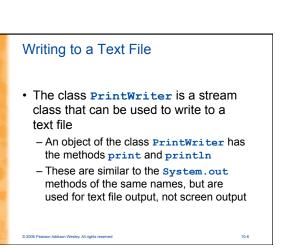


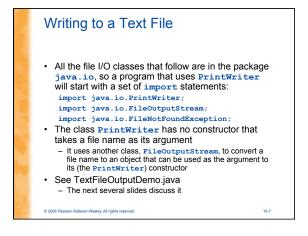
10.5

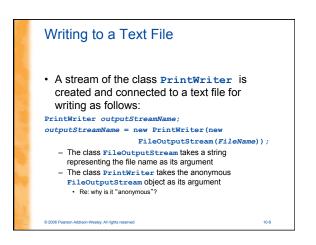
Text Files and Binary Files

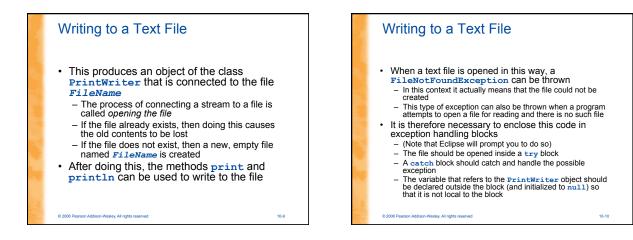
- Files that are designed to be read by programs and that consist of a sequence of binary digits are called *binary files*
 - Binary files are designed to be read on the same type of computer and with the same programming language as the computer that created the file

 - An advantage of binary files is that they are more efficient to process than text files Unlike most binary files, Java binary files have the advantage of being platform independent also









10.11

Writing to a Text File

© 2006 Pearson Addison-Wesley. All rights reserved

 When a program is finished writing to a file, it should always close the stream connected to that file

outputStreamName.close();

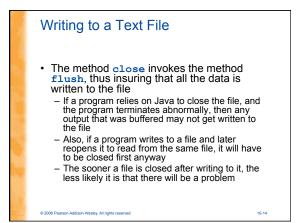
- This allows the system to release any resources used to connect the stream to the file
- If the program does not close the file before the program ends, Java will close it automatically, but it is safest to close it explicitly

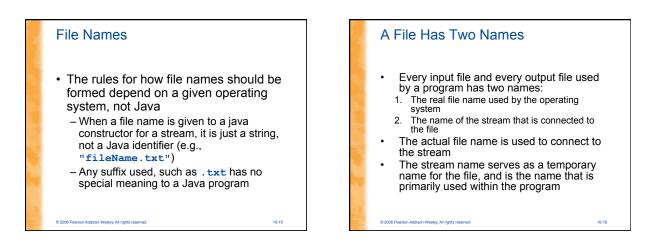


Writing to a Text File

© 2006 Pearson Addison-Wesley. All rights reserved

- Output streams connected to files are usually *buffered*
 - Rather than physically writing to the file as soon as possible, the data is saved in a temporary location (*buffer*)
 - When enough data accumulates, or when the method flush is invoked, the buffered data is written to the file all at once
 - This is more efficient, since physical writes to a file can be slow





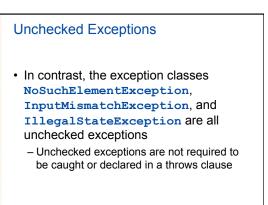
10-13

10-17

IOException

© 2006 Pearson Addison-Wesley. All rights reserved

- When performing file I/O there are many situations in which an exception, such as FileNotFoundException, may be thrown
- Many of these exception classes are subclasses of the class IOException
- The class IOException
 The class IOException is the root class for a variety of exception classes having to do with input and/or output
- These exception classes are all *checked* exceptions
- Therefore, they must be caught or declared in a throws clause

