Java I/O Summary

- File class
  - Is a name for information on the hard disk
  - Can use this class to create, delete, list files, etc.
- Scanner class
  - Simple class for reading text from a file
- Byte Streams & Character Streams
- Filter Streams

By the end of this lecture, you will be able to use the File class and Scanner class to read data from text files in your programs.

You will also be able to use byte & character streams to read and write data using different encodings.

You will also be able to read and write objects to/from a file.

When your program ends, what happens to all of the information you had in memory?

When your computer shuts down, what happens to all of your work (e.g., documents, music, photos, etc.)?

Where does the information get stored:

a) when you run your program

b) when you turn off your computer?

File

- Most operating systems use the metaphor of a file to represent stored information.
- Files are usually stored in a hierarchy within the operating system.
- A file is just a name that any program can use to access a particular part of the hard disk.

File class

- In Java, there is a class that encapsulates the information about a particular file on the hard disk and lets you do operations on that file.

- Exercise: look at the File class in the Java API

Exercise: create a program that lists all of the files in the directory named “Documents”.

String[] list()
Returns an array of strings naming the files and directories in the directory denoted by this abstract pathname.

When your favourite music player (e.g., iTunes) plays an mp3 file, how does it get the information that it contains?

Where does it put the information?

Scanner

- The Scanner class is one of the simplest ways in Java to read textual data from a file.

Scanner

- Two constructors you can already use:
  - one takes a File
  - one takes a String
- Methods:
  - hasNext()
  - next(), nextInt(), nextFloat(), etc.

Did anyone look at the getFile method provided in Assignment #3?

In A3Helper.java…

```
public static List<String> getFile(String filename)
{
  try
  {
    Scanner scanner = new Scanner(new File(filename));

    ArrayList<String> list = new ArrayList<String>();
    while (scanner.hasNext())
    {
      String nextWord = scanner.next();
      nextWord = nextWord.replaceAll("[^A-Za-z']", "");
      if (nextWord.length() > 0)
      {
        list.add(nextWord);
      }
    }

    return list;
  }
  catch (Exception x)
  {
    return null;
  }
}
```

Exercise: create a program that reads a list of floating-point numbers from a file called “numbers.txt” into a LinkedList.

In what form is the information in "numbers.txt" stored on the computer?

Streams

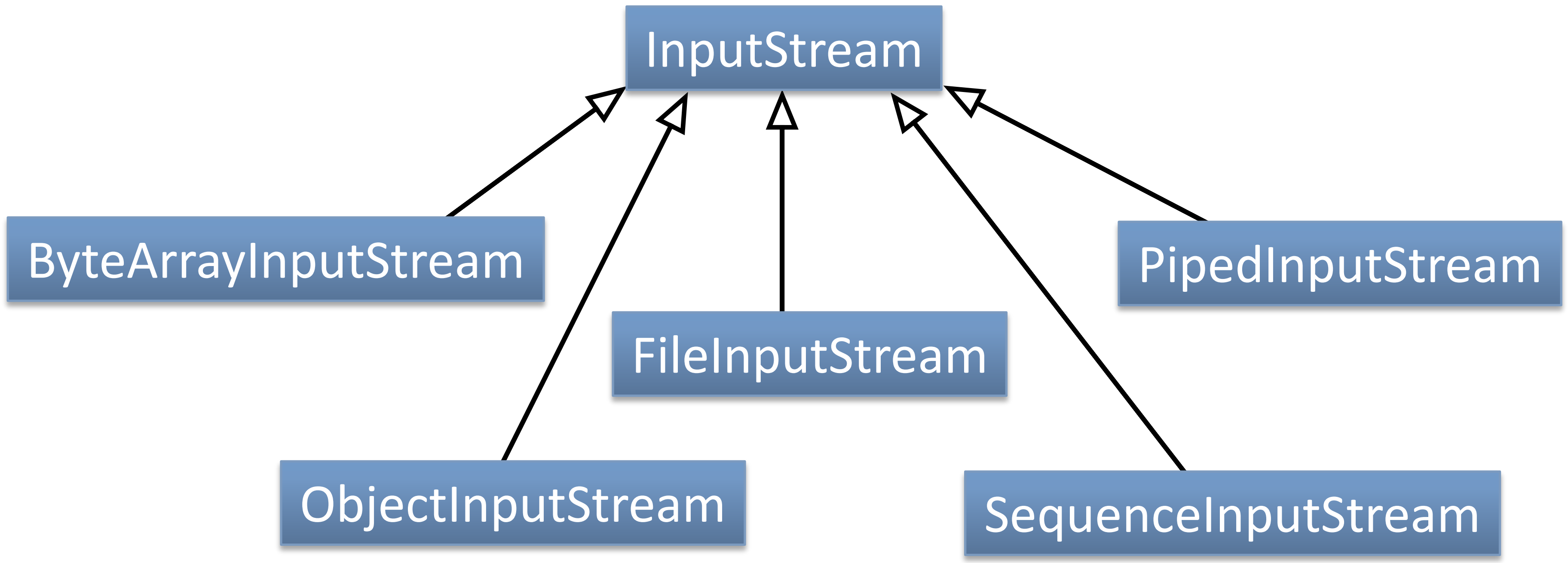
- A series of bytes that we can read or write
- Reading
  - can read each byte from left to write
  - can read until we reach the end of the stream
- Writing
  - bytes stored in the order they are written
  - can write until the operating system stops us
- Can do both at the same time (but it requires special functionality)

