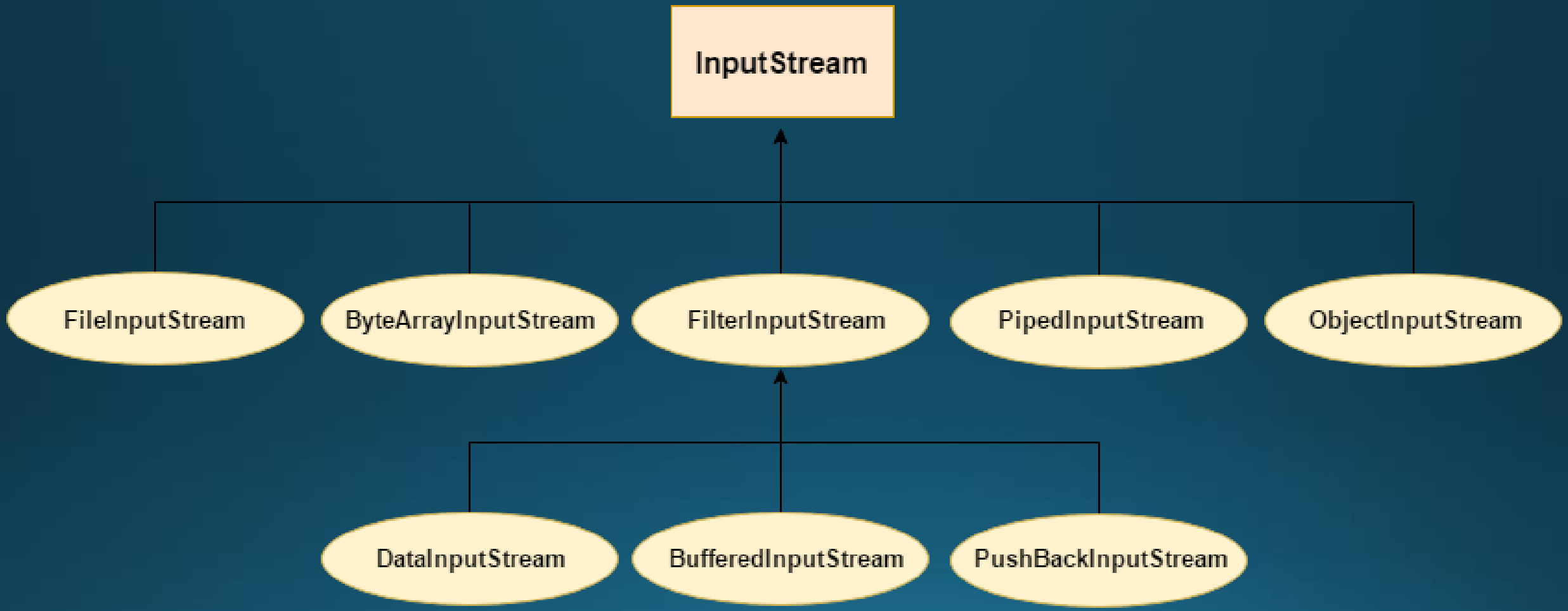
Java'da Dosya İşlemleri Sınıfları

- Bu fonksiyonların aralarındaki farkı daha net bir şekilde ortaya koyabilmek adına bunları 4 ana kategoride inceleyelim.

1. InputStream
2. OutputStream
3. Reader
4. Writer

InputStream

- InputStream sınıfı byte akışını temsil eden bir abstract sınıftır. InputStream abstract bir sınıf olduğu için kendi başına kullanışlı değildir o yüzden InputStream'a ait alt sınıflar veri okumak için kullanılır.
- InputStream alt sınıfları : FileInputStream, ByteArrayInputStream, ObjectInputStream
- InputStream sınıfına ait metotlar:
  - read() : Dosyadan tek baytlık veri okur.
  - read(byte[] array) : Dosyadan verileri bayt cinsinde okur ve belirtilen dizide depolar
  - available() : Kullanılabilir bayt sayısını verir.
  - skip(int) : Girilen parametre kadar bayt atlamaya yarar.


InputStream
FileInputStream
ByteArrayInputStream
FilterInputStream
PipedInputStream
ObjectInputStream
DataInputStream
BufferedInputStream
PushBackInputStream

FileInputStream

- Dosyadan verileri bayt bayt okumak için kullanılır.

