9. Streams in Java

Lesson 1:

Understanding Streams and new I/O capabilities

9.1. Streams and new I/O capabilities

- A **stream** is an abstract representation of an input or output device that is a source of, or destination for data.
- You can write data to a stream and read data from a stream.
- Java programs perform I/O through streams.
- Input Streams sends data from a source into a program
- Output Streams sends data out of a program to a destination

9.2. Understanding streams

9.2.1. Buffered Streams

- Stream input and output methods generally permit very small amounts of data, such as a single character or byte, to be written or read in a single operation.

- Transferring data to or from a stream like this may be extremely inefficient, so a stream is often equipped with a **buffer** in memory, in which case it is called
- a **buffered stream**.

- A buffer is simply a block of memory that is used to batch up the data that is transferred to or from an external device. Reading or writing a stream in reasonably large chunks will reduce the number of input/output operations necessary and thus make the process more efficient.

9.2. Understanding streams

9.2.2. Flushing

- When you write to a buffered output stream, the data is sent to the buffer and not to the external device.
- The amount of data in the buffer is tracked automatically, and the data is usually sent to the device when
- the buffer is full.
- However, you will sometimes want the data in the buffer to be sent to the device before the buffer is full, and methods are provided to do this.
- This operation is usually termed **flushing the buffer**.

9.2. Understanding streams

9.2.3. Binary and Character Streams

- The Java 2 supports two types of streams

- Binary streams
  - These contain binary data.
  - Binary streams are sometimes referred to as Byte streams.
  - When you write data to a binary stream, the data is written to the stream as a series of bytes, exactly as it appears in memory. No transformation of the data takes place.

- Character streams
  - These contain character data.
  - Character streams are used for storing and retrieving text.
  - All binary numeric data has to be converted to a textual representation
  - before being written to a character stream. This involves generating a character representation of the original binary data value.

9.2. Understanding streams

9.2.4. Using Input Streams

- For an input stream, the first step is to **create an object** that is associated with the data source.

- After you have created a stream object, you can **read information** from that stream by using one of the object's methods.
  - For example, **FileInputStream** includes a **read()** method that returns a byte read from the file.

- When you're done reading information from the stream, you call the **close()** method to indicate that you're done using the stream.

9.2. Understanding streams

9.2.5. Using Output Streams

- For an output stream, as in the case for a input stream, the first step is to **create an object** that is associated with the data source.

- After you have created a stream object, you can **write information** from that stream by using one of the object's methods.

- When you're done writing information from the stream, you call the **close()** method to indicate that you're done using the stream.

9.2. Understanding streams

9.2.6. Filtering a Stream

- A **Filter** is a **type of stream** that modifies the way an existing stream is handled.

- The procedure for using a filter on a stream is basically as follows.
  - **Create a stream** associated with a data source or a data destination.
  - **Associate a filter** with that stream.
  - **Read or write data** from the filter rather than the original stream.

- A filter can associate with another filter.