Byte Streams

- The unit being read or written is a byte

- Two important parent classes:
  - InputStream
  - OutputStream

InputStream



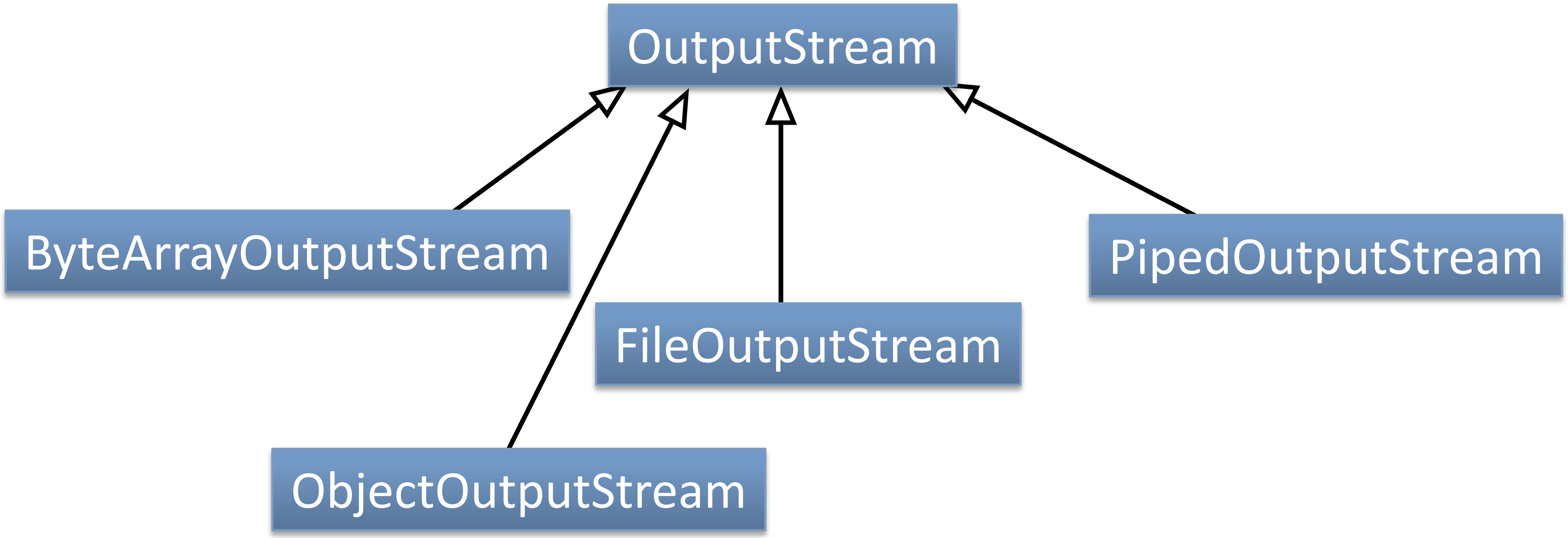
InputStream methods

Method	Description
int read()	Reads a single character, returns as an integer
int read(byte[] buffer)	Reads bytes from stream and places them into the buffer. The maximum number of bytes read will be equal to the size of the buffer. This method returns the number of bytes read
int read(byte[] buffer, int offset, int length)	Reads up to length bytes and places them into the buffer at location buffer[offset]. This method returns the number of bytes read
int available()	The number of bytes which can be read without blocking
long skip (long n)	This method skips over n bytes in the stream
close()	Closes the stream and releases any system resouces associated with the stream
boolean markSupported()	Returns true if this stream supports the mark and reset methods
mark (int readlimit)	Marks the current location within the stream. The readlimit parameter indicates how many bytes can be read before the mark becomes invalidated
reset()	Repositions the stream to the location set with the last call to mark.

InputStream subclasses

Class	Description
ByteArrayInputStream	The constructor for this class is provided with a byte array. This byte array contains the bytes which will be provided by the stream. This class is useful if the programmer wishes to access a byte array using the stream interface (i.e. reading sequential bytes)
ObjectInputStream	This class takes another InputStream as a constructor parameter. It reads bytes from the input stream and interprets them as *Serialized Objects* (which is covered in more detail later).
SequenceInputStream	Constructor takes multiple InputStreams and allows logical concatenation of the streams. When one stream ends, reading continues from the next, and so on. The program is unaware that the stream from which data is being read changes.
