# CSCI 2120:
Software Design & Development II

*UNIT3: I/O management*

*io api*

**DataInputStream**

## Overview

## Introduction

- **DataInputStream in Java** is a filter input stream that provides methods for reading Java's standard data types.

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

- Java `DataInputStream` reads bytes from an underlying stream and converts them into suitable primitive-type values or strings.

- It reads them to its underlying byte stream and encodes these values in a machine-independent way.

- The basic input stream provides read methods only for reading bytes or characters. If we want to read the primitive data types, we need to use a filter class `DataInputStream.`

- `DataInputStream` class works as wrappers on the existing input stream to filter data in the original stream.

# DataInputStream class declaration

`DataInputStream` class extends `FilterInputStream` class that extends `InputStream`. It implements the interface `DataInput` to use methods defined in the `DataInput` interface. `DataInputStream` class also implements `Closeable` and `AutoCloseable` interfaces.

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

```
public class DataInputStream
       extends FilterInputStream
       implements DataInput
```

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

## DataInputStream Constructors

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

**1. DataInputStream(InputStream inputStream)**

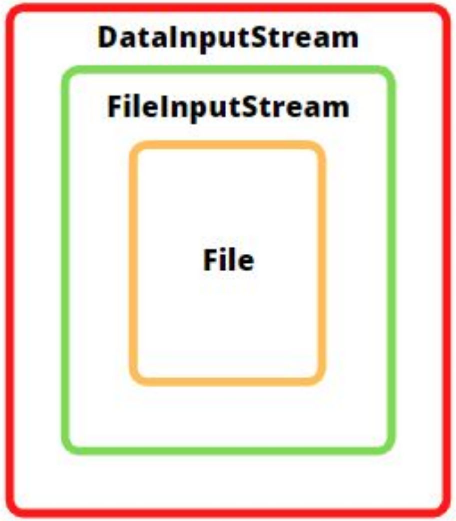
This constructor creates a `DataInputStream` object that uses the specified underlying `InputStream`. Here, `inputStream` defines the input stream from which data will be read.

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

```
FileInputStream fis = new FileInputStream(String filename);
DataInputStream dis = new DataInputStream(fis);
```

These two statements basically wrap `dis` on `fis` and use it as a filter.

## DataInputStream Constructors



```
File file = new File("myfile.txt");
FileInputStrean fis = new FileInputStream(file);
DataInputStream dis = new DataInputStream(fis);

or,

FileInputStrean fis = new FileInputStream("myfile.txt");
DataInputStream dis = new DataInputStream(fis);
```

**A File wrapped with a FileInputStream is wrapped with a DataInputStream**

**Fig: A DataInputStream object for reading binary file.**

## DataInputStream Methods

In addition to methods inherited by `InputStream` and `FilterInputStream` superclasses, `DataInputStream` class uses also methods defined by `DataInput` interface that make it unique.

These methods read a sequence of bytes and convert them into values of primitive data types. A list of important methods provided by `DataInput` interface is as follows:

## DataInputStream Methods

| Method | Description |
|---|---|
| boolean readBoolean() | This method reads byte from the stream and converts into boolean value. It returns true if byte is non zero, false if byte is zero. |
| byte readByte() | This method reads a single byte from contained input stream and returns a signed byte. |
| short readShort() | This method reads two bytes and returns a short value. |
| char readChar() | It reads two bytes from the contained input stream and returns a char value. |
| int readInt() | It reads 4 bytes from contained input stream and returns an int value. |
| long readLong() | This method read 8 bytes from the input stream and returns a long value. |
| float readFloat() | This method reads 4 bytes, converts data into float value, and returns it. |
| double readDouble() | The readDouble() method reads 8 bytes, converts data into double value, and returns it. |

## DataInputStream Methods

| Method | Description |
|---|---|
| String readUTF() | This method reads the length of string and then reads and returns a string that has been encoded using the UTF-8 format. |
| int skipBytes(int n) | This method skips over a specified number of bytes without reading them from an input stream. |
| void readFully(byte[ ] b) | This method reads bytes from the input stream and store them into the buffer array. |
| void readFully(byte[ ] b, int n, int m) | This method reads m bytes from array b starting from nth byte. |

# DataInputStream 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 an example program where we will write primitive data types in a file, then read data from the file, and display them on the screen. 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 DataInputStreamTester1 {
   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