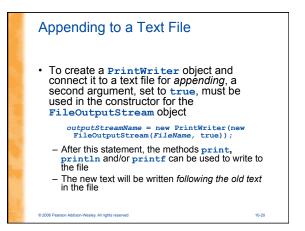


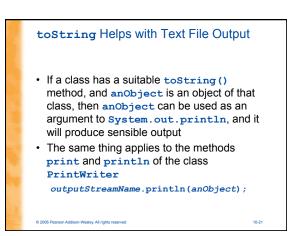
© 2006 Pearson Addison-Wesley. All rights reserved

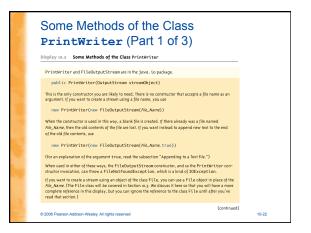
Pitfall: a try Block is a Block

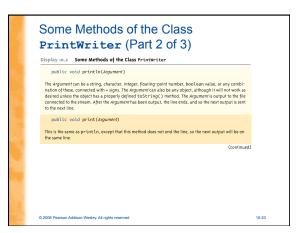
© 2006 Pearson Addison-Wesley, All rights reserved

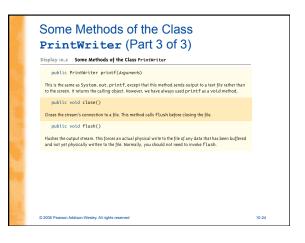
- Since opening a file can result in an exception, it should be placed inside a try block
- If the variable for a **PrintWriter** object needs to be used outside that block, then the variable must be declared outside the block
- Otherwise it would be local to the block, and could not be used elsewhere
- If it were declared in the block and referenced elsewhere, the compiler will generate a message indicating that it is an undefined identifier











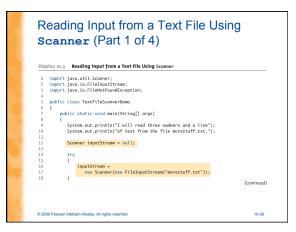
Reading From a Text File Using Scanner

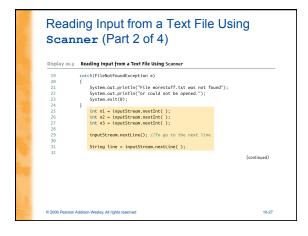
- The class **Scanner** can be used for reading from the keyboard as well as reading from a text file
 - Simply replace the argument System.in (to the Scanner constructor) with a suitable stream that is connected to the text file Scanner StreamObject =

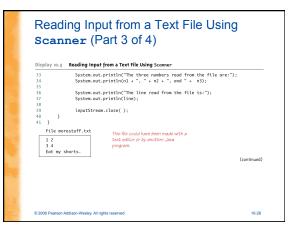
10-25

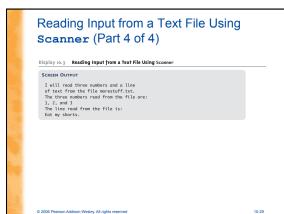
© 2006 Pearson Addison-Wesley, All rights reserved

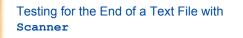
- new Scanner(new FileInputStream(FileName)); Methods of the **Scanner** class for reading input behave the same whether reading from the keyboard or reading from a text file
 - For example, the **nextInt** and **nextLine** methods







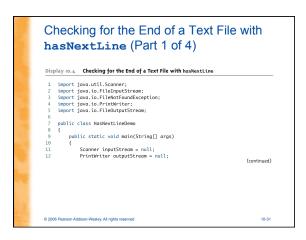


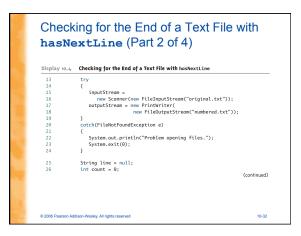


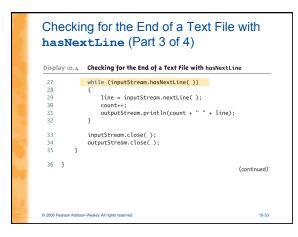
- A program that tries to read beyond the end of a file using methods of the Scanner class will cause an exception to be thrown
- · However, instead of having to rely on an exception to signal the end of a file, the Scanner class provides methods such as hasNextInt and hasNextLine

