Module 10

I/O Fundamentals

Objectives

- Write a program that uses command-line arguments and system properties
- Examine the `Properties` class
- Construct node and processing streams, and use them appropriately
- Serialize and deserialize objects
- Distinguish readers and writers from streams, and select appropriately between them

Copyright Copyright 2008 Sun Microsystems, Inc. All Rights Reserved. Sun Services, Revision G.2

Command-Line Arguments

- Any Java technology application can use command-line arguments.
- These string arguments are placed on the command line to launch the Java interpreter after the class name:

```
java TestArgs arg1 arg2 "another arg"
```

- Each command-line argument is placed in the `args` array that is passed to the static `main` method:

```
public static void main(String[] args)
```


Copyright Copyright 2008 Sun Microsystems, Inc. All Rights Reserved. Sun Services, Revision G.2

Command-Line Arguments

```
1   public class TestArgs {
2     public static void main(String[] args) {
3       for ( int i = 0; i < args.length; i++ ) {
4         System.out.println("args[" + i + "] is '" + args[i] + "'");
5       }
6     }
7   }
```

Example execution:

```
java TestArgs arg0 arg1 "another arg"
args[0] is 'arg0'
args[1] is 'arg1'
args[2] is 'another arg'
```


Copyright Copyright 2008 Sun Microsystems, Inc. All Rights Reserved. Sun Services, Revision G.2

System Properties

- System properties are a feature that replaces the concept of *environment variables* (which are platform-specific).
- The `System.getProperties` method returns a `Properties` object.
- The `getProperty` method returns a `String` representing the value of the named property.
- Use the `-D` option on the command line to include a new property.

Copyright Copyright 2008 Sun Microsystems, Inc. All Rights Reserved. Sun Services, Revision G.2

The `Properties` Class

- The `Properties` class implements a mapping of names to values (a `String`-to-`String` map).
- The `propertyNames` method returns an `Enumeration` of all property names.
- The `getProperty` method returns a `String` representing the value of the named property.
- You can also read and write a properties collection into a file using `load` and `store`.

Copyright Copyright 2008 Sun Microsystems, Inc. All Rights Reserved. Sun Services, Revision G.2

The `Properties` Class

```
1   import java.util.Properties;
2   import java.util.Enumeration;
3
4   public class TestProperties {
5     public static void main(String[] args) {
6       Properties props = System.getProperties();
7       props.list(System.out);
8     }
9   }
```


Copyright Copyright 2008 Sun Microsystems, Inc. All Rights Reserved. Sun Services, Revision G.2

The `Properties` Class

The following is an example test run of this program:

```
java -DmyProp=theValue TestProperties
```

The following is the (partial) output:

```
java.runtime.name=Java(TM) SE Runtime Environment
sun.boot.library.path=C:\jse\jdk1.6.0\jre\bin
java.vm.version=1.6.0-b105
java.vm.vendor=Sun Microsystems Inc.
java.vm.name=Java HotSpot(TM) Client VM
file.encoding.pkg=sun.io
user.country=US
myProp=theValue
```


Copyright Copyright 2008 Sun Microsystems, Inc. All Rights Reserved. Sun Services, Revision G.2

I/O Stream Fundamentals

- A *stream* is a flow of data from a source or to a sink.
- A *source* stream initiates the flow of data, also called an input stream.
- A *sink* stream terminates the flow of data, also called an output stream.
- Sources and sinks are both *node streams*.
- Types of node streams are files, memory, and pipes between threads or processes.

Copyright Copyright 2008 Sun Microsystems, Inc. All Rights Reserved. Sun Services, Revision G.2

Fundamental Stream Classes

Stream	Byte Streams	Character Streams
Source streams	`InputStream`	`Reader`
Sink streams	`OutputStream`	`Writer`

Copyright Copyright 2008 Sun Microsystems, Inc. All Rights Reserved. Sun Services, Revision G.2

Data Within Streams

- Java technology supports two types of streams: character and byte.
- Input and output of character data is handled by readers and writers.
- Input and output of byte data is handled by input streams and output streams:
  - Normally, the term *stream* refers to a byte stream.
  - The terms *reader* and *writer* refer to character streams.