FileInputStream	This is the most commonly used InputStream. The constructor takes a filename, File object or FileDescriptor as a parameter. Data read from this stream comes from the file identified in the constructor.
PipedInputStream	Connects to an Instance of PipedOutputStream. This provides a one-way stream through which two threads may communicate. **Note:** Threading hasn't been covered in this course yet.

OutputStream



OutputStream methods

Method	Description
void write(int data)	Writes the data as a byte
void write(byte[] buffer)	Writes all of the bytes contained within the buffer to the stream
void write(byte[] buffer, int offset, int length)	writes *length* bytes to the stream starting at point buffer[offset]
void flush()	Flushes the OutputStream and forces any buffered output to be written to the stream
void close()	Closes the stream and releases any resources associated with the stream

OutputStream subclasses

Class	Description
ByteArrayOutputStream	All bytes written to this stream will be stored in a byte array. This array can be recovered by using the toByteArray() method
FileOutputStream	Most commonly used OutputStream. The constructor takes a filename, File object or FileDescriptor object as a parameter. All bytes written to this stream will be written to the underlying file. Has constructors which indicate that new data written to the file should be appended to the end of the file.
ObjectOutputStream	The constructor for this stream takes another stream as a parameter. Programmers can serialize objects by writing them to this stream using the writeObject() method.
PipedOutputStream	Connects to an instance of PipedInputStream to provide a one-way communication stream through which 2 threads may communicate

Limitations

- Good for ASCII encodings of characters
- Reading/writing unicode characters requires extra effort
  - e.g., internationalized character sets

