CSCI 2120: Software Design & Development II

UNIT3: I/O management

io api

DataOutputStream

Overview

Introduction

- **DataOutputStream in Java** is a filter output stream that provides methods for writing Java's standard data types.

- It enables you conveniently to write strings and all primitive data types such as boolean, int, float, long, etc to a stream.

- Java `DataOutputStream` first converts primitive-type values or strings into bytes and then writes them to the underlying output stream in an appropriate way.

- Using data input stream, we can then read the data back in. Thus, `DataOutputStream` works as wrappers on the existing output stream to filter data in the original stream.

DataOutputStream class declaration

`DataOutputStream` class extends `FilterOutputStream` class that extends `OutputStream`. It implements the interface `DataOutput` to use methods defined in the `DataOutput` interface. `DataOutputStream` class also implements `Closeable`, `Flushable`, and `AutoCloseable` interfaces.

The general declaration for `DataOutputStream` class in Java is given below:

```
public class DataOutputStream
       extends FilterOutputStream
       implements DataOutput
```

It was added in Java 1.0 version. It is present in the `java.io.DataOutputStream` package.

DataOutputStream Constructors

DataOutputStream class defines only a single constructor in Java that is as follows:

1. DataOutputStream(OutputStream outputStream)

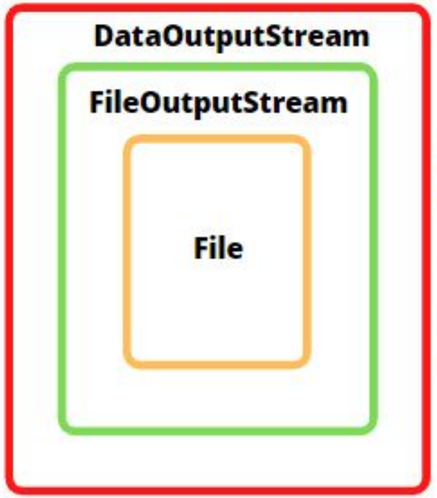
This constructor creates a `DataOutputStream` object that uses the specified underlying `OutputStream`. Here, `outputStream` defines the output stream to which data will be written.

A data output stream object for output can be created as follows:

```
FileOutputStream fos = new FileOutputStream(String filename);
DataOutputStream dos = new DataOutputStream(fos);
```

These two statements basically wrap `dos` on `fos` and use it as a filter.

DataOutputStream Constructors



```
File file = new File("myfile.txt");
FileOutputStrean fos = new FileOutputStream(file);
DataOutputStream dos = new DataOutputStream(fos);

or,

FileOutputStrean fos = new FileOutputStream("myfile.txt");
DataOutputStream dos = new DataOutputStream(fos);
```

A File wrapped with a FileOutputStream is wrapped with a DataOutputStream

Fig: A DataOutputStream object for writing to a file.

DataOutputStream Constructors

To append data to an existing file, use the following syntax:

```
FileOutputStream fos = new FileOutputStream(new File(String filename), true);
DataOutputStream dos = new DataOutputStream(fos);

//or,
FileOutputStream fos = new FileOutputStream(String filename, true);
DataOutputStream dos = new DataOutputStream(fos);
```

DataOutputStream Methods

In addition to methods inherited by `OutputStream` and `FilterOutputStream` superclasses, `DataOutputStream` class uses also methods defined by `DataOutput` interface that make it unique.

These methods convert values of a primitive data type into a byte sequence and then writes it to the underlying stream.

A list of important methods provided by `DataOutput` interface is as follows:

DataOutputStream Methods

Method	Description
void writeBoolean(boolean b)	This method writes a boolean to the output stream as a 1-byte value.
void writeByte(int v)	This method writes out a byte to the output stream as a 1-byte value.
void writeBytes(String s)	This method writes the lower byte of characters in a string to the underlying output stream as a sequence of bytes.
void writeChar(char c)	This method writes a character composed of 2 bytes to the underlying output stream, high byte first.
void writeChars(String s)	This method writes a sequence of characters in a string s to the underlying output stream, 2 bytes per character.
void writeDouble(double v)	This method writes a double value to the underlying output stream.

DataOutputStream Methods

Method	Description
void writeFloat(float v)	This method writes a float value to the underlying output stream.
void writeInt(int v)	It writes an int value to the underlying output stream.
void writeLong(long v)	This method writes a long value to the underlying output stream.
void writeShort(int v)	This method writes a short value to the underlying output stream.
void writeUTF(String str)	This method writes a string to the underlying output stream using UTF-8 format.

DataOutputStream Methods - Checked Exceptions

Almost all the methods in the I/O stream classes throw an exception named `IOException`. This exception is thrown when an Input/Output operation fails because of an interrupted call.

Therefore, we need to declare to throw `java.io.IOException` in the method or put the code in a `try-catch` block, as shown below:

```
//Declaring IOException exception in the method
public static void main(String[] args) throws IOException {
   // Perform I/O operations.
}
//or, Using try-catch block
public static void main(String[] args) {
   try {
       // Perform I/O operations
   }
   catch (IOException ex) {
       ex.printStackTrace();
   }
}
```

Example 1: Write & Read primitive data to/from File

1. Let’s take a simple example program where we will perform reading and writing operations using `DataInputStream` and `DataOutputStream`. Look at the following source code below.

Example 1: Write & Read primitive data to/from File

```java
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
public class DataOutputStreamTester1 {
   public static void main(String[] args) throws IOException {
       String filepath = "./src/mydata.dat";
       // Create a FileOutputStream object to connect with mydata.dat file.
       FileOutputStream fos = new FileOutputStream(filepath);
       // Create a DataOutputStream object to wrap on fos.
       DataOutputStream dos = new DataOutputStream(fos);
       // Write following primitive data to the "mydata.dat" file.
       dos.writeUTF("Welcome to Java world");
       dos.writeInt(1246);
       dos.writeDouble(125.25);
       dos.writeBoolean(true);
       dos.writeChar('S');
       dos.close();
       fos.close();
       // Reading data from the "myfileout.dat" file.
       FileInputStream fis = new FileInputStream(filepath);
       DataInputStream dis = new DataInputStream(fis);
       System.out.println(dis.readUTF());
       System.out.println(dis.readInt());
       System.out.println(dis.readDouble());
       System.out.println(dis.readBoolean());
       System.out.println(dis.readChar());
       dis.close();
       fis.close();
   }
}
```

Example 1: Write & Read primitive data to/from File

Output:

```
Welcome to Java world
1246
125.25
true
S
```

In this program, we have performed reading and writing primitive data types by wrapping `DataInputStream` on `FileInputStream`.

The program first creates "`myfiledata.dat`" file on the mentioned `filepath` and then writes the string and primitive data types into it using data output stream. At the end of writing, streams are closed using `close()` method.

Now the program also constructs a data input stream object and connects it to "`myfiledata.dat`" file. It then reads the following data from the file and displays them on the console. At last, it closes the streams.

Note:

The main method declares that it throws an exception named `IOException.` Therefore, we do not use *Java* *`try-catch block`*.

END