Java I/O and Files Examples

Input and `System.in`

- **interactive program**: Reads input from the console.
  - While the program runs, it asks the user to type input.
  - The input typed by the user is stored in variables in the code.
  - Can be tricky; users are unpredictable and misbehave.
  - But interactive programs have more interesting behavior.
- **`Scanner`**: An object that can read input from many sources.
  - Communicates with `System.in`
  - Can also read from files, web sites, databases, ...

Input and `System.in`

- `System.out`
  - An object with methods named `println` and `print`
- `System.in`
  - not intended to be used directly
  - We use a second object, from a class `Scanner`, to help us.

`Scanner` syntax

- The `Scanner` class is found in the `java.util` package.

  `import java.util.Scanner;`

- Constructing a `Scanner` object to read console input:

  ```
  Scanner name = new Scanner(System.in);
  ```

  - Example:

  ```
  Scanner console = new Scanner(System.in);
  ```

`Scanner` methods

Method	Description
`nextInt()`	reads an `int` from the user and returns it
`nextDouble()`	reads a `double` from the user
`nextLine()`	reads a one-*line* `String` from the user
`next()`	reads a one-word `String` from the user Avoid when Scanner connected to System.in

- Each method waits until the user presses Enter.
- The value typed by the user is returned.
- String nextLine(): Returns the next line of text, or, if you are in the middle of a line, returns the remainder of the line. **Caution: If you are in the middle of a line, nextLine does not return the next line, but instead the remainder of the current line**.

`Scanner` methods

```
System.out.print("How old are you? ");  // prompt
int age = console.nextInt();
System.out.println("You typed " + age);
```

Scanner example

```
import java.util.Scanner;

public class UserInputExample {
    public static void main(String[] args) {
        Scanner console = new Scanner(System.in);

        System.out.print("How old are you? ");
        int age = console.nextInt();

        int years = 65 - age;
        System.out.println(years + " years until
  retirement!");
    }
}
```

age	29
years	36

29

- Console (user input underlined):

```
How old are you? 29
36 years until retirement!
```

`Scanner` example 2

- The `Scanner` can read multiple values from one line.

```
import java.util.Scanner;
public class ScannerMultiply {
    public static void main(String[] args) {
        Scanner console = new Scanner(System.in);

        System.out.print("Please type two numbers: ");
        int num1 = console.nextInt();
        int num2 = console.nextInt();

        int product = num1 * num2;
        System.out.println("The product is " + product);
    }
}
```

- Output (user input underlined):

```
Please type two numbers: 8 6
The product is 48
```

Input tokens

- **token**: A unit of user input, as read by the `Scanner`.
  - Tokens are separated by *whitespace* (spaces, tabs, newlines).
  - How many tokens appear on the following line of input?

```
23  John Smith   42.0 "Hello world"   $2.50    "
19"
```

- When a token is not the type you ask for, it crashes.

```
System.out.print("What is your age? ");
int age = console.nextInt();
```

Output:

```
What is your age? Timmy
java.util.InputMismatchException
        at java.util.Scanner.next(Unknown
Source)
        at java.util.Scanner.nextInt(Unknown
Source)
        ...
```

Boolean Methods

We said that the Scanner methods that read numeric data throw a InputMismatchException exception if the next value isnt what the method expects.

We can avoid that problem using Boolean methods.

Here are four useful Boolean methods that allow us to check to be sure that the next value is what we expect.

Method	*Returns*
boolean hasNextLine()	Returns true if the scanner has another line in its input; false otherwise.
boolean hasNextInt()	Returns true if the next token in the scanner can be interpreted as an int value.
boolean hasNextFloat()	Returns true if the next toke in the scanner can be interpreted as a float value.