Character Streams

- Unit being read or written is a (unicode) character

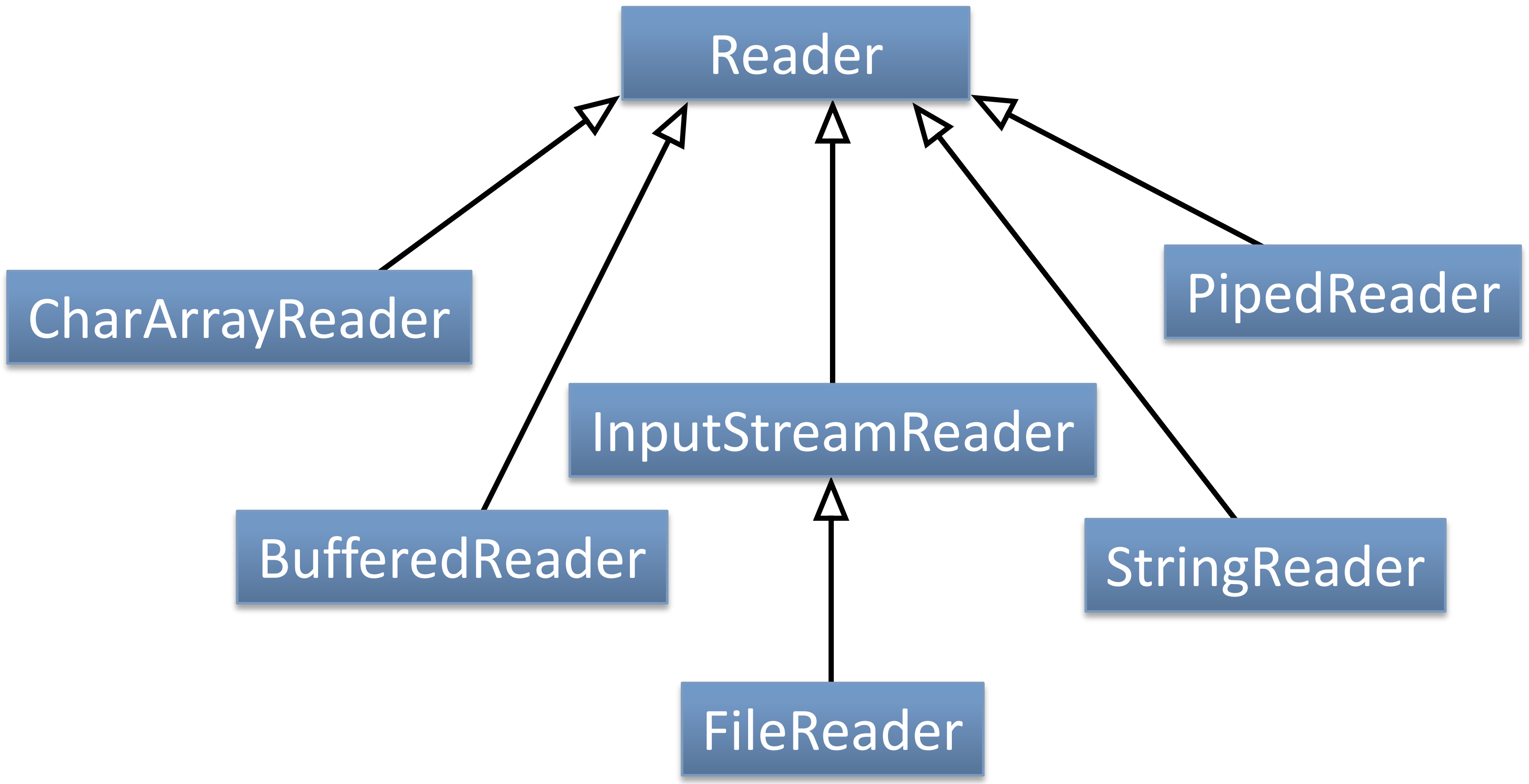
- Two important parent classes
  - Reader
  - Writer

Does it make sense to read in some bytes as if they were characters?

Conversion Classes

- InputStreamReader
  - converts an InputStream into a Reader
- OutputStreamWriter
  - converts an OutputStream into a Writer

Reader


Reader
CharArrayReader
PipedReader
InputStreamReader
BufferedReader
StringReader
FileReader

