RandomAccessFile

This class allows the random movement of the file pointers. The file pointer can be moved back and forth.

Constructor:

RandomAccessFile(String name, String mode)

Creates a random access file stream to read from, and optionally to write to, a file with the specified name.

Mode can be either of "r" or "rw".

"r" - Open for reading only. Invoking any of the write methods of the resulting object will cause an IOException to be thrown.

"rw" - Open for reading and writing. If the file does not already exist then an attempt will be made to create it.

If the file is not found it throws FileNotFoundException.

Methods

long length() - Returns the length of this file.

int read() - Reads a byte of data from this file. The byte is returned as an integer in the range 0 to 255 or -1 if the end of the file has been reached. This method blocks if no input is yet available.

int read(byte[] b) - Reads up to b.length bytes of data from this file into an array of bytes.

boolean readBoolean() - Reads a boolean from this file.

byte readByte() - Reads a signed eight-bit value from this file.

char readChar() - Reads a Unicode character from this file.

double readDouble() - Reads a double from this file.

float readFloat() - Reads a float from this file.

int readInt() - Reads a signed 32-bit integer from this file.

String readLine() - Reads the next line of text from this file.

long readLong() - Reads a signed 64-bit integer from this file.

short readShort() - Reads a signed 16-bit number from this file.

int readUnsignedByte() - Reads an unsigned eight-bit number from this file.

int readUnsignedShort() - Reads an unsigned 16-bit number from this file.

void seek(long pos) - Sets the file-pointer offset, measured from the beginning of this file, at which the next read or write occurs.

void setLength(long newLength) - Sets the length of this file. If the present length of the file as returned by the length method is greater than the newLength argument then the file will be truncated

int skipBytes(int n) - Attempts to skip over n bytes of input discarding the skipped bytes.

void write(int b) - Writes the specified byte to this file. The write starts at the current file pointer. b - the byte to be written.

void writeBoolean(boolean v) - Writes a boolean to the file as a one-byte value.

void writeByte(int v) - Writes a byte to the file as a one-byte value.

void writeBytes(String s) - Writes the string to the file as a sequence of bytes.
void writeChar(int v) - Writes a char to the file as a two-byte value, high byte first.
void writeChars(String s) - Writes a string to the file as a sequence of characters.
void writeDouble(double v) - 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) - 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 writeInt(int v) - Writes an int to the file as four bytes, high byte first.
void writeLong(long v) - Writes a long to the file as eight bytes, high byte first.
void writeShort(int v) - Writes a short to the file as two bytes, high byte first

Visibility

Access or visibility rules determine whether a method or a data variable can be accessed by another method in another class or subclass.

Java provides for 4 access modifiers :

1. **public** - It makes classes, methods, or data available to any other method in any other class or subclass.
2. **protected** - accessible by the package classes and any subclasses that are in other packages .
3. **"-"** - accessible to classes in the same package but not by classes in other packages, even if these are subclasses.
4. **private** - accessible only within the class. Even methods in subclasses in the same package do not have access.

Note that a java file can have only one public class.

Binary I/O

The java.io package contains two classes, InputStream and OutputStream, from which most of the other classes in the package derive.

The InputStream class is an abstract superclass that provides a minimal programming interface and a partial implementation of input streams. The InputStream class defines methods for reading bytes or arrays of bytes, marking locations in the stream, skipping bytes of input, finding out the number of bytes available for reading, and resetting the current position within

the stream. An input stream is automatically opened when you create it. You can explicitly close a stream with the close method.

The OutputStream class is an abstract superclass that provides a minimal programming interface and a partial implementation of
output streams. OutputStream defines methods for writing bytes or arrays of bytes to the stream. An output stream is
automatically opened when you create it. You can explicitly close an output stream with the close method.

OutputStream

This abstract class is the superclass of all classes representing an output stream of bytes.
An output stream accepts output bytes and sends them to some sink.

Methods

void close() - Closes this output stream and releases any system resources associated with this stream.
void flush() - Flushes this output stream and forces any buffered output bytes to be written out.
abstract void write(int b) - Writes the specified byte to this output stream.

Throws IOException if an I/O error occurs

InputStream

public abstract int read() throws IOException

Reads the next byte of data from the input stream.
The value byte is returned as an int in the range 0 to 255.
If no byte is available because the end of the stream has been reached, the value -1 is returned.
This method blocks until input data is available, the end of the stream is detected, or an exception is thrown.

Returns: the next byte of data, or -1 if the end of the stream is reached.
Throws: IOException - if an I/O error occurs.

public long skip(long n) throws IOException

Skips over and discards n bytes of data from this input stream. The skip method may,
for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0.
This may result from any of a number of conditions; reaching end of file before

n bytes have been skipped is only one possibility. The actual number of bytes skipped is returned.
If n is negative, no bytes are skipped.

The skip method of InputStream creates a byte array and then repeatedly reads into it until n bytes
have been read or the end of the stream has been reached.

Returns: the actual number of bytes skipped.
Throws: IOException - if an I/O error occurs.

public void close() throws IOException

Closes this input stream and releases any system resources associated with the stream.

Throws: IOException - if an I/O error occurs.

Reader Class

Constructor: protected Reader() - Create a new character-stream reader.

public int read() throws IOException

Read a single character. This method will block until a character is available, an I/O error occurs,
or the end of the stream is reached.

Returns: The character read, as an integer in the range 0 to 65535 or -1 if the end of the stream has been reached
Throws: IOException - If an I/O error occurs

public long skip(long n) throws IOException

Skip n characters. This method will block until some characters are available, an I/O error occurs,
or the end of the stream is reached.

Returns: The number of characters actually skipped
Throws: IllegalArgumentException - If n is negative.
IOException - If an I/O error occurs

public abstract void close() throws IOException

Close the stream. Once a stream has been closed, further read() invocations will throw an IOException.
Closing a previously-closed stream, however, has no effect.

Throws: IOException - If an I/O error occurs

Packages

In Java source files, the package that the file belongs to is specified with the package keyword.
ex:- **package** java.awt.event;

Classes within a package can access classes and members declared with default access and class members
declared with the protected access modifier. Default access is enforced when neither the public, protected nor
private access modifier is specified in the declaration. By contrast, classes in other packages cannot access
classes and members declared with default access. Class members declared as protected can be accessed from the
classes in same as well as classes in other packages that are subclasses of the declaring class.

The CLASSPATH is an environment variable, that tells the Java Virtual Machine where to look for user-defined classes
and packages in Java programs.