Using try-catch block

```
import java.util.* ;
public class SquareUser
{  public static void main ( String[] a )
  { Scanner scan = new Scanner( System.in  );
    int num = 0 ;
    boolean goodData = false;
    while ( !goodData )
    {System.out.print("Enter an integer: ");
       try
      { num = scan.nextInt();
        goodData = true;
      }  catch (InputMismatchException  ex )
      { System.out.println("You entered bad data." );
        System.out.println("Please try again.\n" );
        String flush = scan.next();
      }    }    System.out.println("The square of " + num + " is " + num*num );
  }}
```

Add more efficiency

- `BufferedReader` reads text from a character-input stream, buffering characters

BufferedReader br = new BufferedReader(new FileReader(args[0]));

- Create a class **EfficientReader** which has a main method which will read a full file
- **`Loop`** **`br.readLine()`** to read from the file line by line

```
import java.io.*;
public class EfficientReader {
   public static void main (String[] args) {
     try {
         BufferedReader br = new BufferedReader(new
                                                     FileReader(args[0]));
        String line = br.readLine();
        while (line != null) {
              System.out.println("Read a line:");
              System.out.println(line);
              line = br.readLine();
        }
        br.close();
     } catch(FileNotFoundException fe) {
          System.out.println("File not found: “+ args[0]");
     } catch(IOException ioe) {
          System.out.println("Can’t read from file: “+args[0]);
     }  }  }
```

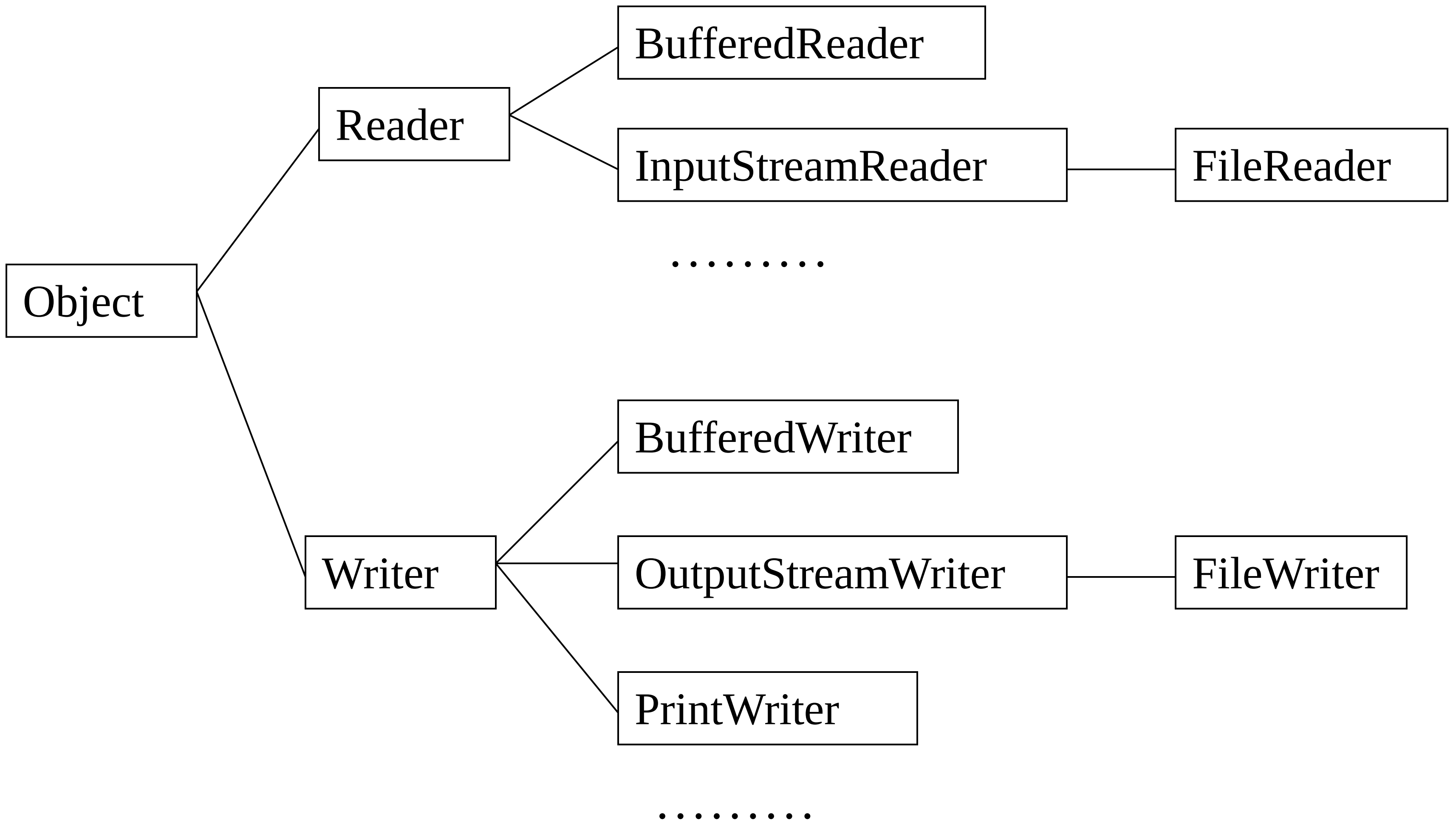
Files and Directories

```
class FileDemo {
  public static void main(String[] args) {
    try {
     // Display Constant
       System.out.println("pathSeparatorChar =
" +        File.pathSeparatorChar);
       System.out.println("separatorChar = " +
         File.separatorChar);

       // Test Some Methods
         File file = new File(args[0]);
        System.out.println("getName()
= " +   file.getName());

System.out.println( "getParent() =
" +     file.getParent());

System.out.println("getAbsolutePath() =
" +         file.getAbsolutePath());
        System.out.println("getPath() = " +
          file.getPath());
```