Copyright Copyright 2008 Sun Microsystems, Inc. All Rights Reserved. Sun Services, Revision G.2

The `InputStream` Methods

- The three basic `read` methods are:

```
int read()
int read(byte[] buffer)
int read(byte[] buffer, int offset, int length)
```

- Other methods include:

```
void close()
int available()
long skip(long n)
boolean markSupported()
void mark(int readlimit)
void reset()
```


Copyright Copyright 2008 Sun Microsystems, Inc. All Rights Reserved. Sun Services, Revision G.2

The `OutputStream` Methods

- The three basic `write` methods are:

```
void write(int c)
void write(byte[] buffer)
void write(byte[] buffer, int offset, int length)
```

- Other methods include:

```
void close()
void flush()
```


Copyright Copyright 2008 Sun Microsystems, Inc. All Rights Reserved. Sun Services, Revision G.2

The `Reader` Methods

- The three basic `read` methods are:

```
int read()
int read(char[] cbuf)
int read(char[] cbuf, int offset, int length)
```

- Other methods include:

```
void close()
boolean ready()
long skip(long n)
boolean markSupported()
void mark(int readAheadLimit)
void reset()
```


Copyright Copyright 2008 Sun Microsystems, Inc. All Rights Reserved. Sun Services, Revision G.2

The `Writer` Methods

- The basic `write` methods are:

```
void write(int c)
void write(char[] cbuf)
void write(char[] cbuf, int offset, int length)
void write(String string)
void write(String string, int offset, int length)
```

- Other methods include:

```
void close()
void flush()
```


Copyright Copyright 2008 Sun Microsystems, Inc. All Rights Reserved. Sun Services, Revision G.2

Node Streams

Type	Character Streams	Byte Streams
File	`FileReader` `FileWriter`	`FileInputStream` `FileOutputStream`
Memory: array	`CharArrayReader` `CharArrayWriter`	`ByteArrayInputStream` `ByteArrayOutputStream`
Memory: string	`StringReader` `StringWriter`	N/A
Pipe	`PipedReader` `PipedWriter`	`PipedInputStream` `PipedOutputStream`

Copyright Copyright 2008 Sun Microsystems, Inc. All Rights Reserved. Sun Services, Revision G.2

A Simple Example

This program performs a copy file operation using a manual buffer:

java TestNodeStreams file1 file2

```
1   import java.io.*;
2   public class TestNodeStreams {
3     public static void main(String[] args) {
4       try {
5         FileReader input = new FileReader(args[0]);
6         try {
7           FileWriter output = new FileWriter(args[1]);
8           try {
9             char[] buffer = new char[128];
10            int charsRead;
11
12            // read the first buffer
13            charsRead = input.read(buffer);
14            while ( charsRead != -1 ) {
15              // write buffer to the output file
```


Copyright Copyright 2008 Sun Microsystems, Inc. All Rights Reserved. Sun Services, Revision G.2

A Simple Example

```
16              output.write(buffer, 0, charsRead);
17
18              // read the next buffer
19              charsRead = input.read(buffer);
20            }
21
22          } finally {
23            output.close();}
24        } finally {
25          input.close();}
26      } catch (IOException e) {
27        e.printStackTrace();
28      }
29    }
30  }
```


Copyright Copyright 2008 Sun Microsystems, Inc. All Rights Reserved. Sun Services, Revision G.2

Buffered Streams

This program performs a copy file operation using a built-in buffer:

java TestBufferedStreams file1 file2

```
1   import java.io.*;
2   public class TestBufferedStreams {
3     public static void main(String[] args) {
4       try {
5         FileReader input = new FileReader(args[0]);
6         BufferedReader bufInput = new BufferedReader(input);
7         try {
8           FileWriter output = new FileWriter(args[1]);
9           BufferedWriter bufOutput= new BufferedWriter(output);
10          try {
11            String line;
12            // read the first line
13            line = bufInput.readLine();
14            while ( line != null ) {
15              // write the line out to the output file
```


Copyright Copyright 2008 Sun Microsystems, Inc. All Rights Reserved. Sun Services, Revision G.2

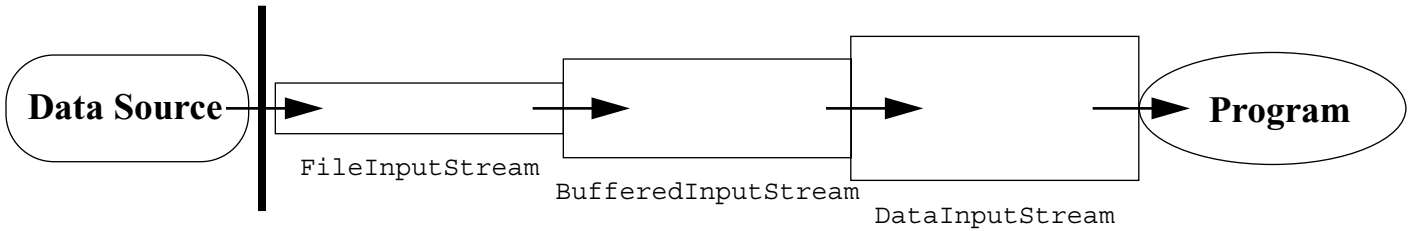
Buffered Streams

```
16              bufOutput.write(line, 0, line.length());
17              bufOutput.newLine();
18              // read the next line
19              line = bufInput.readLine();
20            }
21          } finally {
22            bufOutput.close();
23          }
24        } finally {
25          bufInput.close();
26        }
27      } catch (IOException e) {
28        e.printStackTrace();
29      }
30    }
31  }
32
33
```