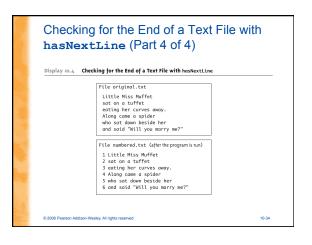
© 2006 Pearson Addison-Wesley. All rights reserved

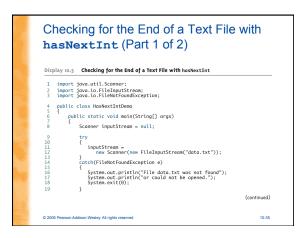
- These methods can also be used to check that the next token to be input is a suitable element of the appropriate type

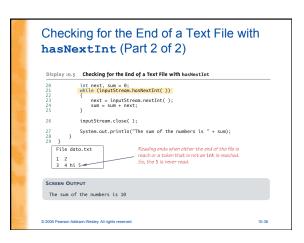












Methods in the Class **Scanner** (Part 1 of 11)

Display 10.6 Methods in the Class Scanner

Scanner is in the java.util package.

© 2006 Pearson Addison-Wesley, All rights reserved

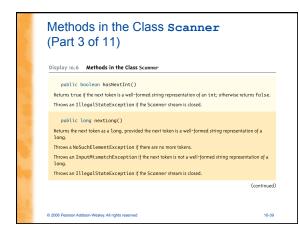
- public Scanner(InputStream streamObject)
- There is no constructor that accepts a file name as an argument. If you want to create a stream using a file name, you can use new Sconner(new FileInputStreom(File_Name))

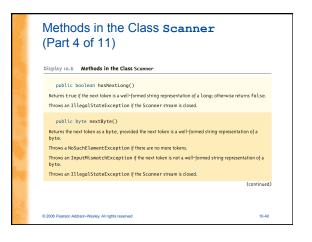
When used in this way, the FileInputStream constructor, and thus the Scanner constructor invocation, can throw a FileNotFoundException, which is a kind of IOException. To crate a stream connected to the keyboard, use new Scanner(System.in)

(continued)

10-37

<section-header><section-header><section-header><section-header><section-header><section-header><text><text><text><text><text><text><text>





Methods in the Class **Scanner** (Part 5 of 11)

Display 10.6 Methods in the Class Scanner

public boolean hasNextByte()

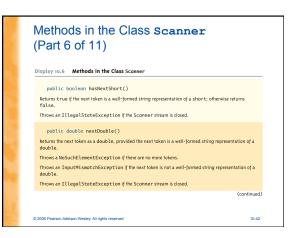
Returns true if the next token is a well-formed string representation of a byte; otherwise returns false. Throws an IllegalStateException if the Scanner stream is closed.

- public short nextShort()
- Returns the next token as a short, provided the next token is a well-formed string representation of a short.
- Throws a NoSuchElementException if there are no more tokens.
- Throws an InputMismotchException if the next token is not a well-formed string representation of a short.

Throws an IllegalStateException if the Scanner stream is closed.

(continued)

10-41



Methods in the Class **Scanner** (Part 7 of 11)

Display 10.6 Methods in the Class Scanner

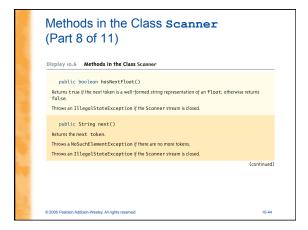
public boolean hasNextDouble()

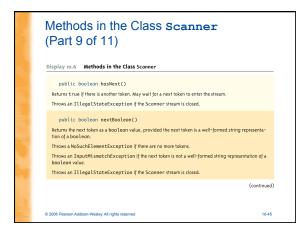
- Returns true if the next token is a well-formed string representation of an double; otherwise returns false.
 Throws an IllegalStateException if the Scanner stream is closed.
- s seamer steam is those.
- public float nextFloat() Returns the next token as a float, provided the next token is a well-formed string representation of a float.
- float. Throws a NoSuchElementException if there are no more tokens.
- Throws an InputMismotchException if the next token is not a well-formed string representation of a floot.