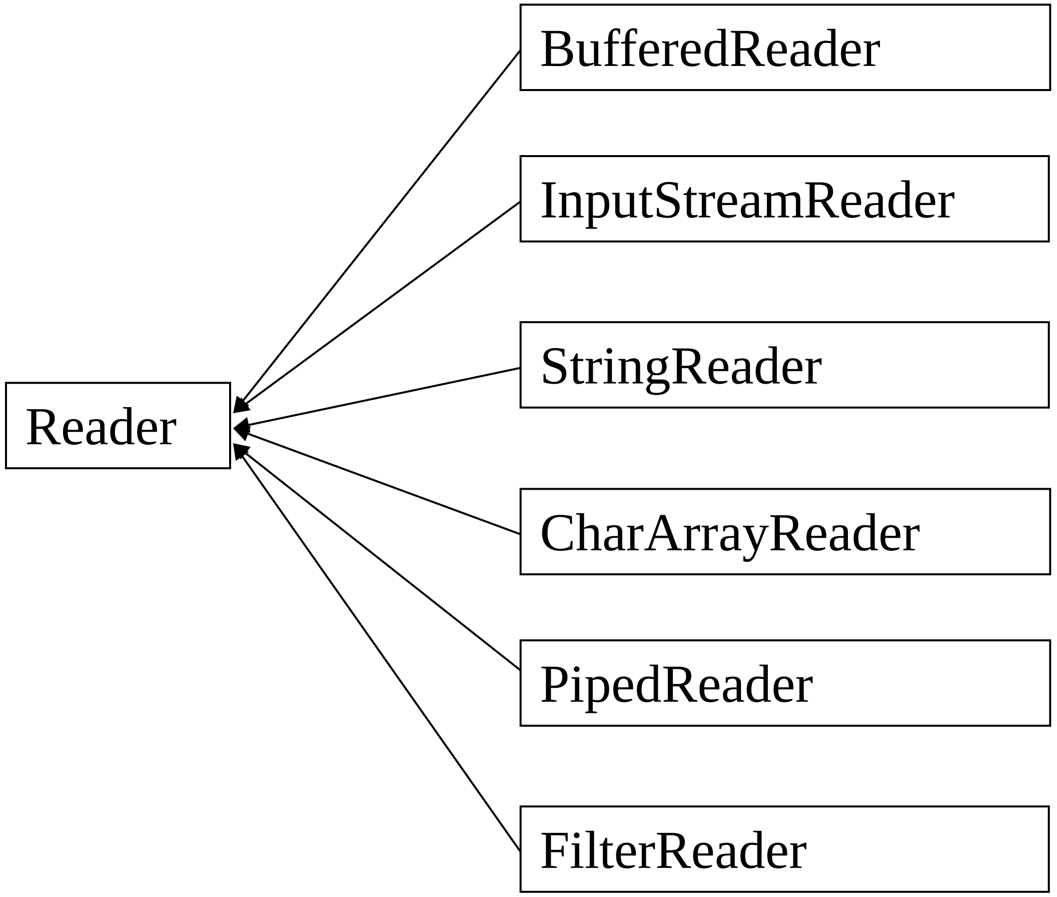
```
System.out.println("canRead() = " +
        file.canRead());
      System.out.println("canWrite() = "
+
        file.canWrite());
    }
    catch(Exception e) {
      e.printStackTrace();
    }  }
}
```

```
Result :
pathSeparatorChar = ;
separatorChar = \
getName() = FileDemo.java
getParent() = null
getAbsolutePath() = D:\lecture\FileDemo.java
getPath() = FileDemo.java
canRead() = true
canWrite() = true
```