  FileInputStream input = new FileInputStream(File fileObject);

```
FileInputStream input = new FileInputStream("input.txt");
System.out.println("Dosyadaki veriler: ");
int i = input.read();                    // İlk baytı okur
while (i != -1) {                        // EOF'a kadar okur
System.out.print((char) i);              // ASCII koda göre char dönüşümü
i = input.read();                        // Dosyadan sonraki baytı okur
}
input.close();
```

ByteArrayInputStream

- Verilen bayt dizisinin tamamını ya da bir bölümünü okur.

```
ByteArrayInputStream input = new ByteArrayInputStream(byte[] arr);
ByteArrayInputStream input = new ByteArrayInputStream(byte[] arr, int start, int length);
```

- ByteArrayInputStream sınıfı bir bayt dizisinden okur. Oysa FileInputStream ve FilterInputStream, verileri dosyadan okuyordu. Ortak noktaları üçünün de bayt bayt okuması.

FilterInputStream

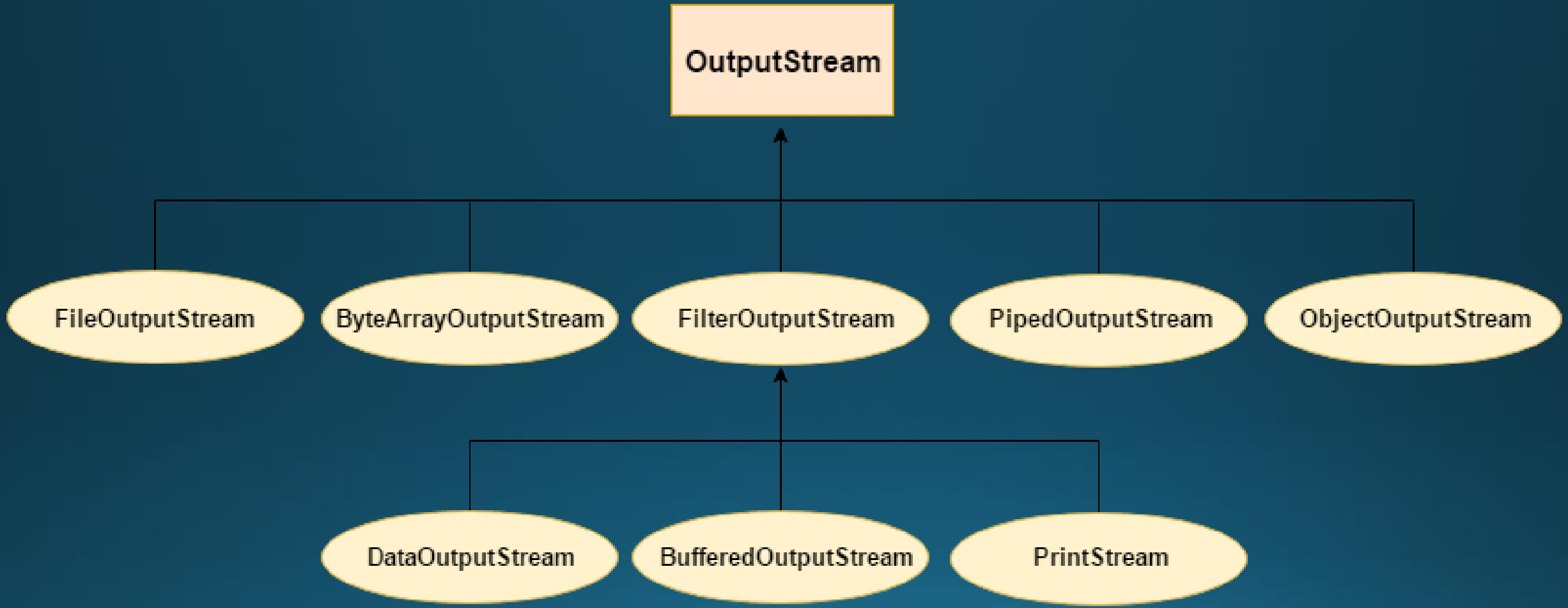
- Dosyadan verileri bayt olarak, daha verimli bir şekilde okur.

```
FilterInputStream input = new FilterInputStream(File fileObject);
```