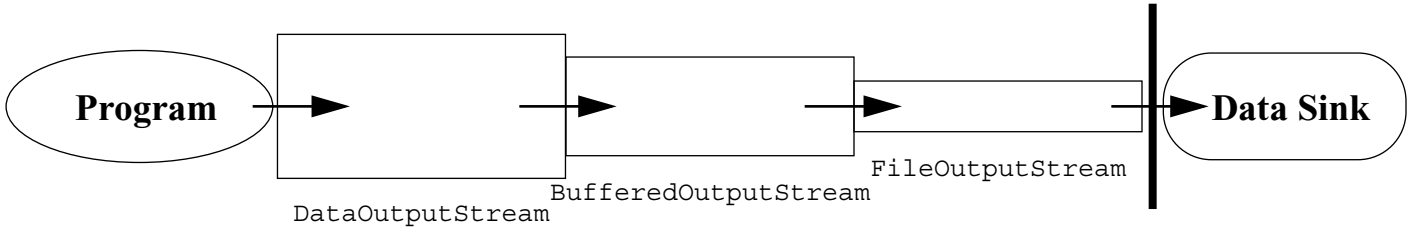

Copyright Copyright 2008 Sun Microsystems, Inc. All Rights Reserved. Sun Services, Revision G.2

I/O Stream Chaining

Input Stream Chain



Output Stream Chain



Copyright Copyright 2008 Sun Microsystems, Inc. All Rights Reserved. Sun Services, Revision G.2

Processing Streams

Type	Character Streams	Byte Streams
Buffering	`BufferedReader` `BufferedWriter`	`BufferedInputStream` `BufferedOutputStream`
Filtering	*`FilterReader`* *`FilterWriter`*	*`FilterInputStream`* *`FilterOutputStream`*
Converting between bytes and character	`InputStreamReader` `OutputStreamWriter`	
Performing object serialization		`ObjectInputStream` `ObjectOutputStream`

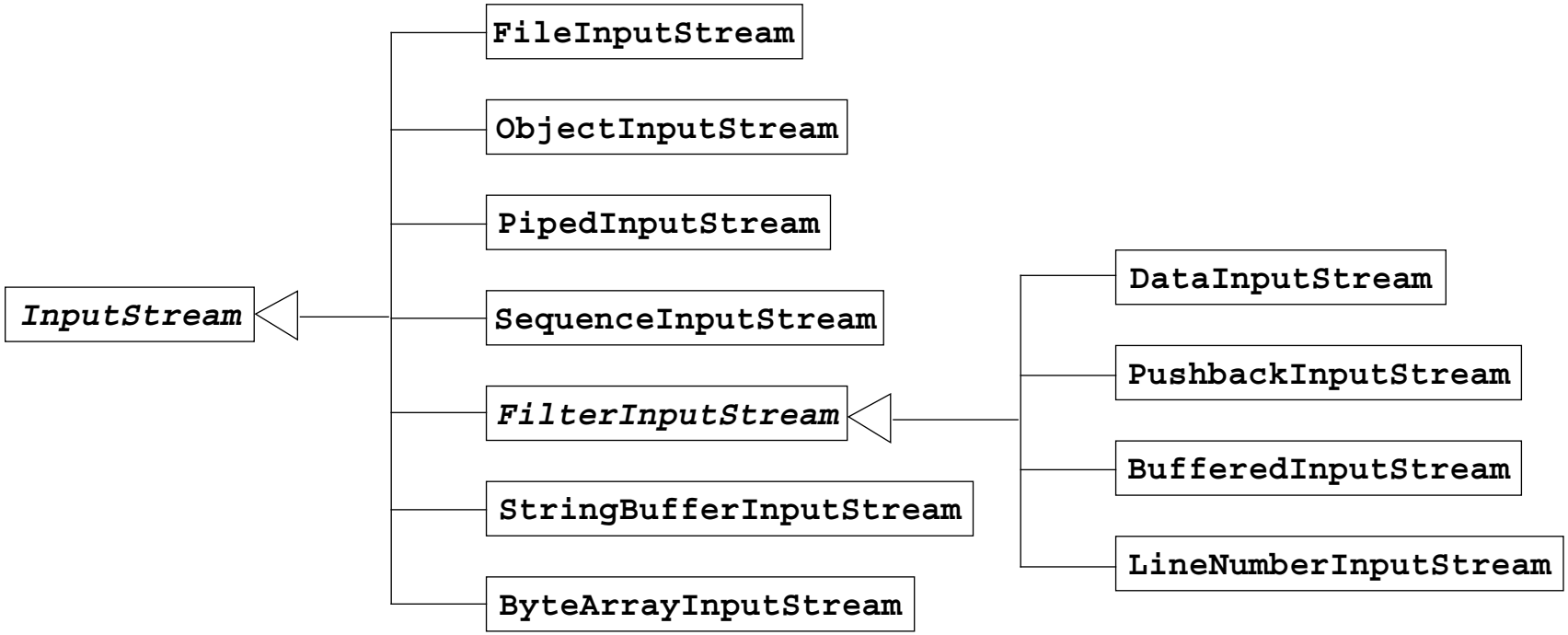
Copyright Copyright 2008 Sun Microsystems, Inc. All Rights Reserved. Sun Services, Revision G.2