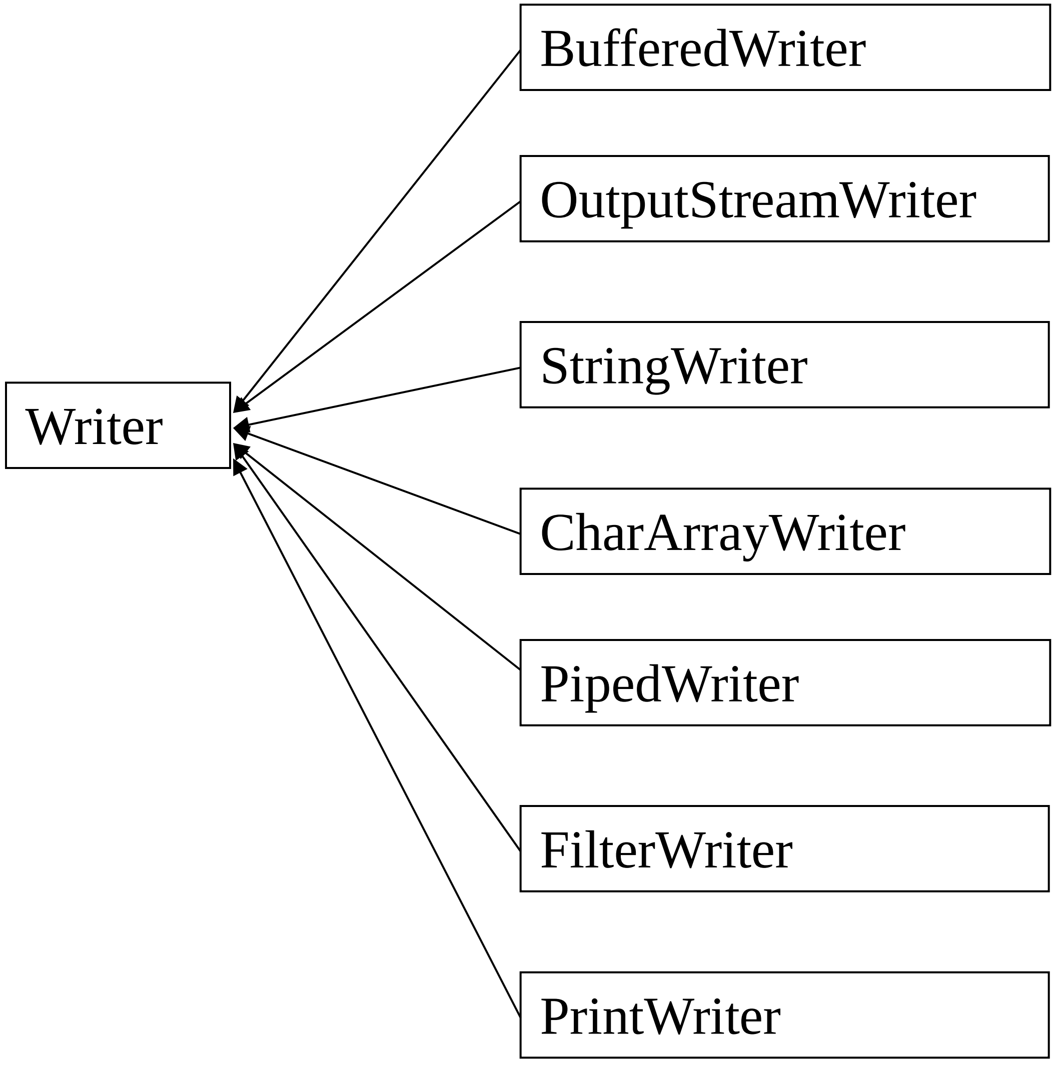
Character Streams



Character Streams



Character Streams



Character Streams

Writer Constructors

Writer()

Writer(Object obj)

Methods Defined by the Writer

abstract void close() throws IOException

abstract void flush() throws IOException

void write(int c) throws IOException

void write(char buffer[]) throws IOException

abstract void write(char buffer[], int index, int size) throws IOException

void write(String s) throws IOException

void write(String s, int index, int size) throws IOException

OutputStreamWriter Constructors

OutputStreamWriter(OutputStream os)

OutputStreamWriter(OutputStream os, String encoding)

getEncoding() Method

String getEncoding()

FileWriter Constructors

FileWriter(String filepath) throws IOException

FileWriter(String filepath, boolean append) throws IOException

FileWriter(String filepath) throws IOException

Refer tohttp://java.sun.com/j2se/1.5.0/docs/api/java/io/Writer.html

Character Streams

Methods Defined by the Reader

abstract void close() throws IOException

void mark(int numChars) throws IOException

boolean markSupported()

int read() throws IOException

int read(char buffer[]) throws IOException

int read(char buffer[], int offset, int numChars) throws IOException

boolean ready() throws IOException

void reset() throws IOException

int skip(long numChars) throws IOException

InputStreamWriter Constructors

InputStreamWriter(InputStream os)

InputStreamWriter(InputStream os, String encoding)

