Binary I/O Streams v0.1
for Kotlin

Alexander Kornilov 2018

Table of content

Basic information

The **BinaryStreams** library is intended for I/O on **Kotlin** language.
The library has some advantages over Java I/O.
About all the features of the **BinaryStreams** please read in the next chapter.

Name	**BinaryStreams**
Type	Library (JAR)
Version	0.1
Language	Kotlin
ArtifactId	**binary-streams**
Package	loggersoft.kotlin.streams
Project URL	https://bitbucket.org/akornilov/binary-streams
API documentation	https://akornilov.bitbucket.io/doc/binary-streams
This document	https://bitbucket.org/akornilov/binary-streams/downloads/binary-streams.pdf
Gradle	compile 'org.bitbucket.akornilov.kotlin:binary-streams:0.1'
Maven central	https://repo.maven.apache.org/maven2/org/bitbucket/akornilov/kotlin/binary-streams/0.1/
Author	Alexander Kornilov (mailto:akornilov.82@gmail.com)
License	Apache v2.0 (http://www.apache.org/licenses/LICENSE-2.0)

Main features

- Extended support of data streams with configurable byte-order.
- The size of an integer in bytes is arbitrary for data streams.
- Integers with a size larger than 64-bit are supported using the *BigInteger*.
- Possibility to choose signed or unsigned representation of an integer.
- Unsigned integer (64-bits and more) using the *BigInteger*.
- Float and Double with specified byte order.
- Hint about fetch ability.
- Read or write an arbitrary number of bits from the *BitStream*.
- Can seek to a specified bit in the *BitStream*.
- Build-in support of string encoding: ASCII, UTF-8, UTF-16LE, UTF-16BE, UTF-32LE, UTF-32BE.
- Detection and writing BOM for text files.
- Reading chars as code points at all.
- Reads all lines from a text file in "*.useLines*" Kotlin style.
- String encoding and byte order might be set for all stream or specified in place.
- Input or output buffering on the fly with a specified buffer size.
- Stream implementation for random access file with reading and writing buffering.
- Stream implementation to read or write from *kotlin.ByteArray*.
- Adapters for *java.io.InputStream* and *java.io.OutputStream*.
- Helpful tools to make easy work with bits, code-points and other binary operations in *StreamUtils.kt*.

Code examples

```
// Creates stream over java.io.InputStream with default stream byte order
big-endian and default buffer size 4096.

FileInputStream("file.bin").toBinaryBufferedStream(byteOrder =
ByteOrder.BigEndian).use { stream ->

    // Reads byte as unsigned integer
    stream.readByteUnsigned()

    // Reads Double with specified in place byte-order
    stream.readDouble(ByteOrder.LittleEndian)

    // Skips ten bytes
    stream.skip(10)

    // Reads BigInteger with size 10 bytes and unsigned representation
    stream.readLong(10, false)

    // Can read three bytes?
    stream.canRead(3)

    // Reads UTF-16 string contains 20 code points
    stream.readString(StringEncoding.UTF16, 20)
}
```

```
// Creates stream over java.io.OutputStream with native byte order and
default string encoding UTF-8

FileOutputStream("file.bin").toBinaryStream().use { stream ->

    // Writes single byte
    stream.writeByte(33)

    // Writes long value using += operator
    val longValue = 0xFE30023L
    stream += longValue

    // Flushes data
    stream.flush()

    // Write code point 65 in UTF32 encoding
    stream.writeChar(65, StringEncoding.UTF32)

    // Writes BOM in current stream encoding
    stream.writeBom()

    // Writes line in current stream encoding
    stream.writeLine("Hello")

    // Writes double value
    stream.writeDouble(33.3)
}
```

```
// Creates stream for random access file for read/write and default read
buffer 4096:

File("file.bin").openBinaryStream(false).use { stream ->

    // Try detect BOM
    if (stream.tryDetectBom()) {
        // The BOM was detected and default stream encoding and byte order
were updated.
    }

    // Seeks to last 20 bytes of the file.
    stream.position = stream.size - 20L

    // Reads Int value from the stream
    val intValue: Int = stream.read()

    // Writes Int value
    stream.write(intValue)

    // Seeks to begin of the file
    stream.position = 0

    // Reads signed integer with size 3 bytes.
    stream.readInt(3, true)

    // Reads all lines to the end of file
    stream.forLines {
        for (line in it) {
            println(line)
        }
    }
}
```