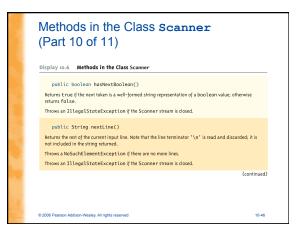
(continued)

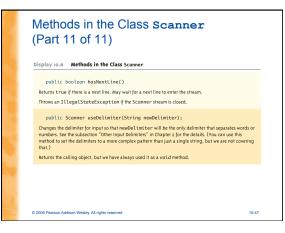
10-43

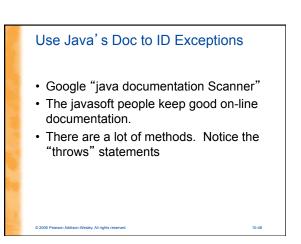
Throws an IllegalStateException if the Scanner stream is closed.











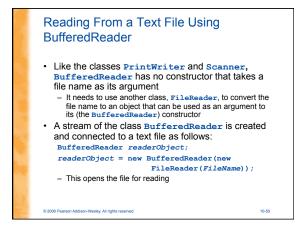
Reading From a Text File Using BufferedReader

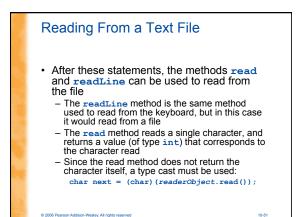
- The class BufferedReader is a stream class that can be used to read from a text file - An object of the class BufferedReader has the methods
 - read and readLine

10-49

- A program using **BufferedReader**, like one using PrintWriter, will start with a set of import statements:
- import java.io.BufferedReader;
- import java.io.FileReader;
- import java.io.FileNotFoundException; import java.io.IOException;

© 2006 Pearson Addison-Wesley, All rights reserved

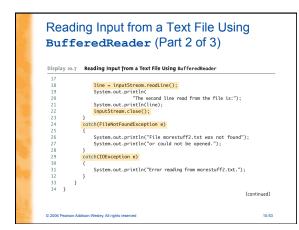


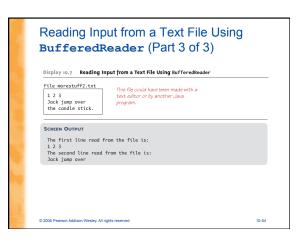


Reading Input from a Text File Using **BufferedReader** (Part 1 of 3) Display 10.7 Reading Input from a Text File Using BufferedReader import iqva.io.BufferedReader: import java.io.FileReader; import java.io.FileReader; import java.io.FileNotFoundException; import java.io.IOException; public class TextFileInputDemo public static void main(String[] args) try { 10 11 12 BufferedReader inputStream = new BufferedReader(new FileReader("morestuff2.txt")); 13 14 15 16

(continued)







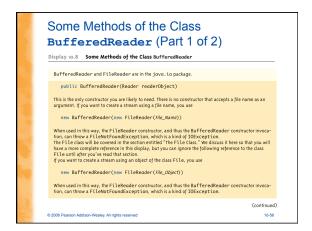
Reading From a Text File

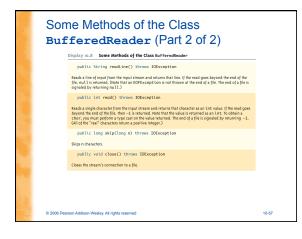
© 2006 Pearson Addison-Wesley. All rights reserved

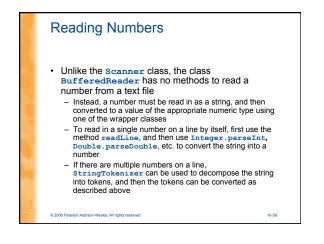
- A program using a BufferedReader object in this way may throw two kinds of exceptions
 - An attempt to open the file may throw a FileNotFoundException (which in this case has the expected meaning)
 - An invocation of **readLine** may throw an **IOException**
 - Both of these exceptions should be handled

