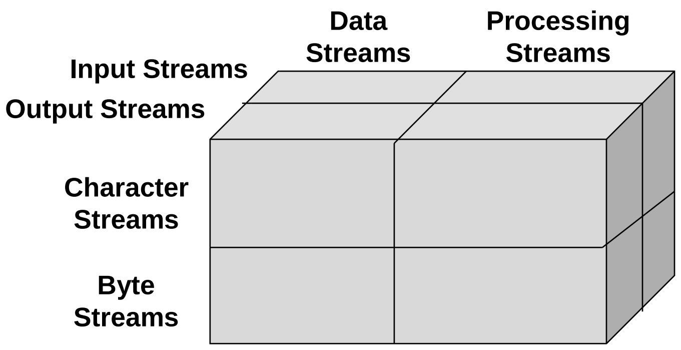
I/O Streams

- A *stream* is a sequence of bytes that flow from a source to a destination
- In a program, we read information from an input stream and write information to an output stream
- A program can manage multiple streams simultaneously

I/O Streams

- The `java.io` package contains many classes that allow us to define various streams with particular characteristics
- Some classes assume that the data consists of characters
- Others assume that the data consists of raw bytes of binary information
- Streams can be further subdivided as follows:
  - *data stream*, which acts as either a source or destination
  - *processing stream*, which alters or manipulates the basic data in the stream

I/O Streams



Character vs. Byte Streams

- A *character stream* manages 16-bit Unicode characters
- A *byte stream* manages 8-bit bytes of raw binary data
  - A program must determine how to interpret and use the bytes in a byte stream
  - Typically they are used to read and write sounds and images
- The `InputStream` and `OutputStream` classes (and their descendants) represent byte streams
- The `Reader` and `Writer` classes (and their descendants) represent character streams

Data vs. Processing Streams

- A *data stream* represents a particular source or destination such as a string in memory or a file on disk
- A *processing stream* (also called a *filtering stream*) manipulates the data in the stream
  - It may convert the data from one format to another
  - It may buffer the stream

The IOException Class

- Operations performed by the I/O classes may throw an `IOException`
  - A file intended for reading or writing might not exist
  - Even if the file exists, a program may not be able to find it
  - The file might not contain the kind of data we expect
- An IOException is a checked exception

Standard I/O

- There are three standard I/O streams:
  - *standard input* – defined by `System.in`
  - *standard output* – defined by `System.out`
  - *standard error* – defined by `System.err`
- `System.in` typically represents keyboard input
- `System.out` and `System.err` typically represent a particular window on the monitor screen
- We use `System.out` when we execute `println` statements

Standard I/O

- `PrintStream` objects automatically have `print` and `println` methods defined for them
- The `PrintWriter` class is needed for advanced internationalization and error checking

Text Files

- Information can be read from and written to text files by declaring and using the correct I/O streams
- The `FileReader` class represents an input file containing character data
- The `FileReader` and `BufferedReader` classes together create a convenient text file output stream

Text Files

- The `FileWriter` class represents a text output file, but with minimal support for manipulating data
- Therefore, the `PrintWriter` class provides `print` and `println` methods
- Output streams should be closed explicitly

Object Serialization

- *Object serialization* is the mechanism for saving an object, and its current state, so that it can be used again in another program
- The idea that an object can "live" beyond the program execution that created it is called *persistence*
- Object serialization is accomplished using the `Serializable` interface and the `ObjectOutputStream` and `ObjectInputStream classes`
- The `writeObject` method is used to serialize an object
- The `readObject` method is used to deserialize an object

Object Serialization

- ObjectOutputStream and ObjectInputStream are processing streams that must be wrapped around an OutputStream or an InputStream
- Once serialized, the objects can be read again into another program

Object Serialization

- Serialization takes into account any other objects that are referenced by an object being serialized, saving them too
- Each such object must also implement the Serializable interface
- Many classes from the Java class library implement Serializable, including the String class
- The `ArrayList` class also implements the `Serializable` interface, permitting an entire list of objects to be serialized in one operation