Processing Streams

Type	Character Streams	Byte Streams
Performing data conversion		`DataInputStream` `DataOutputStream`
Counting	`LineNumberReader`	`LineNumberInputStream`
Peeking ahead	`PushbackReader`	`PushbackInputStream`
Printing	`PrintWriter`	`PrintStream`

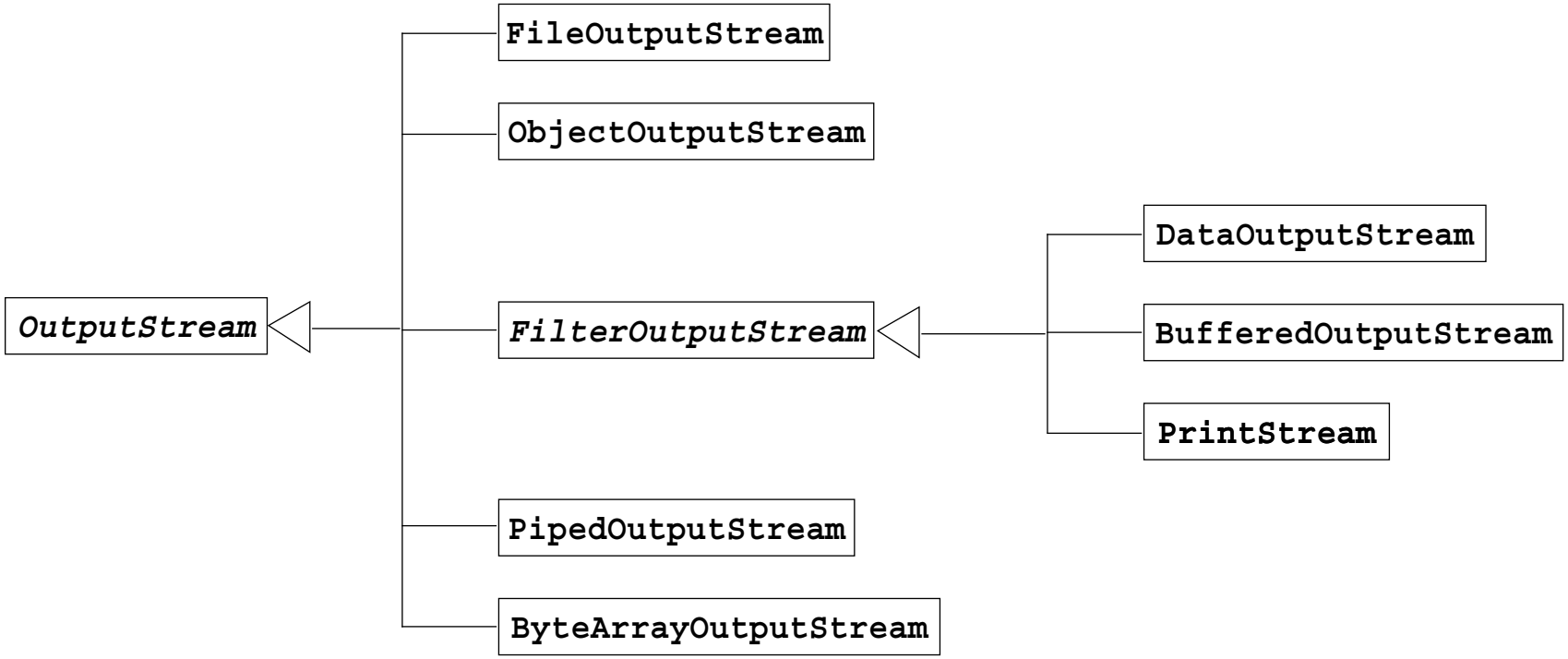
Copyright Copyright 2008 Sun Microsystems, Inc. All Rights Reserved. Sun Services, Revision G.2