- Kullanımının temel amacı, kullanıcının isteğine göre faydalı farklı alt sınıflar bulundurmasıdır. Örneğin: BufferedInputStream, CheckedInputStream, DataInputStream vs. gibi.
- Örneğin BufferedInputStream, 8192 baytlık bir dahili arabellek tutar. BufferedInputStream'deki okuma işlemi sırasında, diskten bir bayt yığını okunur ve dahili tamponda saklanır. Ayrıca dahili arabellekten baytlar ayrı ayrı okunur. Böylelikle diskle iletişim sayısı azalır. BufferedInputStream kullanarak bayt okumanın daha hızlı olmasının nedeni budur.
- Mesela bazı verilerden tüm çoklu boşlukları kaldırmak isteyelim. FilterInputStream'in kendi alt sınıfını oluşturup read() metodunu override edebiliriz. Dikkat etmemiz gereken nokta, bu gibi sınıflar son kullanıcının doğrudan kullanması için tasarlanmamıştır. Geliştirici tarafı için kullanılır.

OutputStream

- Byte tabanlıdır. Byte dizilerin ya da byte'ların dosyaya tek tek yazılmasını sağlar.
- OutputStream alt sınıfları : FileOutputStream, ByteArrayOutputStream, ObjectOutputStream
- OutputStream sınıfına ait metotlar:
  - write() : Çıkış akışına tek baytlık veri yazar.
  - write(byte[] array) : Çıkış akışına belirtilen byte dizisini yazar.
  - flush() : Çıkış akışı bağlantısını temizlemeye yarar. close() metodundan önce kullanılır.
  - close() : Bağlantıyı sonlandırmak için kullanılır.


OutputStream
FileOutputStream
ByteArrayOutputStream
FilterOutputStream
PipedOutputStream
ObjectOutputStream
DataOutputStream
BufferedOutputStream
PrintStream

FileOutputStream

- Dosyaya verileri tek bayt olarak ya da bayt dizisi şeklinde yazmak için kullanılır.

```
FileOutputStream output = new FileOutputStream(File fileObject);
```

```
FileOutputStream fout=new FileOutputStream("D:\\testout.txt");
String s="Java'da Dosyaya Yazma İşlemi";
byte b[]=s.getBytes();                    //String ifadeyi byte dizisine atar
fout.write(b);
fout.close();
```

Çıktı olarak dosyaya belirtilen stringi yazmış oluruz.

ByteArrayOutputStream

- Bir bayt dizisini kopya olarak saklar. Sakladığı diziyi kendine ait writeTo(OutputStream obj) metodu ile dosyaya yazabilir.

```
ByteArrayOutputStream output = new ByteArrayOutputStream();
```

```
FileOutputStream fout1=new FileOutputStream("D:\\f1.txt");
FileOutputStream fout2=new FileOutputStream("D:\\f2.txt");
ByteArrayOutputStream bout=new ByteArrayOutputStream();
bout.write(65);
bout.writeTo(fout1);
bout.writeTo(fout2);
```

Çıktı : İki dosyaya da ‘A’ yazdırmış oluruz.

FilterOutputStream

- Dosyaya verileri bayt bayt yazar.