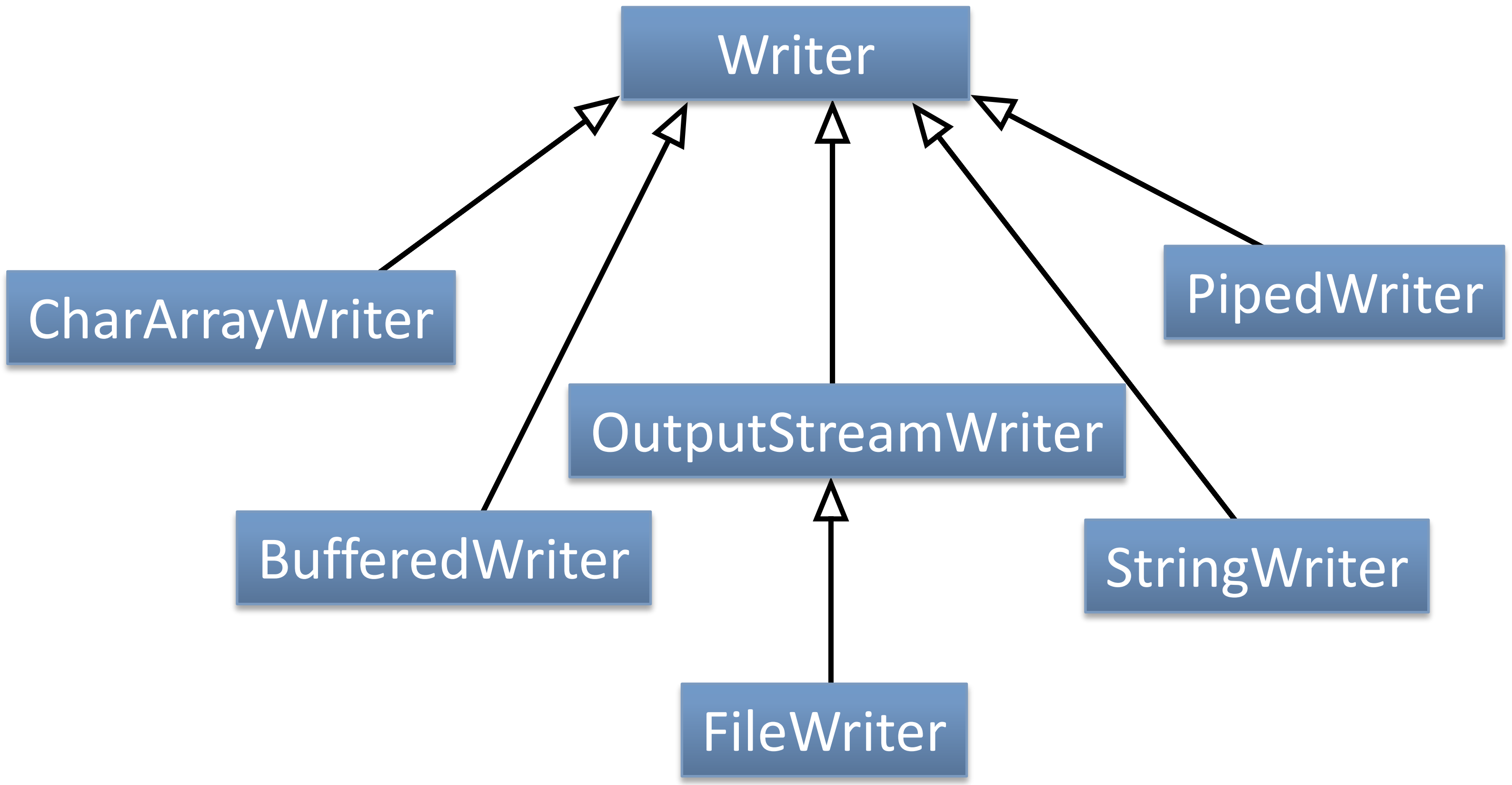
Reader methods

Method	Description
int read()	reads a character and returns it as an integer (only 16 bits are valid)
read (char[] buffer)	reads characters into the array up to the length of the array
read (char[] buffer, int offset, int length)	reads *length* characters into a character array starting at poing buffer[offset]
close()	closes the stream and releases any resources associated with the Reader
boolean ready()	returns true if the next call to read will not result in a block
boolean markSupported()	returns true if this Reader supports the mark() and reset() operations
mark(int readAheadLimit)	mark the present location in the stream.
reset()	resets the stream to the location previously set with mark()
skip (long n)	skips n characters in the stream

