir.



Bu bölümde, iki önemli stream olan **FileInputStream** ve **FileOutputStream** ele alınacaktır.

15.2 FileInputStream

Bu stream, dosyalardan veri okumak için kullanılır. Nesneler, new anahtar sözcüğü kullanılarak oluşturulur ve birçok türde constructor mevcuttur.

Aşağıdaki constructor, dosya okuma amacıyla bir input stream nesnesi oluşturmak için türü string olan bir dosya ismi alır.

```
InputStream f = new FileInputStream("C:/java/hello");
```

Aşağıdaki constructor, dosya okuma amacıyla bir input stream nesnesi oluşturmak için bir dosya nesnesi alır. İlk olarak, File() metodunu kullanarak aşağıdaki gibi bir dosya nesnesi oluştururuz:

```
File f = new File("C:/java/hello");
InputStream f = new FileInputStream(f);
```

Bir kere , *InputStream* nesnesi elinizde mevcutsa, stream okumak için kullanılabileceğiniz veya stream üzerinden farklı işlemler yapabileceğiniz listede gösterilen yardımcı metotları kullanabilirsiniz.

Methods with Description
public void close() throws IOException{} Bu metot , file output stream'i kapatır. Dosya ile ilgili sistem kaynaklarını bırakır. IOException atar.
public int read(byte[] r) throws IOException{} Bu metot, bir input stream'den dizinin içine r.length bit okur. Okunan bitlerin toplam sayısını döndürür. Eğer dosya sonu -1 ise döndürülecektir.
public int available() throws IOException{} Bir file input stream'den okunabilen bit sayısını verir. int türünde değişken döndürür.

15.3 FileOutputStream

FileOutputStream, bir dosya oluşturup içine veri yazmak için kullanılır. Stream, output için açılmadan önce, bir dosya yaratacaktır.(eğer önceden oluşturulmadıysa)

Burada, FileOutputStream nesnesi oluşturmak için kullanılan iki constructor gösterilmektedir.

Aşağıdaki constructor, dosyaya yazma amacıyla bir input stream nesnesi oluşturmak için türü string olan bir dosya ismi alır.

```
OutputStream f = new FileOutputStream("C:/java/hello")
```

Aşağıdaki constructor, dosyaya yazma amacıyla bir input stream nesnesi oluşturmak için bir dosya nesnesi alır. İlk olarak, File() metodunu kullanarak aşağıdaki gibi bir dosya nesnesi oluştururuz:

```
File f = new File("C:/java/hello");
OutputStream f = new FileOutputStream(f);
```

Bir kere , *OutputStream* nesnesi elinizde mevcutsa, stream'e yazmak için kullanılabileceğiniz veya stream üzerinden farklı işlemler yapabileceğiniz listede gösterilen yardımcı metotları kullanabilirsiniz.

Methods with Description
public void close() throws IOException{} Bu metot, file output stream'i kapatır. Dosya ile ilgili sistem kaynaklarını bırakır. IOException atar
public void write() throws IOException{} Bu metot, output stream'e belirlenen bitte veri yazar.

Örnek:

Aşağıda, InputStream ve OutputStream örnekleri gösterilmektedir:

```
import java.io.*;

public class fileStreamTest{

   public static void main(String args[]){

   try{
      char[] data = {'a','b','c','d','e'};
      OutputStream outfile = new FileOutputStream("C:/test.txt");
      for(int x=0; x < data.length ; x++){
         outfile.write( data[x] ); // writes the bytes
      }
      outfile.close();

      InputStream infile = new FileInputStream("C:/test.txt");
      int size = infile.available();

      for(int i=0; i< size; i++){
         System.out.print((char) infile.read() + "  ");
      }
      infile.close();

   }catch(IOException e){
      System.out.print("Exception");
   }
   }
}
```

Yukarıdaki kod, test.txt dosyasını oluşturup, verilen karakterleri içine yazacaktır. Aynısı, stdout ekranında gösterilecektir.

15.4 Dosya Navigasyonu ve I/O

Dosya Navigasyonu ve I/O temellerini öğrenmek için inceleyeceğimiz birkaç sınıf vardır.