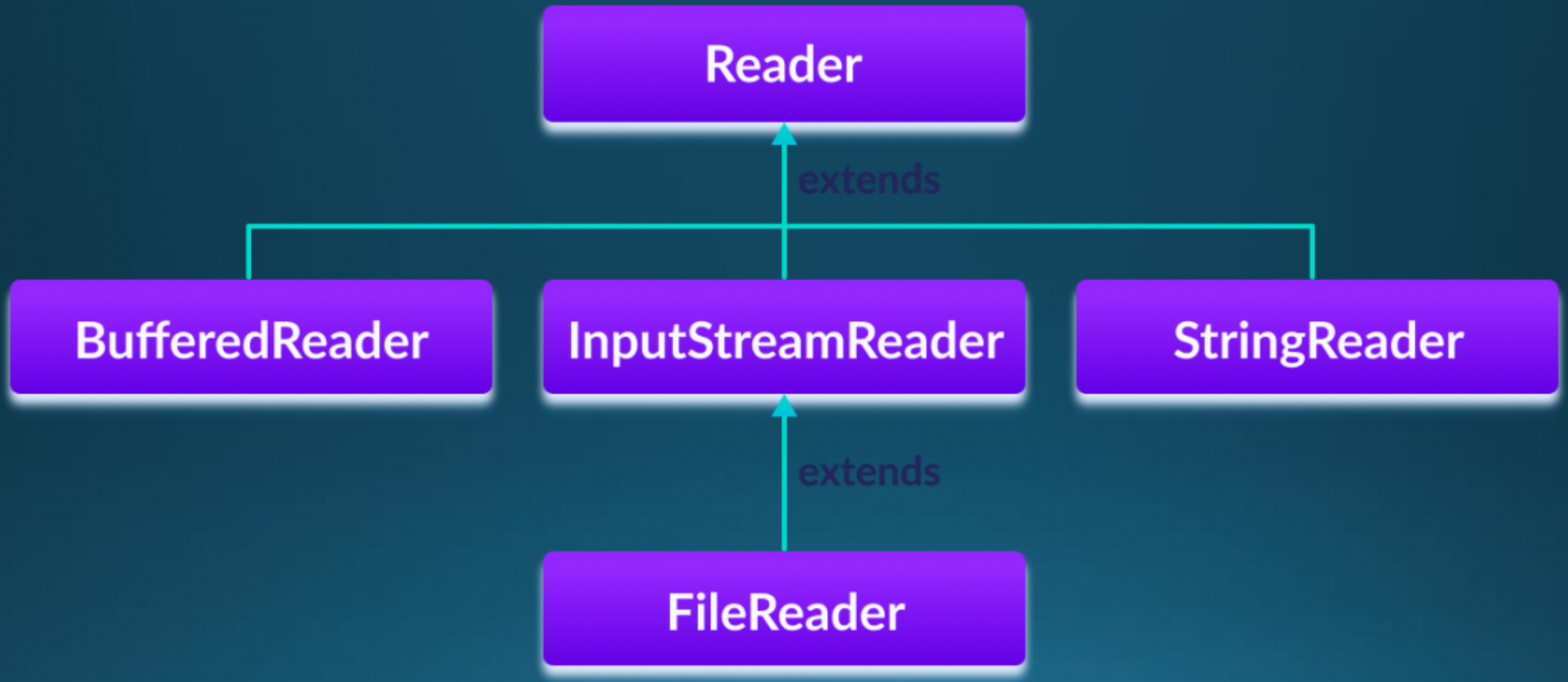
```
FilterOutputStream output = new FilterOutputStream(File fileObject);
```

- Kullanımının temel amacı, kullanıcının isteğine göre faydalı farklı alt sınıflar bulundurmasıdır. FilterInputStream'da değindiğimiz gibi Filter sınıfının ana özelliği geliştiricinin ihtiyacına uygun olarak alt sınıflar türetebilmesidir. Son kullanıcı yerine geliştiricinin kullandığı bir sınıftır.

- Özet : FileOutputStream ile ByteArrayOutputStream arasındaki fark, FileOutputStream direkt olarak dosyaya yazmaktan sorumluydu. ByteArrayOutputStream ise depolama gibi düşünülebilir. İçinde tuttuğu byte veriyi FileOutputStream aracılığıyla istenen dosyaya yazabilir. FilterOutputStream için ise FileOutputStream'in daha fonksiyonel hali diyebiliriz.

Reader

- Karakter tabanlıdır. Dosyadan byte, karakter ya da string verisi almayı sağlayan sınıftır.
- Reader alt sınıfları : BufferedReader, InputStreamReader, StringReader
- Reader sınıfına ait metotlar:
  - read() : Dosyadan tek karakterlik veri okur.
  - read(char[] array) : Dosyadan karakter dizisi okur.
  - ready() : Dosyadan veri okumaya hazır olunduğunda kullanılan metot.
  - reset() : Akışı yeniden başlatır.