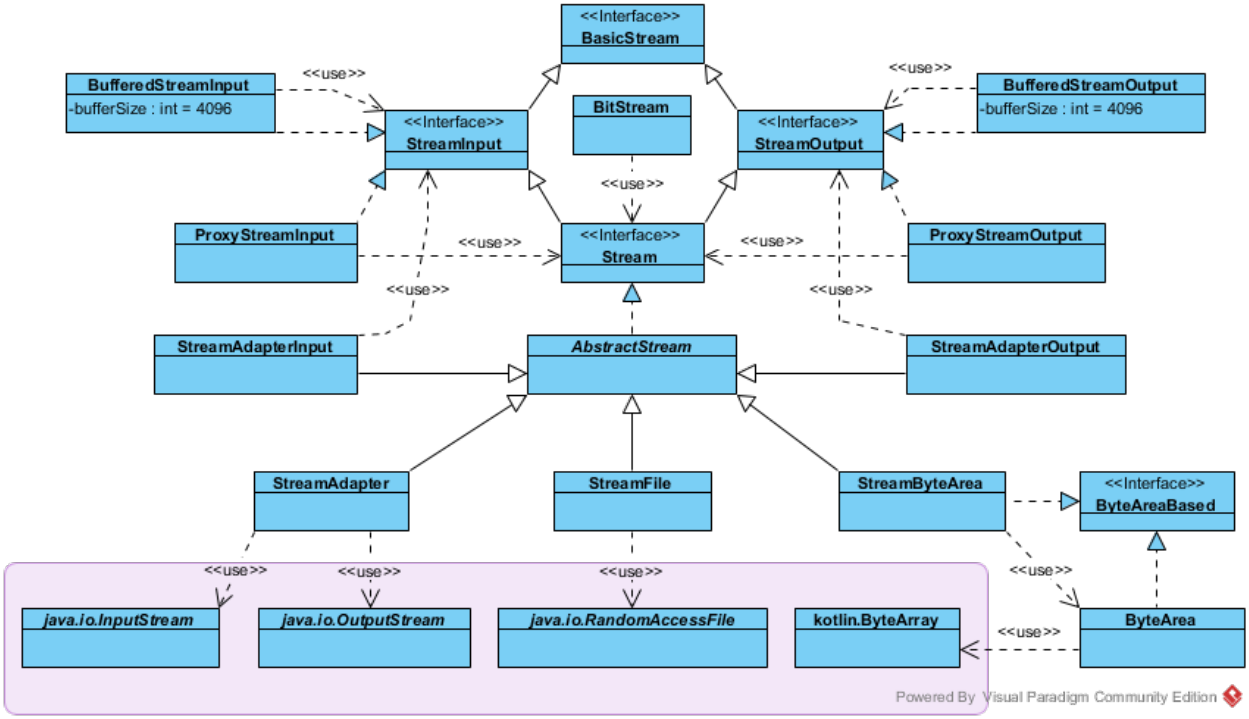
```
// Reads all lines from file in UTF-32

FileInputStream("file_urf32.txt").toBinaryBufferedStream(encoding =
StringEncoding.UTF32).useLines {
    for (line in it) {
        println(line)
    }
}
```

```
// Opens BitStream and read/write bits.

BitStream(File("file.bin").openBinaryStream(false)).use { stream ->

    // Seeks to 99 byte
    stream.position = 99

    // Seeks to 3rd bit in 99 byte
    stream.offset = 3

    // Seeks to 451 bit
    stream.bitPosition = 451L

    // Reads byte
    stream.readByte()

    // Read bit
    stream.readBit()

    // Read 33 bits as signed integer
```

```
    stream.readBits(33, true)

    // Write bit 1
    stream += true

    // Write bit 0
    stream += false

    // Write 4 bits
    stream.write(0b1101, 4)

    // Read 128 bits to BigInteger as unsigned
    stream.readBigInteger(128, false)
}
```

```
// Creates stream over ByteArray

val byteStream = StreamByteArea(ByteArea(16))

// Reads Short
byteStream.readShort()

// Seek in stream
byteStream.position = 10

// Etc, working with stream as well...
```

Class diagram



Classes description

Class name	Description
BasicStream	Root interface of inheritance: contains the most generic properties of any stream.
StreamInput	Represents streams for reading.
StreamOutput	Represents streams for writing.
Stream	Represents input and output stream.
AbstractStream	This class is recommended as the base for any *Stream* implementations.
StreamAdapter	Provides *Stream* interface from *java.io.InputStream* and *java.io.OutputStream*.
BufferedStreamInput	The decorator of *StreamInput* for buffering.
BufferedStreamOutput	The decorator of *StreamOutput* for buffering.
ProxyStreamInput	Provides *StreamInput* interface from *Stream*.
ProxyStreamOutput	Provides *StreamOutput* interface from *Stream*.
StreamAdapterInput	Provides *Stream* interface from *StreamInput*.
StreamAdapterOutput	Provides *Stream* interface from *StreamOutput*.
StreamFile	Implementation of the *Stream* for random access file (with optional buffering).
StreamByteArea	Implementation of the *Stream* for *ByteArea* (in fact for any *ByteAreaBased* objects).
ByteArea	Represents area inside byte array with specified offset and size (*).
BitStream	Bit access over *Stream*.