9. Streams in Java

Lesson 2:

Standard Streams and classes for input and output – Part 2

9.3. Classes for Input and Output

9.3.9. InputStreamReader and BufferedReader

- You could create an InputStreamReader object like this, for example:
  - InputStreamReader keyboard = new InputStreamReader(System.in);
- The parameter to the InputStreamReader constructor is of type InputStream, so you can pass an object of any class derived from InputStream to it.
  - The sample above creates an InputStreamReader object, keyboard, from the object System.in, the keyboard input stream.
- The **read(...)**methods in the InputStreamReader class read bytes from the underlying stream and return them as Unicode characters.
- The operations with a reader would be much more efficient if you buffered it using a
- BufferedReader object like this:
  - BufferedReader keyboard = new BufferedReader(new InputStreamReader(System.in));
- Here, you wrap an InputStreamReader object around System.in and then buffer it using a BufferedReader object. This will make the input operations much more efficient. Your read operations will be from the buffer belonging to the BufferedReader object, and this object will take care of filling the buffer from System.in when necessary via the underlying InputStreamReader object.

9.3. Classes for Input and Output

9.3.10. Subclasses of the Writer class

- OutputStreamWriter
  - This is used for writing a character stream
- PipedWriter
  - This is used for writing to a PipedReader
- BufferedWriter
  - This is used for buffering other writers
- CharArrayWriter
  - This is used for writing to a char array
- PrintWriter
  - This is used for writing formated data
- StringWriter
  - This is used for writing to a string
- FilterWriter
  - This is used for writing filtered streams

9.3. Classes for Input and Output

9.3.11. OutputStreamWriter and PrintWriter

- The **OutputStreamWriter** class writes characters to an underlying binary stream.
  - It also has a subclass, **FileWriter**, that writes characters to a stream encapsulating a file.

- The **PrintWriter** class defines methods for formatting binary data as characters and writing it to a character stream. It defines overloaded print() and println() methods that accept an argument of each of the basic data types, of type char[], of type String, and of type Object.
  - The data that is written is a character representation of the argument. Numerical values and objects are converted to a string representation using the static valueOf() method in the String class. Overloaded versions of this method exist for all of the primitive types plus type Object.
  - You can create a PrintWriter object from a stream or from another Writer object.

9.3. Classes for Input and Output

9.3.12. FileInputStream

- A file input stream can be created with the **FileInputStream(String)** constructor.
  - The string argument should be the name of the file.

- The following statement creates a file input stream from the file “**scores.dat**”.
  - **FileInputStream fis = new FileInputStream(“scores.dat”);**

- After you create a file input stream, you can read bytes from the stream by calling its **read()** method.
  - To read more than one byte of data from the stream, call its **read(byte[], int, int)** method.

9.3. Classes for Input and Output

9.3.13. FileOutputStream

- A file output stream can be created with the **FileOutputStream(String)** constructor.
- You can create a file output stream that appends data after the end of an existing file with the **FileOutputStream(String, boolean)** constructor.
- The file output stream's **write(int)** method is used to write bytes to the stream.
  - To write more than one byte, the **write(byte[],int,int)** method can be used.

9.3. Classes for Input and Output

9.3.14. DataInputStream & DataOutputStream

- When you need to work with data that isn't represented as bytes or characters, you can use data input & data output streams.

- These streams **filter** an existing byte stream so that each of the following primitive types can be read or written directly from the stream:
  - boolean, byte, double, float, int, long & short

- A data input stream is created with the **DataInputStream(InputStream)** constructor. Similarly, a data output stream is created with the **DataOutputStream(OutputStream)** constructor.

9.4. The Standard Streams

9.4.1. Predefined Streams in Systems

- Your operating system will typically define three standard streams that are accessible through members of the System class in Java:
    - A **standard input stream, System.in** that usually corresponds to the keyboard by default. This is encapsulated by the in member of the System class and is of type InputStream.
    - A **standard output, System.out** stream that corresponds to output on the command line. This is encapsulated by the out member of the System class and is of type PrintStream.
    - A **standard error, System.err** output stream for error messages that usually maps to the command-line output by default. This is encapsulated by the err member of the System class and is also of type PrintStream.
- You can reassign any of these to another stream within a Java application. The System class provides the static methods setIn(), setOut(), and setErr() for this purpose.