The `InputStream` Class Hierarchy



Copyright Copyright 2008 Sun Microsystems, Inc. All Rights Reserved. Sun Services, Revision G.2

The `OutputStream` Class Hierarchy



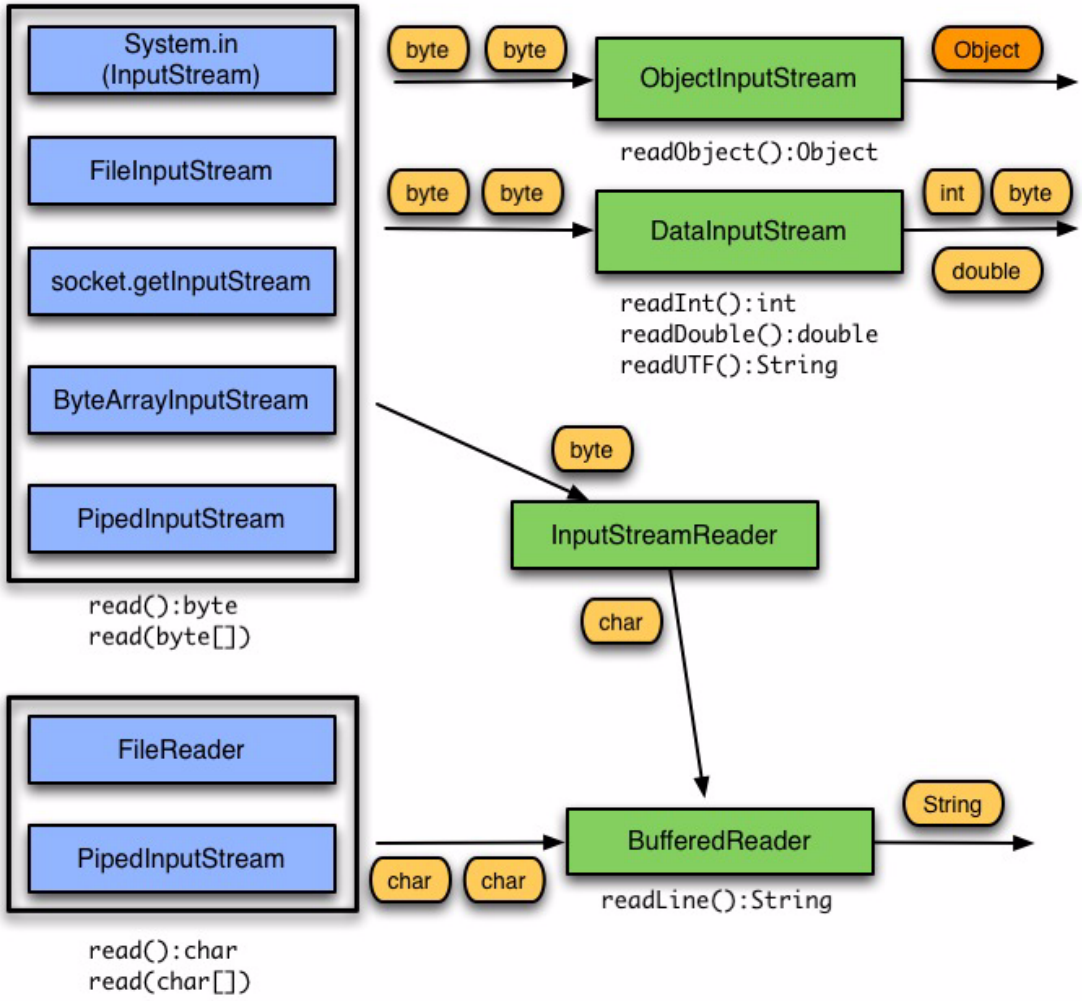
Copyright Copyright 2008 Sun Microsystems, Inc. All Rights Reserved. Sun Services, Revision G.2

The `ObjectInputStream` and The `ObjectOutputStream` Classes

- The Java API provides a standard mechanism (called object serialization) that completely automates the process of writing and reading objects from streams.
- When writing an object, the object input stream writes the class name, followed by a description of the data members of the class, in the order they appear in the stream, followed by the values for all the fields on that object.
- When reading an object, the object output stream reads the name of the class and the description of the class to match against the class in memory, and it reads the values from the stream to populate a newly allocation instance of that class.
- Persistent storage of objects can be accomplished if files (or other persistent storage) are used as streams.