getEncoding() Method

String getEncoding()

FileReader Constructors

FileReader(String filepath) throws FileNotFoundException

FileReader(File fileObj) throws FileNotFoundException

Refer tohttp://java.sun.com/j2se/1.5.0/docs/api/java/io/Reader.html

Character Stream Examples

```
import java.io.*;
class FileWriterDemo {
  public static void main(String[] args) {
    try {
      // Create a FileWriter
       FileWriter fw = new
         FileWriter(args[0]);
       // Write string to the file
       for (int i = 0; i < 12; i++) {
         fw.write("Line " + i + "\n");
       }
       // Close a FileWriter
       fw.close();
    }
    catch (Exception e) {
      System.out.println("Exception: " + e);
    }
  }
}
```

```
class FileReaderDemo {
  public static void main(String[] args) {
    try {
      FileReader fr = new FileReader(args[0]);
      int i;
      while((i = fr.read()) != -1) {
        System.out.print((char)i);
      }
     fr.close();
    }
    catch(Exception e) {
      System.out.println("Exception: " + e);
    }
  }
}
```

Run :

```
java FileWriterDemo output.txt
java FileReaderDemo output.txt
```

Result :

```
Line 0
Line 1
Line 2
Line 3
Line 4
Line 5
Line 6
Line 7
Line 8
Line 9
Line 10
Line 11
```

Buffered Character Streams

BufferedWriter Constructors

BufferedWriter(Writer w)

BufferedWriter(Writer w, int bufSize)

newLine() Method

void newLine() throws IOException

BufferedReader Constructors

BufferedReader(Reader r)