10-55

10-59







Testing for the End of a Text File The method readLine of the class BufferedReader returns null when it tries to read beyond the end of a text file

 A program can test for the end of the file by testing for the value null when using readLine

The method read of the class
 BufferedReader returns -1 when it tries to read beyond the end of a text file
 A program can test for the end of the file by testing for the value -1 when using read

© 2006 Pearson Addison-Wesley. All rights reserved

Path Names

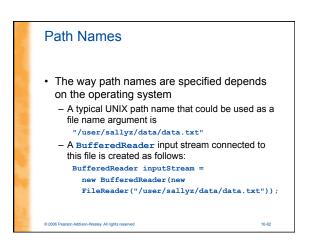
© 2006 Pearson Addison-Wesley. All rights reserved

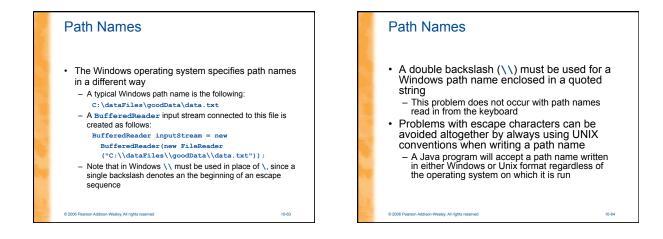
- When a file name is used as an argument to a constructor for opening a file, it is assumed that the file is in the same directory or folder as the one in which the program is run
- If it is not in the same directory, the full or relative path name must be given

Path Names

© 2006 Pearson Addison-Wesley. All rights reserved

- A path name not only gives the name of the file, but also the directory or folder in which the file exists
- A full path name gives a complete path name, starting from the root directory
- A relative path name gives the path to the file, starting with the directory in which the program is located



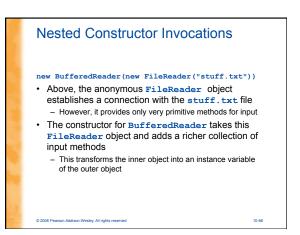


10-61

10-65



- Each of the Java I/O library classes serves only one function, or a small number of functions
 - Normally two or more class constructors are combined to obtain full functionality
- Therefore, expressions with two constructors are common when dealing with Java I/O classes



System.in, System.out, and System.err

- The standard streams System.in, System.out, and System.err are automatically available to every Java program

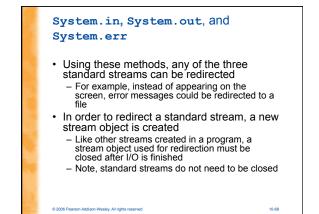
 System.out is used for normal screen output
 - System.out is used for normal screen output
 System.err is used to output error messages to the screen
- The System class provides three methods (setIn, setOut, and setErr) for redirecting these standard streams:

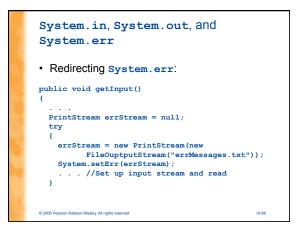
public static void setIn(InputStream inStream)
public static void setOut(PrintStream outStream)
public static void setErr(PrintStream outStream)

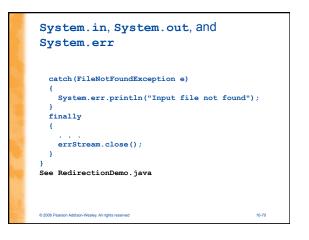
10-67

10-71

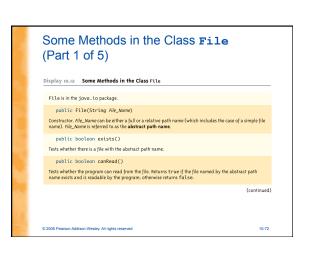
© 2006 Pearson Addison-Wesley, All rights reserved