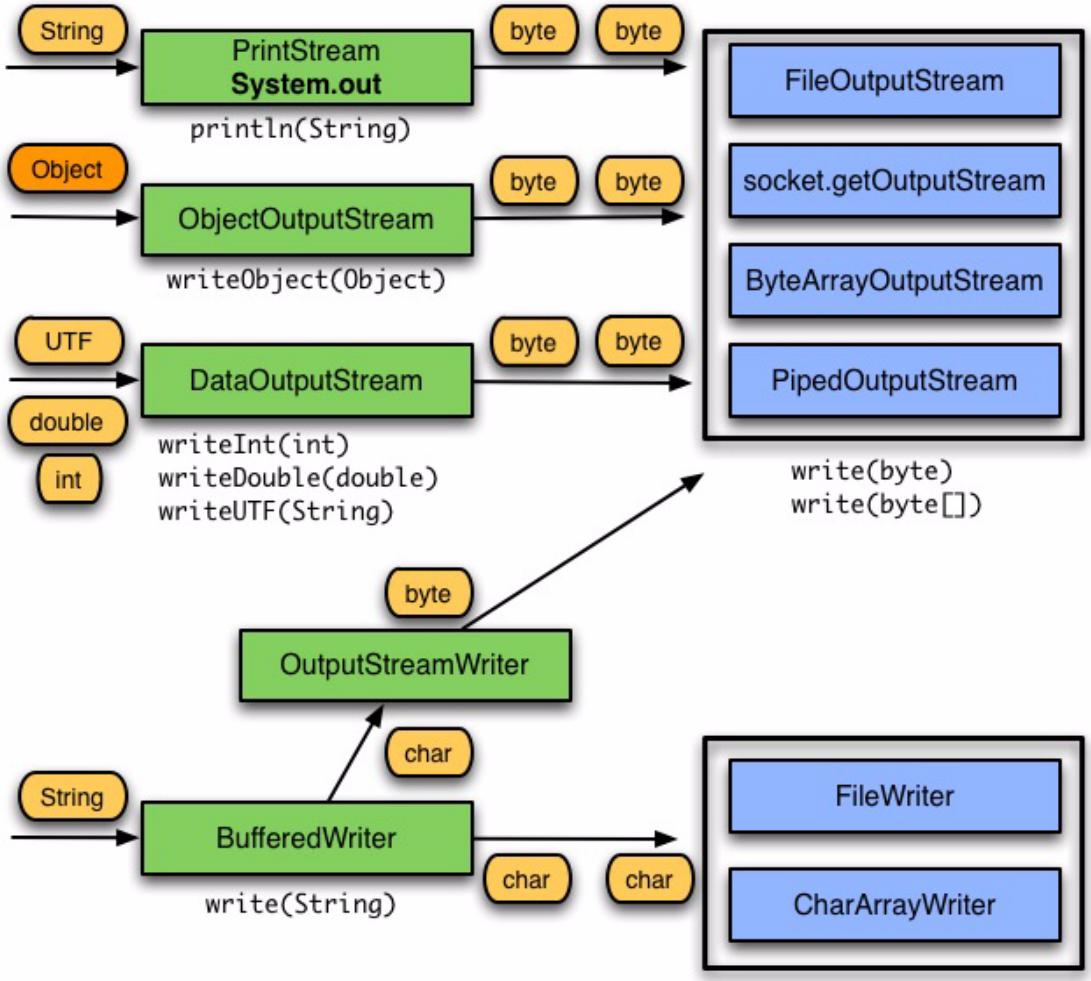
Copyright Copyright 2008 Sun Microsystems, Inc. All Rights Reserved. Sun Services, Revision G.2

Input Chaining Combinations: A Review



Copyright Copyright 2008 Sun Microsystems, Inc. All Rights Reserved. Sun Services, Revision G.2

Output Chaining Combinations: A Review



Copyright Copyright 2008 Sun Microsystems, Inc. All Rights Reserved. Sun Services, Revision G.2

Serialization

- Serialization is a mechanism for saving the objects as a sequence of bytes and rebuilding them later when needed.
- When an object is serialized, only the fields of the object are preserved
- When a field references an object, the fields of the referenced object are also serialized
- Some object classes are not serializable because their fields represent transient operating system-specific information.