BufferedReader(Reader r, int bufSize)

readLine() Method

String readLine() throws IOException

```
import java.io.*;
class BufferedWriterDemo {
  public static void main(String[] args) {
    try {
      FileWriter fw = new FileWriter(args[0]);
      BufferedWriter bw = new
BufferedWriter(fw);
      for (int i = 0; i < 12; i++) {
        bw.write("Line " + i + "\n");
      }
      bw.close();
    }
    catch (Exception e) {
      System.out.println("Exception: " + e);
    }
  }
}
```

Refer to
http://java.sun.com/j2se/1.5.0/docs/api/java/io/BufferedWriter.html
http://java.sun.com/j2se/1.5.0/docs/api/java/io/BufferedReaderr.html

Character Stream Examples

```
class BufferedReaderDemo {
  public static void main(String[] args) {
    try {
      FileReader fr = new
FileReader(args[0]);
      BufferedReader br = new
BufferedReader(fr);
      String s;
      while((s = br.readLine()) != null)
        System.out.println(s);
      fr.close();
    }
    catch (Exception e) {
      System.out.println("Exception: " +
e);
    }
  }
}
```

Run :

```
java BufferedWriterDemo output.txt
java BufferedReaderDemo output.txt
```

```
class ReadConsole {
  public static void main(String[] args) {
    try {
      InputStreamReader isr =
        new InputStreamReader(System.in);
      BufferedReader br = new
BufferedReader(isr);
      String s;
      while((s = br.readLine()) != null) {
        System.out.println(s.length());
      }
      isr.close();
    }
    catch (Exception e) {
      System.out.println("Exception: " + e);
    }
  }
}
```

Result :

```
Line 0
Line 1
Line 2
Line 3
Line 4
Line 5
Line 6
Line 7
Line 8
Line 9
Line 10
Line 11
```

```
import java.io.*;
class Str{
public static void main(String args[])throws IOException
{
String s1 = new String();
String s2=  new String();
BufferedReader br = new BufferedReader(new  InputStreamReader(System.in));
s1=br.readLine();
s2=br.readLine();
if((s1.compareTo(s2))==0)
System.out.println("equal");
else
System.out.println("not equal");
s1=s1.concat("two");
System.out.println(s1);
}
}
```

Reading and Writing Files

In java, all files are byte-oriented.

```
import java.io.*;
class DisplayFile{
    public static void main(String args[])throws IOException{
     int i;
    FileInputStream  fin;
    try{
         fin = new FileInputStream(args[0]);
    }catch(FileNotFoundException e){ return;}
      do{
          i=fin.read();
          if(i!=-1) System.out.println((char)i);
      } while(i!=-1);
    fin.close();} }
```

