# 1 PDF Objects

There are eight basic types of objects in PDF. They're explained in sections 7.3.2 to 7.3.9 of ISO-32000-1.

## 1.1 The basic PDF objects

These eight objects are implemented in iText as subclasses of the abstract `PdfObject` class. Table 1.1 lists these types as well as their corresponding objects in iText.

**Table 1.1: Overview of the basic PDF objects**

| PDF Object | iText object | Description |
|---|---|---|
| Boolean | `PdfBoolean` | This type is similar to the Boolean type in programming languages and can be `true` or `false`. |
| Numeric object | `PdfNumber` | There are two types of numeric objects: integer and real. Numbers can be used to define coordinates, font sizes, and so on. |
| String | `PdfString` | String objects can be written in two ways:<br>as a sequence of literal characters enclosed in parentheses `( )` or as hexadecimal data enclosed in angle brackets `< >`.<br>Beginning with PDF 1.7, the type is further qualified as text string, PDFDocEncoded string, ASCII string, and byte string, depending upon how the string is used in each particular context. |
| Name | `PdfName` | A name object is an atomic symbol uniquely defined by a sequence of characters. Names can be used as keys for a dictionary, to define an explicit destination type, and so on. You can easily recognize names in a PDF file because they're all introduced with a forward slash: `/`. |
| Array | `PdfArray` | An array is a one-dimensional collection of objects, arranged sequentially between square brackets. For instance, a rectangle is defined as an array of four numbers: `[0 0 595 842]`. |
| Dictionary | `PdfDictionary` | A dictionary is an associative table containing pairs of objects known as dictionary entries. The key is always a name; the value can be (a reference to) any other object. The collection of pairs is enclosed by double angle brackets: `<<` and `>>`. |
| Stream | `PdfStream` | Like a string object, a stream is a sequence of bytes. The main difference is that a PDF consumer reads a string entirely, whereas a stream is best read incrementally. Strings are used for small pieces of data; streams are used for large amounts of data. |

**Table 1.1: Overview of the basic PDF objects**

| PDF Object | iText object | Description |
|---|---|---|
| | | Each stream consists of a dictionary followed by zero or more bytes enclosed between the keywords `stream` (followed by a newline) and `endstream`. |
| Null object | `PdfNull` | This type is similar to the `null` object in programming languages. Setting the value of a dictionary entry to to `null` is equivalent to omitting the entry. |

If you look inside iText, you'll find subclasses of these basic PDF implementations created for specific purposes.

- `PdfDate` extends `PdfString` because a date is a special type of string in the Portable Document Format.
- `PdfRectangle` is a special type of `PdfArray`, consisting of four number values: `[llx, lly, urx, ury]` representing the coordinates of the lower-left and upper-right corner of the rectangle.
- `PdfAction`, `PdfFormField`, `PdfOutline` are examples of subclasses of the `PdfDictionary` class.
- `PRStream` is a special implementation of `PdfStream` that needs to be used when extracting a stream from an existing PDF document using `PdfReader`.

When creating or manipulating PDF documents with iText, you'll use high-level objects and convenience methods most of the time. This means you probably won't be confronted with these basic objects very often, but it's interesting to take a look under the hood of iText.

## 1.2 iText's PdfObject implementations

Let's take a look at some simple code samples for each of the basic types.

### 1.2.1 PdfBoolean

As there are only two possible values for the `PdfBoolean` object, you can use a static instance instead of creating a new object.

**Code sample 1.1: C0101_BooleanObject**

```java
public static void main(String[] args) {
    showObject(PdfBoolean.PDFTRUE);
    showObject(PdfBoolean.PDFFALSE);
}
public static void showObject(PdfBoolean obj) {
    System.out.println(obj.getClass().getName() + ":");
    System.out.println("-> boolean? " + obj.isBoolean());
    System.out.println("-> type: " + obj.type());
    System.out.println("-> toString: " + obj.toString());
    System.out.println("-> booleanvalue: " + obj.booleanValue());
}
```