Reader
extends
BufferedReader
InputStreamReader
StringReader
extends
FileReader

BufferedReader

- Dosyadan verileri string olarak verimli bir şekilde okumaya yarar. Parametre olarak okuma sınıfı alır. Kendine ait readLine() metodu bulunur.

```
BufferedReader br= new BufferedReader(Reader rd);
```

```
InputStreamReader r=new InputStreamReader(System.in);
BufferedReader br=new BufferedReader(r);
System.out.println("İsminizi girin:");
String name=br.readLine();
System.out.println("Hoş geldin "+name);
//Kullanıcının ismini çıktı olarak ekrana verir.
```

InputStreamReader

- Bayt veri akışı ile karakter veri akışı arasında köprü görevi gören bir Reader sınıfıdır. Parametre olarak dosya, konsol gibi veri ortamlarını alır. Buna ek olarak, kullanmak istediğimiz karakter setini de ikinci parametre olarak girebiliriz.

```
InputStreamReader inputstreamreader= new InputStreamReader(InputStream in, Charset cs);
```

```
InputStream stream = new FileInputStream("file.txt");
Reader reader = new InputStreamReader(stream);
int data = reader.read();
while (data != -1)
{
        System.out.print((char) data);
        data = reader.read();
}
```

Çıktı : Dosyadaki veriyi yazdırmış oluruz.

StringReader

- Parametre olarak string alan Reader sınıfıdır.

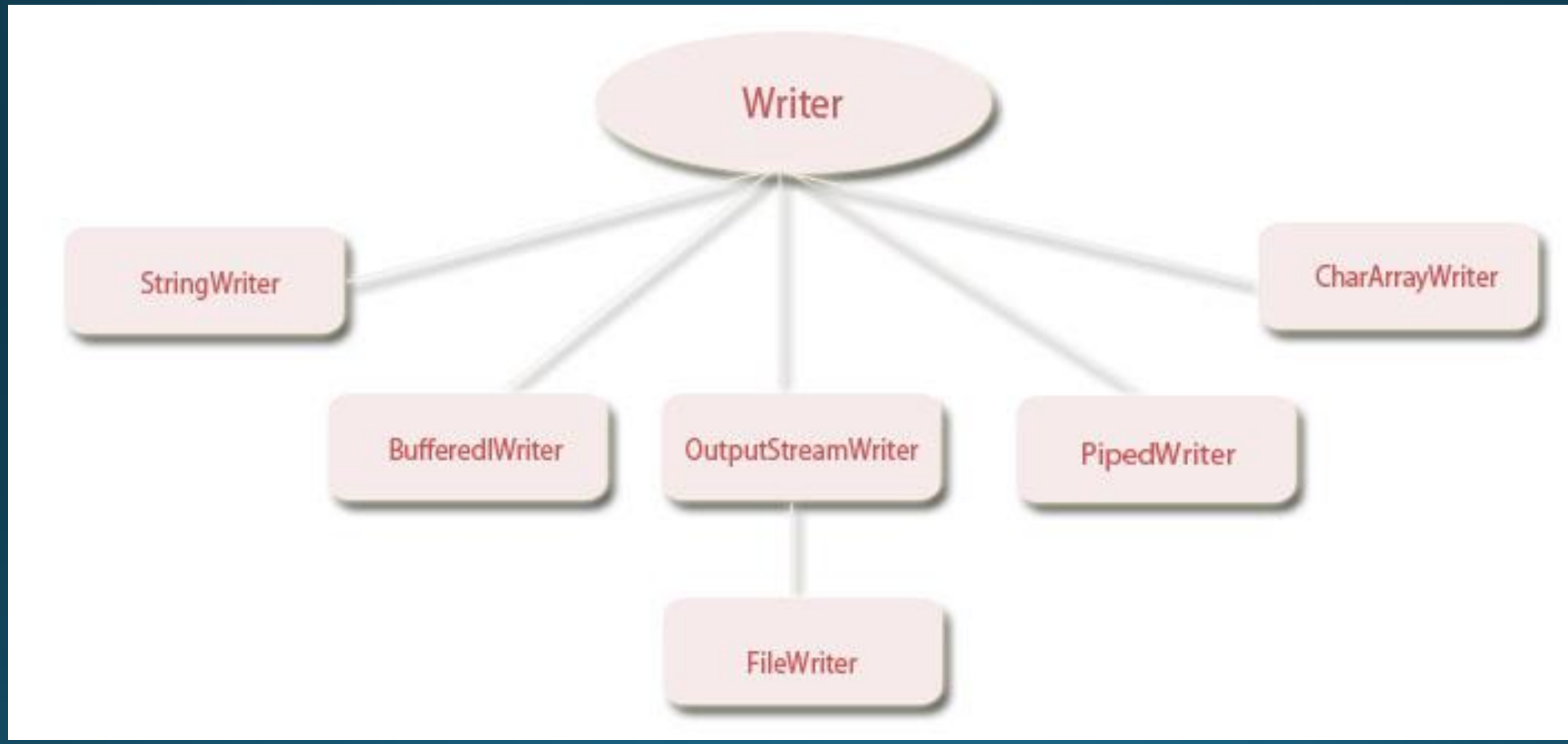
```
StringReader reader = new StringReader(String);


String st = "Merhaba\nDünya.";
StringReader reader = new StringReader(st);
int k=0;
while((k=reader.read())!=-1
   System.out.print((char)k);
```