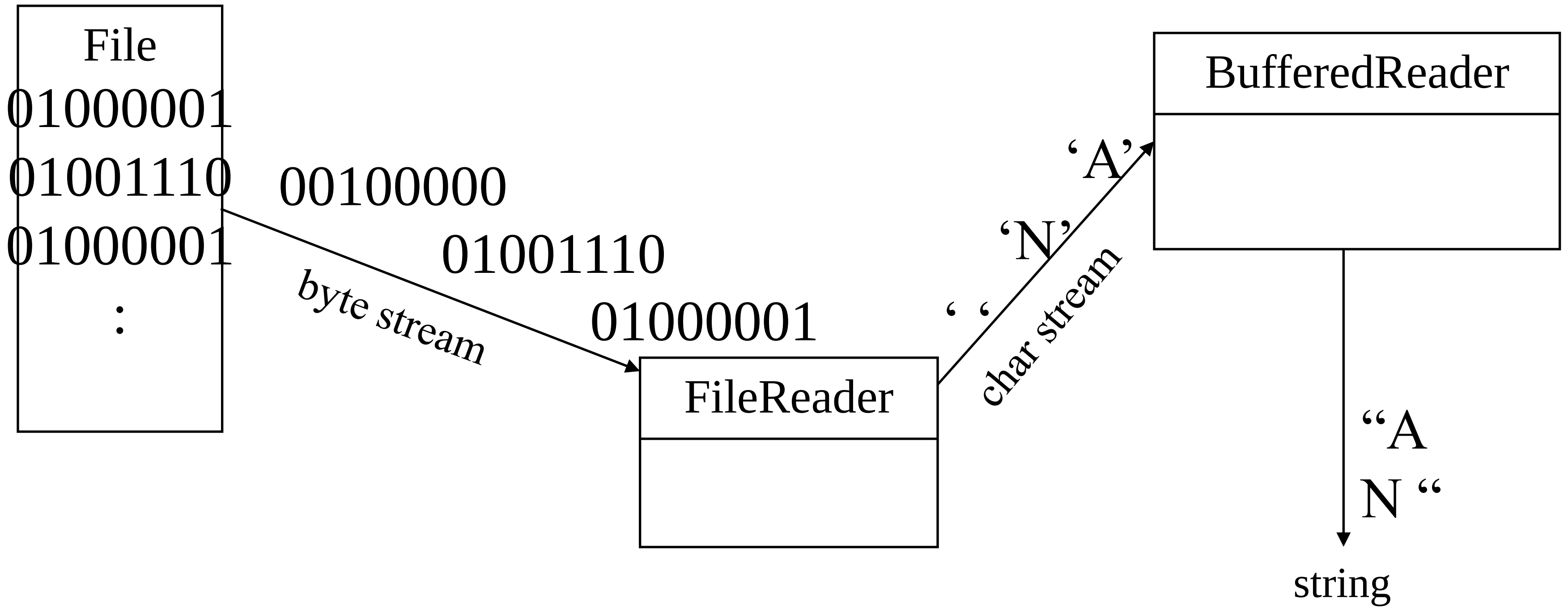
Copy on File to another File

```
FileInputStream fin;
FileOutputStream fout;
fin = new FileInputstream(args[0]);
fout = new FileOutputStream(args[1]);
do{
    int i=fin.read();
    if(i!=-1)fout.write(i);
} while(i!=-1);

finally{
     fin.close();
     fout.close();
```