Reader subclasses

Class	Description
PipedReader	Used to create a one-way pipe between threads. A PipedReader object represents the receiving side of the pipe
BufferedReader	Provides a mechanism for reading characters from an input source while buffering the characters so that more efficient reading can occurr
CharArrayReader	Similar to ByteArrayInputStream. Used so that a character array can provide the data for a Reader. This is useful if the programmer wishes to read from a Character Array using the stream interfaces.
StringReader	Similar to the CharArrayReader where the data source is a String object.
InputStreamReader	An InputStreamReader is a bridge from byte streams to character streams: It reads bytes and decodes them into characters using a specified character set.
FileReader	A convenience class for reading textfiles.

Writer


Writer
CharArrayWriter
PipedWriter
OutputStreamWriter
BufferedWriter
StringWriter
FileWriter

Writer methods

Method	Description
write(int c)	writes the character to the stream. Note that even though the parameter is an integer, only 16 bits are written to the stream
write(String s)	writes the String to the stream
write(char[] buffer)	writes the buffer to the stream
write (String s, int offset, int length)	write *length* characters from the specified String starting at the specified offset
flush()	forces any characters currently being buffered to be written to the stream
close()	closes the stream and releases any resources associated with the stream

Writer subclasses

Class	Description
PipedWriter	Used in conjunction with PipedReader to create a one-way communication between two threads.
BufferedWriter	Writes text to a character-oriented stream while buffering characters to provide for efficiency
CharArrayWriter	Writes characters to a character array which can be recovered using the toCharArray() or toString() methods
FileWriter	Convenience class for writing character files (text)
StringWriter	Writes characters to a StringBuffer which can be recovered using the toString() method
OutputStreamWriter	Bridge class between character-oriented streams and byte-oriented streams

Exercise: create a method that writes out the contents of an ArrayList<String> to a file using the FileWriter class.

Filter Streams

- Similar to pipes on the command line
  - the output of one stream is the input of another
  - each filter modifies the data in some manner

Filter Streams

Class	Description
DataInputStream	read primitive data types from an underlying input stream
DataOutputStream	writes primitive data types to an underlying output stream
PushbackInputStream	Allows the ability to push read data back onto the stream
GZIPInputStream	reads compressed data in the GZIP format
GZIPOutputStream	writes compressed data to the GZIP format
ZipInputStream	reads compressed data in the Zip format
ZipOutputStream	writes compressed data to the Zip format
PushbackReader	Allows the ability to push read data back onto the reader

Example

```java
import java.io.*;
import java.util.zip.*;

public class CompressProgram
{
  public static void main(String[] args) throws IOException
  {
    File file = new File("zipped-text.zip");
    FileOutputStream fos = new FileOutputStream(file);
    ZipOutputStream zos = new ZipOutputStream(fos);
    OutputStreamWriter osw = new OutputStreamWriter(zos);
    BufferedWriter writer = new BufferedWriter(osw);

    zos.putNextEntry(new ZipEntry("text.txt"));

    writer.write("Line one");
    writer.newLine();
    writer.write("Line two");
    writer.newLine();

    writer.close();
  }
}
```

Discussion: How would you store an object that you created in a file (e.g., a Tag Cloud)?

Serializable interface

- An interface with no methods that flags any class as something that can be written to a file

- Use ObjectInputStream's readObject method and ObjectOutputStream's writeObject method to read/write objects

Example

```
public class TagCloud implements Serializable
{
  // Document must also implement Serializable
  private Document document;

  // ...
}
```

Exercise: make this Serializable

```
public class PeriodicTable
{
  private HashMap<String, Atom> atomMap;

  // ...
}
```

Java I/O Summary

- File class
  - Is a name for information on the hard disk
  - Can use this class to create, delete, list files, etc.
- Scanner class
  - Simple class for reading text from a file
- Byte Streams & Character Streams
- Filter Streams

Next Class

- Design Patterns