The File Class The File class is like a wrapper class for file names The constructor for the class File takes a name, (known as the abstract name) as a string argument, and produces an object that represents the file with that name The File object and methods of the class File can be used to determine information about the file and its properties See FileClassDemo.java



Some Methods in the Class File (Part 2 of 5)

Display 10.12 Some Methods in the Class File

public boolean setReadOnly()
Sets the file represented by the abstract path name to be read only. Returns true if successful; otherwise
returns folse.

public boolean canWrite()

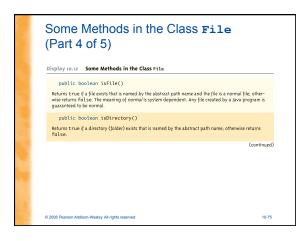
© 2006 Pearson Addison-Wesley, All rights reserved

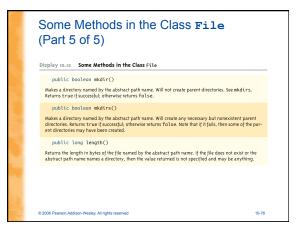
Tests whether the program can write to the file. Returns true if the file named by the abstract path name exists and is writable by the program; otherwise returns false. public booleon delete()

10-73

Tries to delete the file or directory named by the abstract path name. A directory must be empty to be removed. Returns true if it was able to delete the file or directory. Returns false if it was unable to delete the file or directory.

<section-header><section-header><section-header><section-header><section-header><section-header><section-header><section-header><section-header><section-header><section-header>

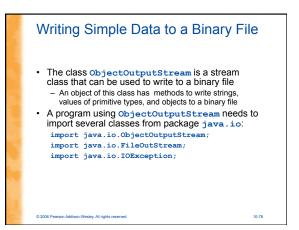


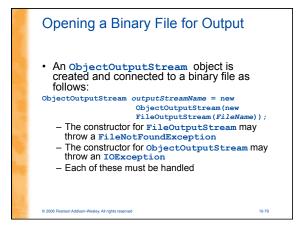


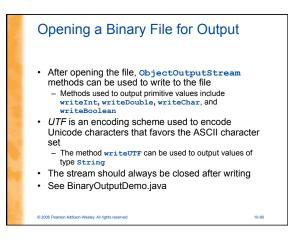
Binary Files

© 2006 Pearson Addison-Wesley. All rights reserved

- Binary files store data in the same format used by computer memory to store the values of variables
 - No conversion needs to be performed when a value is stored or retrieved from a binary file
 - Java binary files, unlike other binary
 - language files, are portable
 - A binary file created by a Java program can be moved from one computer to another
 These files can then be read by a Java program,
 - but only by a Java program





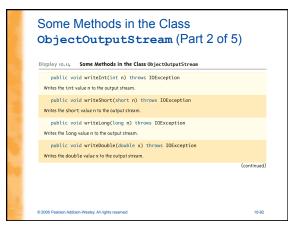


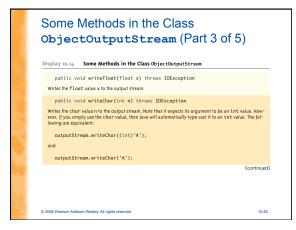
Some Methods in the Classs objectOutputStream (Part 1 of 5) Diplay 10.10 Ome Methods in the class objectOutputStream Diplay 10.10 ObjectOutputStream (OutputStream stream (big) (c) Diplay 10.10 Diplay

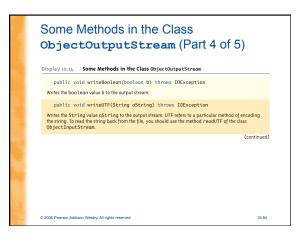
The constructor for FileOutputStream may throw a FileNotFoundException, which is a kind of IOException. If the FileOutputStream constructor succeeds, then the constructor for ObjectOutputStream may throw a different IOException.