- FileReader Sınıfı
- FileWriter Sınıfı
- File Sınıfı

File Reader

Bu sınıf, InputStream sınıfından miras alır. FileReader, karakter streamlerini okuma için kullanılır.

Aşağıdaki syntax , okumak için File nesnesi verilen yeni bir FileReader nesnesi oluşturur.

```
FileReader(File file)
```

File Writer

Bu sınıf, OutputStream sınıfından miras alır. Bu sınıf, karakter streamlerini yazma için kullanılır.

Aşağıdaki syntax, File nesnesi verilen bir FileWriter nesnesi oluşturur.

```
FileWriter(File file)
```

Örnek:

Aşağıda bu sınıflarla ilgili örnek gösterilmektedir:

```
import java.io.*;

public class FileRead{

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

      File file = new File("Hello1.txt");
      // creates the file
      file.createNewFile();
      // creates a FileWriter Object
      FileWriter writer = new FileWriter(file);
```

```
      // Writes the content to the file
      writer.write("This\n is\n an\n example\n");
      writer.flush();
      writer.close();

      //Creates a FileReader Object
      FileReader fr = new FileReader(file);
      //Creates a LineNumberReader Object
      LineNumberReader lnreader = new LineNumberReader(fr);
      String line = "";
      while ((line = lnreader.readLine()) != null) {
            System.out.println(lnreader.getLineNumber() + ": " + line);
      }
      fr.close();
   }
}
```

Bu aşağıdaki sonucu üretecektir:

```
1: This
2:  is
3:  an
4:  example
```

15.4 Java Dizinleri

Dizinleri Oluşturma

Burada, dizin oluşturmak için kullanılan iki adet File yardımcı metotları gösterilmektedir:

- **mkdir()** metod dizin oluşturur, başarılı(succes) durumda true değer, başarısız(failure) durumda iste false değer döndürür. Failure, File nesnesinde belirlenen yolun zaten var olması veya bütün yolun var olmamasından dolayı dizinin oluşturalamamasını belirtir.
- **mkdirs()** metodu, hem dizini oluşturur hem de dizinin bütün üst öğelerini oluşturur.

Aşağıdaki örnek, “c:/java/example/folder” dizinini oluşturur:

```
import java.io.File;

public class CreateDir {
   public static void main(String args[]) {
      String dirname = "c:/java/example/folder";
      File d = new File(dirname);
      // Create directory now.
      if(d.mkdirs())
       { System.out.print(“created”); }
```

```
      else
       { System.out.print("error! Not created"); }
  }
}
```

Not: Java, UNIX ve Windows'taki yol ayırıcılarının kurallara göre olmasına otomatik olarak dikkat eder. Eğer , Java'nın Windows versiyonunda ileri slash(/) kullanırsanız, yol yinede doğru şekilde çözümlenecektir.

Dizinleri okuma

Dizin, diğer dosyaların ve dizinlerin listesini içeren bir File nesnesidir. Bir File nesnesi yarattığınızda ve bu bir dizinse, isDirectory() metodu true değer döndürecektir.

Nesnenin içindeki diğer dosyalar ve dizinlerin listesini çıkartmak için list() metodunu çağırabilirsiniz. Burada gösterilen program, dizin içeriğini almak için list() metodunun ne şekilde kullanacağınızı örneklemektedir.

```
import java.io.File;

public class DirList {
   public static void main(String args[]) {
      String dirname = "c:/mysql";
      File f1 = new File(dirname);
      if (f1.isDirectory()) {
         System.out.println( "Directory of " + dirname);
         String s[] = f1.list();
         for (int i=0; i < s.length; i++) {
            File f = new File(dirname + "/" + s[i]);
            if (f.isDirectory()) {
               System.out.println(s[i] + " is a directory");
            } else {
               System.out.println(s[i] + " is a file");
            }
         }
      } else {
         System.out.println(dirname + " is not a directory");
    }
  }
}
```

Bu aşağıdaki sonucu üretecektir:

```
Directory of /mysql
bin is a directory
lib is a directory
demo is a directory
test.txt is a file
README is a file
index.html is a file
include is a directory
```