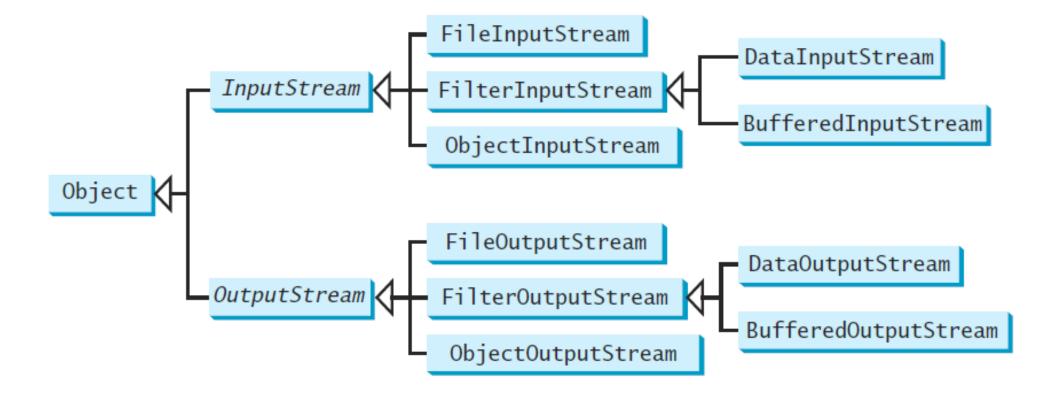
CSCB07 - Software Design I/O and Regular Expressions

Input and Output (I/O)

- Input sources include:
 - Keyboard
 - > File
 - > Network
- Output destinations include:
 - > Console
 - > File
 - > Network

Input and Output Streams

Java handles inputs and outputs using streams



Standard I/O

System.in

- Object of type InputStream
- > Typically refers to the keyboard
- > Reading data could be done using the **Scanner** class. Its methods include:
 - String next()
 - String nextLine()
 - o int nextInt()
 - double nextDouble()

System.out

- Object of type PrintStream
- > Typically refers to the console

The **File** class

- Contains methods for obtaining the properties of a file/directory and for renaming and deleting a file/directory
- Files could be specified using absolute or relative names
- Constructing a File instance does not create a file on the machine
- Methods include:
 - boolean createNewFile()
 - boolean delete()
 - boolean exists()
 - boolean isDirectory()
 - File [] listFiles()

File I/O

- Reading could be done using the Scanner class
 - > E.g. Scanner input = new Scanner(new File(filename));
- Writing could be done using the **FileWriter** class
 - > E.g. FileWriter output = new FileWriter(filename, append);

Regular Expressions

- A regular expression (abbreviated regex) is a string that describes a pattern for matching a set of strings.
- Regular expressions provide a simple and effective way to validate user input
 - > E.g. phone numbers

Regular Expressions

- Java supports regular expressions using the java.util.regex package
- The **Pattern** class can be used to define the pattern
 - The **compile** method takes a string representing the regular expression as an argument and compiles it into a pattern
- The **Matcher** class can be used to search for the pattern. Its methods include:
 - boolean find()
 - boolean matches()
- Example

```
Pattern pattern = Pattern.compile("H.*d");
Matcher matcher = pattern.matcher("Hello World");
System.out.println(matcher.matches());
```

Commonly Used Regular Expressions

Regular Expression	Matches	Example
	any single character	Java matches Ja
(ab cd)	ab or cd	ten matches t(en im)
[abc]	a, b, or c	<pre>Java matches Ja[uvwx]a</pre>
[^abc]	any character except a, b, or c	Java matches Ja[^ars]a
[a-z]	a through z	<pre>Java matches [A-M]av[a-d]</pre>
[^a-z]	any character except a through z	Java matches Jav[^b-d]
[a-e[m-p]]	a through e or m through p	<pre>Java matches [A-G[I-M]]av[a-d]</pre>

Commonly Used Regular Expressions

Regular Expression	Matches	Example
[a-e&&[c-p]]	intersection of a-e with c-p	Java matches [A-P&&[I-M]]av[a-d]
\d	a digit, same as [0-9]	<pre>Java2 matches "Java[\\d]"</pre>
\D	a non-digit	<pre>\$Java matches "[\\D][\\D]ava"</pre>
\w	a word character	<pre>Java1 matches "[\\w]ava[\\w]"</pre>
\W	a non-word character	<pre>\$Java matches "[\\W][\\w]ava"</pre>
\s	a whitespace character	"Java 2" matches "Java\\s2"
\S	a non-whitespace char	<pre>Java matches "[\\S]ava"</pre>

Commonly Used Regular Expressions

Regular Expression	Matches	Example
p*	zero or more occurrences of pattern <i>p</i>	<pre>aaaabb matches "a*bb" ababab matches "(ab)*"</pre>
p+	one or more occurrences of pattern <i>p</i>	<pre>a matches "a+b*" able matches "(ab)+.*"</pre>
p?	zero or one occurrence of pattern p	Java matches "J?Java" Java matches "J?ava"
<i>p</i> {n}	exactly n occurrences of pattern p	<pre>Java matches "Ja{1}.*" Java does not match ".{2}"</pre>
<i>p</i> {n,}	at least n occurrences of pattern p	<pre>aaaa matches "a{1,}" a does not match "a{2,}"</pre>
<i>p</i> {n,m}	between n and m occur- rences (inclusive)	<pre>aaaa matches "a{1,9}" abb does not match "a{2,9}bb"</pre>