(continued)

10-81







Some Methods in the Class ObjectOutputStream (Part 5 of 5)

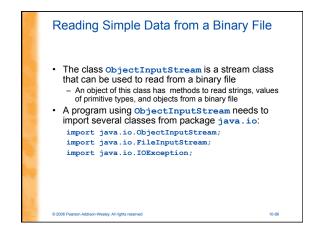
Display 10.14 Some Methods in the Class ObjectOutputStream

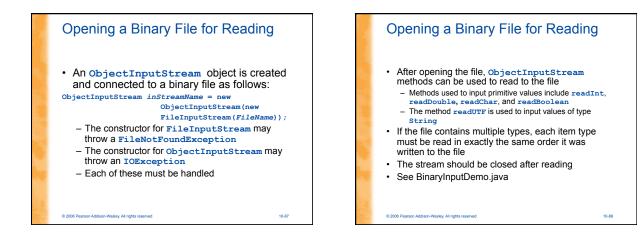
public void writeObject(Object anObject) throws IOException Writes its argument to the output stream. The object argument should be an object of a serializable class, a concept discussed later in this chapter. Throws various IOExceptions.

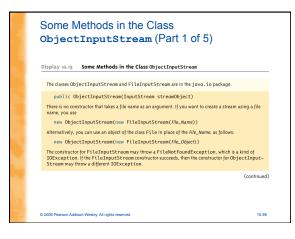
- public void close() throws IOException
- Closes the stream's connection to a file. This method calls flush before closing the file. public void flush() throws IOException

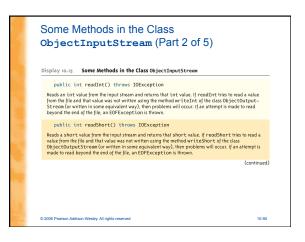
© 2006 Pearson Addison-Wesley, All rights reserved

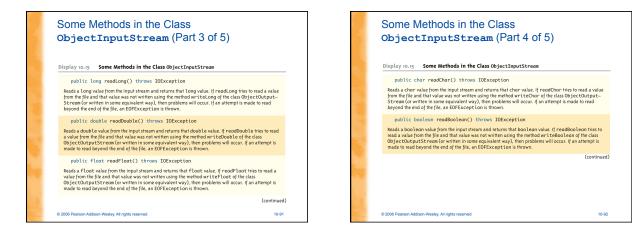
Flushes the output stream. This forces an actual physical write to the file of any data that has been buffered and not yet physically written to the file. Normally, you should not need to invoke flush.

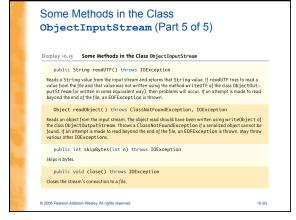


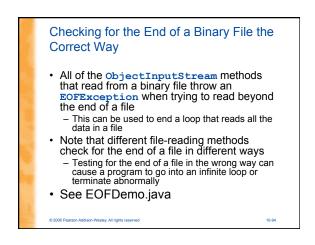


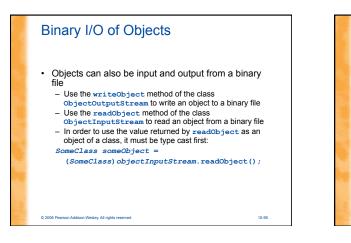


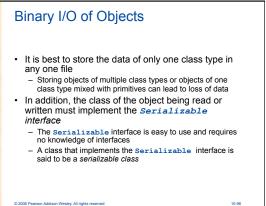










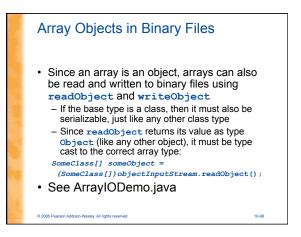


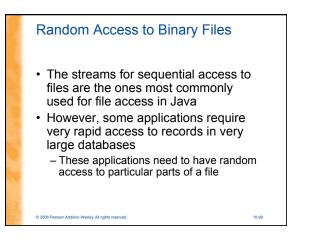
The Serializable Interface

- In order to make a class serializable, simply add implements Serializable to the heading of the class definition public class SomeClass implements Serializable
- When a serializable class has instance variables of a class type, then all those classes must be serializable also
 - A class is not serializable unless the classes for all instance variables are also serializable for all levels of instance variables within classes