Copyright Copyright 2008 Sun Microsystems, Inc. All Rights Reserved. Sun Services, Revision G.2

The `SerializeDate` Class

```
1   import java.io.*;
2   import java.util.Date;
3
4   public class SerializeDate {
5
6     SerializeDate() {
7       Date d = new Date ();
8
9       try {
10        FileOutputStream f =
11            new FileOutputStream ("date.ser");
12        ObjectOutputStream s =
13            new ObjectOutputStream (f);
14        s.writeObject (d);
15        s.close ();
16      } catch (IOException e) {
17        e.printStackTrace ();
18      }
19    }
```


Copyright Copyright 2008 Sun Microsystems, Inc. All Rights Reserved. Sun Services, Revision G.2

The `SerializeDate` Class

```
20
21    public static void main (String args[]) {
22      new SerializeDate();
23    }
24  }
```


Copyright Copyright 2008 Sun Microsystems, Inc. All Rights Reserved. Sun Services, Revision G.2

The `DeSerializeDate` Class

```
1   import java.io.*;
2   import java.util.Date;
3
4   public class DeSerializeDate {
5
6     DeSerializeDate () {
7       Date d = null;
8
9       try {
10        FileInputStream f =
11            new FileInputStream ("date.ser");
12        ObjectInputStream s =
13            new ObjectInputStream (f);
14        d = (Date) s.readObject ();
15        s.close ();
16      } catch (Exception e) {
17        e.printStackTrace ();
18      }
```

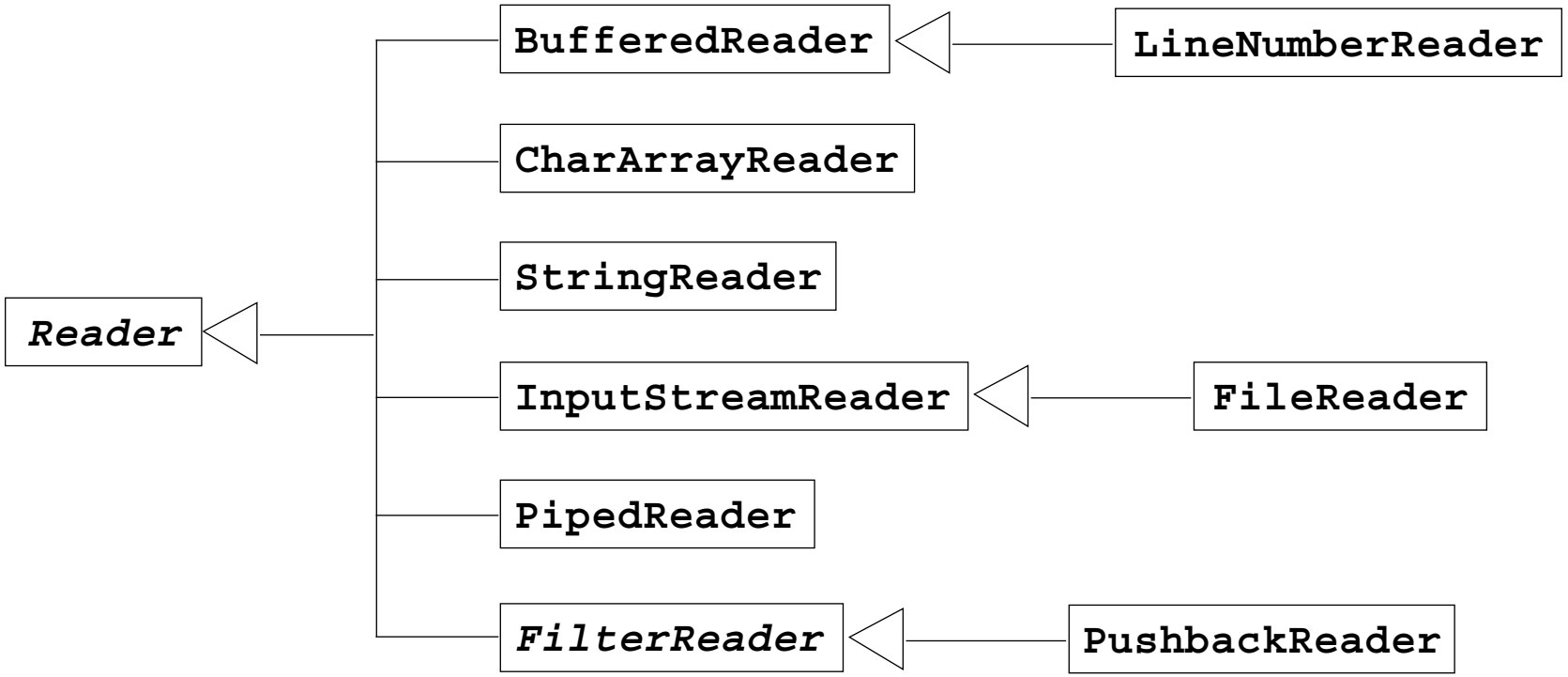

Copyright Copyright 2008 Sun Microsystems, Inc. All Rights Reserved. Sun Services, Revision G.2

