**RandomAccessFile**

**Advantages:**The advantages of using RandomAccessFile in java is that you can read and write portions of data at any position in the file by using a file pointer. By using a file pointer, you can go backwards and forwards within the same file unlike I/O streams which would require you to go in order in sequence. The records can be of different sizes as well.

**Disadvantages:**

The disadvantages of using RandomAccessFile in Java is that it is very resource heavy compared to Sequential access. RandomAccessFile doesn't inherit from the InputStream or OutputStream, this means that you can't apply the same filters to RandomAccessFiles that you can to streams.

**Usage:**

A real-life example of RandomAccessFile would be how data is accessed in a CD opposed to a magnetic tape data storage. In a CD, although you can access the data sequentially, it would be inefficient compared to random access. It would allow rapid access to segments of data instead of loading the entire CD. Magnetic tape data storage would require sequential access would be time consuming and inefficient.

**Syntax:**

Following is the declaration for Java.io.RandomAccessFile class.

*public class RandomAccessFile*

*extends Object*

*implements DataOutput, DataInput, Closeable*

**Constructors:**  
There are two constructors for RandomAccessFile,  
1. RandomAccessFile(File file, String mode)  
2. RandomAccessFile(String name, String mode)

**Methods:**

Common methods that are used with Random Access File.

|  |  |  |
| --- | --- | --- |
| **Modifier and Type** | **Method** | **Method** |
| void | close() | It closes this random access file stream and releases any system resources associated with the stream. |
| FileChannel | getChannel() | It returns the unique [FileChannel](https://www.javatpoint.com/data-transfer-between-channels) object associated with this file. |
| int | readInt() | It reads a signed 32-bit integer from this file. |
| String | readUTF() | It reads in a string from this file. |
| void | seek(long pos) | It sets the file-pointer offset, measured from the beginning of this file, at which the next read or write occurs. |
| void | writeDouble(double v) | It converts the double argument to a long using the doubleToLongBits method in class Double, and then writes that long value to the file as an eight-byte quantity, high byte first. |
| void | writeFloat(float v) | It converts the float argument to an int using the floatToIntBits method in class Float, and then writes that int value to the file as a four-byte quantity, high byte first. |
| void | write(int b) | It writes the specified byte to this file. |
| int | read() | It reads a byte of data from this file. |
| long | length() | It returns the length of this file. |
| void | seek(long pos) | It sets the file-pointer offset, measured from the beginning of this file, at which the next read or write occurs. |

**References:**

<https://docs.oracle.com/javase/7/docs/api/java/io/RandomAccessFile.html>

<https://peda.net/kenya/css/subjects/computer-studies/form-three/driac2/data-processing/fom>

<https://www.math.uni-hamburg.de/doc/java/tutorial/essential/io/rafs.html>

<https://www.javatpoint.com/java-randomaccessfile-class>

<https://www.tutorialspoint.com/java/io/java_io_randomaccessfile.htm>