10-97

See ObjectIODemo.java





Reading and Writing to the Same File

- The stream class RandomAccessFile, which is in the java.io package, provides both read and write random access to a file in Java
- A random access file consists of a sequence of numbered bytes
 - There is a kind of marker called the *file pointer* that is always positioned at one of the bytes
 - All reads and writes take place starting at the file pointer location
 - The file pointer can be moved to a new location with the method seek

10-100

© 2006 Pearson Addison-Wesley. All rights reserved

Reading and Writing to the Same File

- Although a random access file is byte oriented, there are methods that allow for reading or writing values of the primitive types as well as string values to/from a random access file
 - These include readInt, readDouble, and readUTF for input, and writeInt, writeDouble, and writeUTF for output

© 2006 Pearson Addison-Wesley. All rights reserved

 It does not have writeObject OF readObject methods, however

10-101

<section-header><list-item><list-item><list-item><list-item><list-item><list-item><table-row></table-row></table-row></table-row></table-row></table-row></table-row></table-row></table-row></table-row></table-row></table-row></table-row></table-row>

Pitfall: A Random Access File Need Not Start Empty

- If the file already exists, then when it is opened, the length is not reset to 0, and the file pointer will be positioned at the start of the file
 - This ensures that old data is not lost, and that the file pointer is set for the most likely position for reading (not writing)
- The length of the file can be changed with the setLength method
- In particular, the setLength method can be used to empty the file

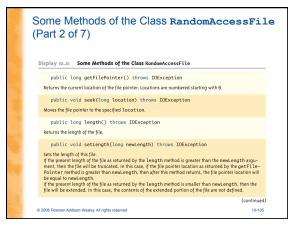
10-103

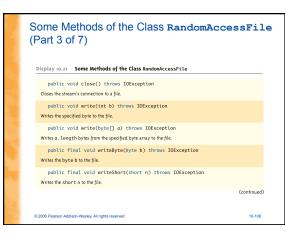
10-107

See RandomAccessDemo.java

© 2006 Pearson Addison-Wesley, All rights reserved

<section-header><section-header><section-header><section-header><section-header><text><text><text><text><text><text><text><text><text><text>





Some Methods of the Class RandomAccessFile (Part 4 of 7)

public final void writeInt(int n) throws IOException	
Writes the int n to the file.	
public final void writeLong(long n) throws IOException	
Writes the long n to the file.	
public final void writeDouble(double d) throws IOException	
Writes the double d to the file.	
public final void writeFloat(float f) throws IOException	
Writes the float f to the file.	
	(contin

© 2006 Pearson Addison-Wesley. All rights reserved

space of the Class RandomAccessFile
public final word writeChar(char) throws DException
write the char c to the file.
 public final word writeBoolean(boolean b) throws DException
 while in the data form the file and tetums it as an integer in the tange o to ags.
 public in the data form the file lie into the array of bytes a. Betums the number of bytes read or
 if the end of the file is encountered.
 ContinueD

© 2006 Pearson Addison-Wesley. All rights reserved

Some Methods of the Class RandomAccessFile (Part 6 of 7)

Display 10.21 Some Methods of the Class RandomAccessFile

- public find(byte readByte() throws IOException Reads a byte value from the file and returns that value. If an attempt is made to read beyond the end of the file, an EOFException is thrown.
- public final short readShort() throws IOException Reads a short value from the file and returns that value. If an attempt is made to read beyond the end of the file, an EOFException is thrown.
- public final int readInt() throws IOException Reads an int value from the file and returns that value. If an attempt is made to read beyond the end of the file, an EOFException is thrown.
- public final long readLong() throws IOException Reads a long value from the file and returns that value. If an attempt is made to read beyond the end of the file, an EOFException is thrown.

(continued)

10-109