The `DeSerializeDate` Class

```
19
20      System.out.println(
21        "Deserialized Date object from date.ser");
22      System.out.println("Date: "+d);
23    }
24
25    public static void main (String args[]) {
26      new DeSerializeDate();
27    }
28  }
```

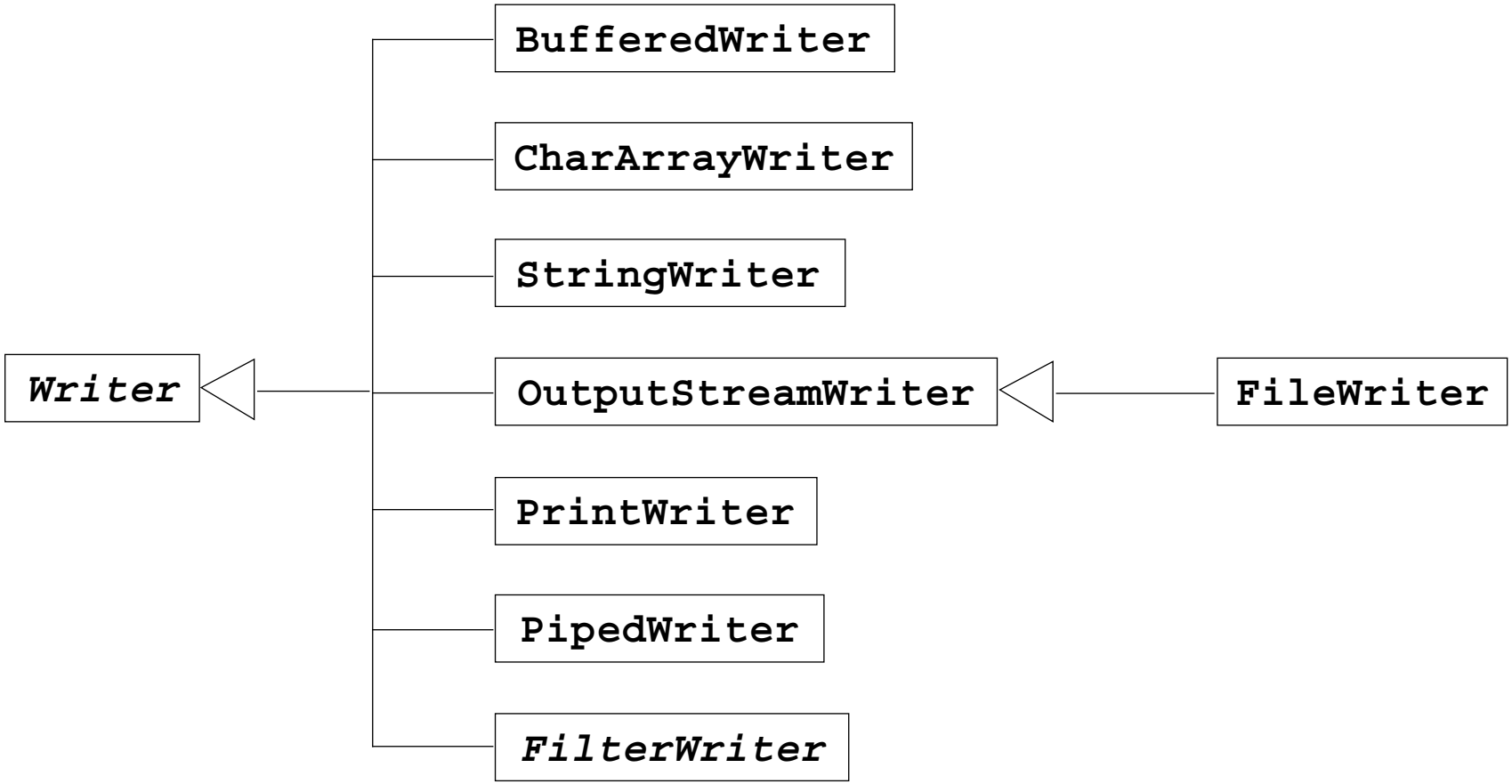

Copyright Copyright 2008 Sun Microsystems, Inc. All Rights Reserved. Sun Services, Revision G.2

The `Reader` Class Hierarchy



Copyright Copyright 2008 Sun Microsystems, Inc. All Rights Reserved. Sun Services, Revision G.2

The `Writer` Class Hierarchy



Copyright Copyright 2008 Sun Microsystems, Inc. All Rights Reserved. Sun Services, Revision G.2