- Özet : BufferedReader sınıfı, parametre olarak diğer okuyucuları alır. Daha verimli veri okumayı sağlar. InputStreamReader bayt türünde veri okur ve ikinci parametre olarak aldığı karakter setine çevirir. StringReader ise string parametreli bir okuyucudur.

Writer

- Karakter tabanlıdır. Dosyaya ya da ekrana byte, karakter ya da string verisi yazmayı sağlayan sınıftır.
- Writer alt sınıfları : BufferedWriter, OutputStreamWriter, StringWriter
- Writer sınıfına ait metotlar:
  - append(char) : Yazıcının sonuna karakter ekler.
  - write() : Parametre değerini yazar. Parametre olarak String, char[], int alabilir.

```
Writer w = new FileWriter("output.txt");
String content = "Java'da dosya işlemleri";
w.write(content);
w.close();
System.out.println("İşlem başarılı.");
```


Writer
StringWriter
CharArrayWriter
BufferedIWriter
OutputStreamWriter
PipedWriter
FileWriter

BufferedWriter

- Dosyadan verileri string olarak verimli bir şekilde okumaya yarar. Parametre olarak yazma sınıfı alır. Kendine ait newLine() metodu bulunur.

    BufferedWriter bw= new BufferedWriter(Writer wt);

```
FileWriter writer = new FileWriter("D:\\testout.txt");
BufferedWriter buffer = new BufferedWriter(writer);
buffer.write("Merhaba Dünya.");
buffer.close();
System.out.println("İşlem başarılı.");
```

OutputStreamWriter

- Bayt veri akışı ile karakter veri akışı arasında köprü görevi gören bir Writer sınıfıdır. Parametre olarak yazılacak dosya, konsol gibi veri ortamlarını alır. Buna ek olarak, kullanmak istediğimiz karakter setini de ikinci parametre olarak girebiliriz. getEncoding() metodu ile kullanılan karakter setini döndürür.

```
OutputStreamWriteer outputstreamwriter= new OutputStreamWriter(OutputStream out, Charset cs);
```

```
OutputStream outputStream = new FileOutputStream("output.txt");
Writer outputStreamWriter = new OutputStreamWriter(outputStream);
outputStreamWriter.write("Hello World");
outputStreamWriter.close();
```

PrintWriter

- Parametre olarak çıkış hedefini alan Writer sınıfıdır. print ve println metotları bulunur.

```
PrintWriter writer = new PrintWriter(out);
```

- PrintWriter kullanarak istediğimiz tipte veriyi çıkış hedefine yazdırabiliriz.(Dosyaya ya da konsol ekranına)

StringWriter

- Parametre olarak string alan Writer sınıfıdır.

```
StringWriter writer = new StringWriter(String);
```

- Özet : BufferedWriter sınıfı, parametre olarak diğer yazıcıları alır. Daha verimli veri yazmayı sağlar. OutputStreamWriter karakter türünde tuttuğu veriyi, bayt olarak yazar ve yazarken girildiyse ikinci parametre olarak aldığı karakter setini kullanır. PrintWriter istenen tipte veriyi yazdırır. StringReader ise string parametreli bir yazıcıdır.

Stream ile Writer/Reader Farkı

- Stream sınıfı bayt tabanlı olduğu için ses, görüntü ve video gibi analog veriler için daha kullanışlıdır.

- Karakter tabanlı olan Writer ve Reader sınıfları ise metinsel ifadeler için daha kullanışlıdır.

Kaynakça

- app.patika.dev
- www.javatpoint.com
- www.stackoverflow.com
- www.programiz.com
- www.tasarimkodlama.com
- www.emrecelen.com.tr
- www.decodejava.com

Okuduğunuz İçin Teşekkür Ederim.

- Berkay Zaim 171421002