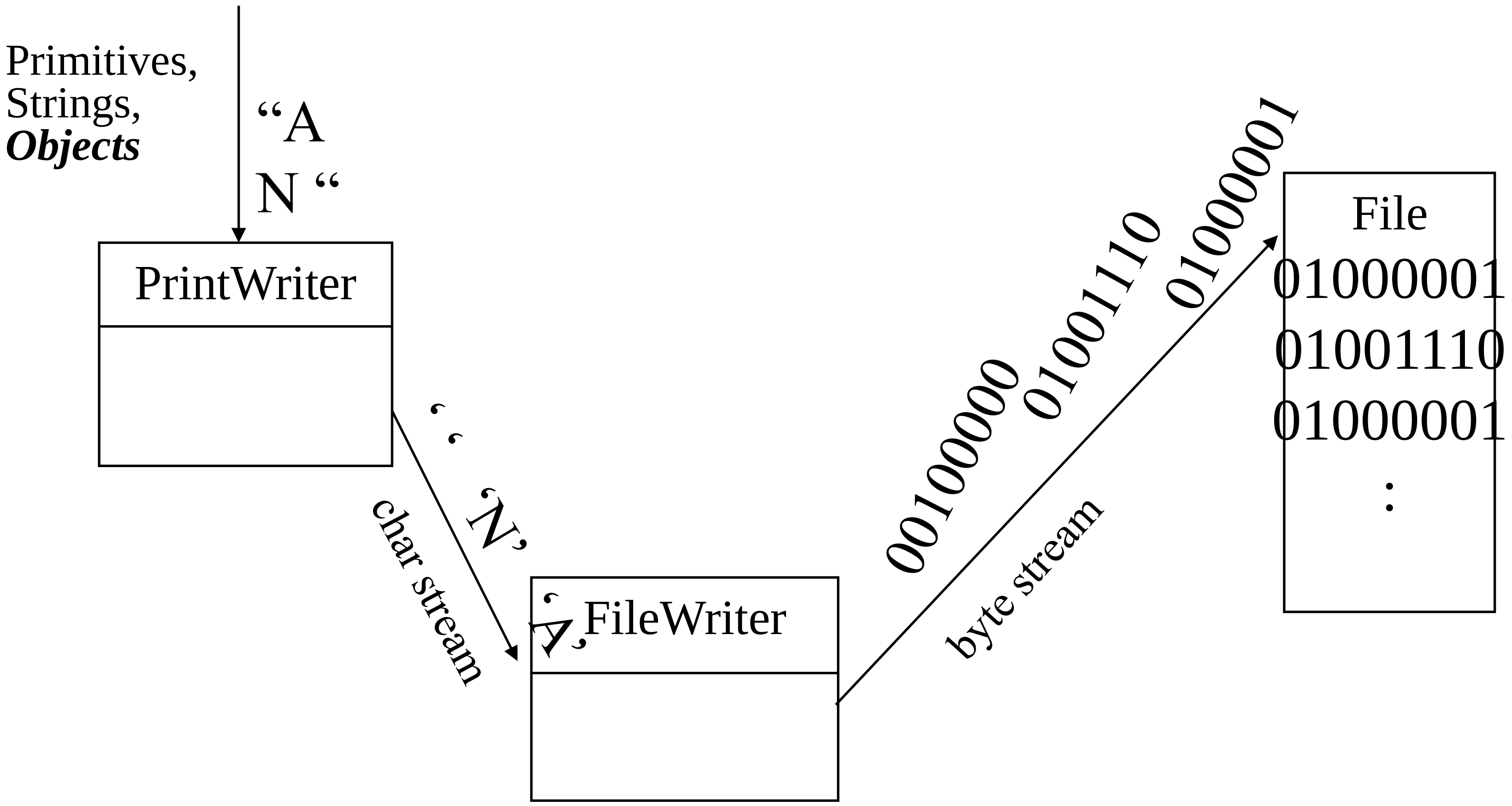
Text Files:Reading and Writing

```
import java.io.*;

public class InAndOut {

public static void main(String[] args)throws
IOException {

  File inputFile = new File(“myInfile.txt");

  File outputFile = new File(“myOutFile.txt");

  FileReader inData = new
FileReader(inputFile);

  FileWriter outData = new
FileWriter(outputFile); int c;

     while ((c = inData.read()) != -1)

          outData.write(c);   inData.close();

     outData.close(); } }
```

Reading Text Input From A File



Writing Text Output To A File



Creating a Text File

```
import java.io.*;
import java.awt.*;
public class PriceListWriter
{
 public static void main( String args[ ]  ) throws IOException
  {
   PrintWriter outfile = new PrintWriter(
                          new BufferedWriter(
                           new FileWriter(
                             new File( "pricelist.txt" ) ) ) );

    outfile.println( "Sugar" );
    outfile.println( "0.84" );
    outfile.println( "Butter" );
    outfile.println( "1.02" );

    outfile.close( );

    System.exit( 0 );
  }
}
```

Reading a Text File

```
import java.io.*;

public class PriceListReader
{
  public static void main( String args[ ] ) throws IOException
  {
    String line;
    BufferedReader infile = new BufferedReader(
                               new FileReader (
                                new File( "pricelist.txt" ) ) );
    line = infile.readLine( );
    while (line != null){
      System.out.println(line);
      line = infile.readLine( );
    }
    System.out.println(“End of list”);
    infile.close( );
    System.exit(0);
  }
}
```

Creating a File through the keyboard

1. Open file

```
import java.io.* ;
class KeyboardToDisk {
    public static void main( String args[ ] ) throws IOException {
        String str ;
        FileWriter fw ;
        BufferedReader br =
            new BufferedReader( new InputStreamReader( System.in ) ) ;
        try {  fw = new FileWriter ( "test.txt" ) ;      } // try
        catch( IOException e) {      System.out.println( "Cannot open file." ) ;
            return ; } // catch
```

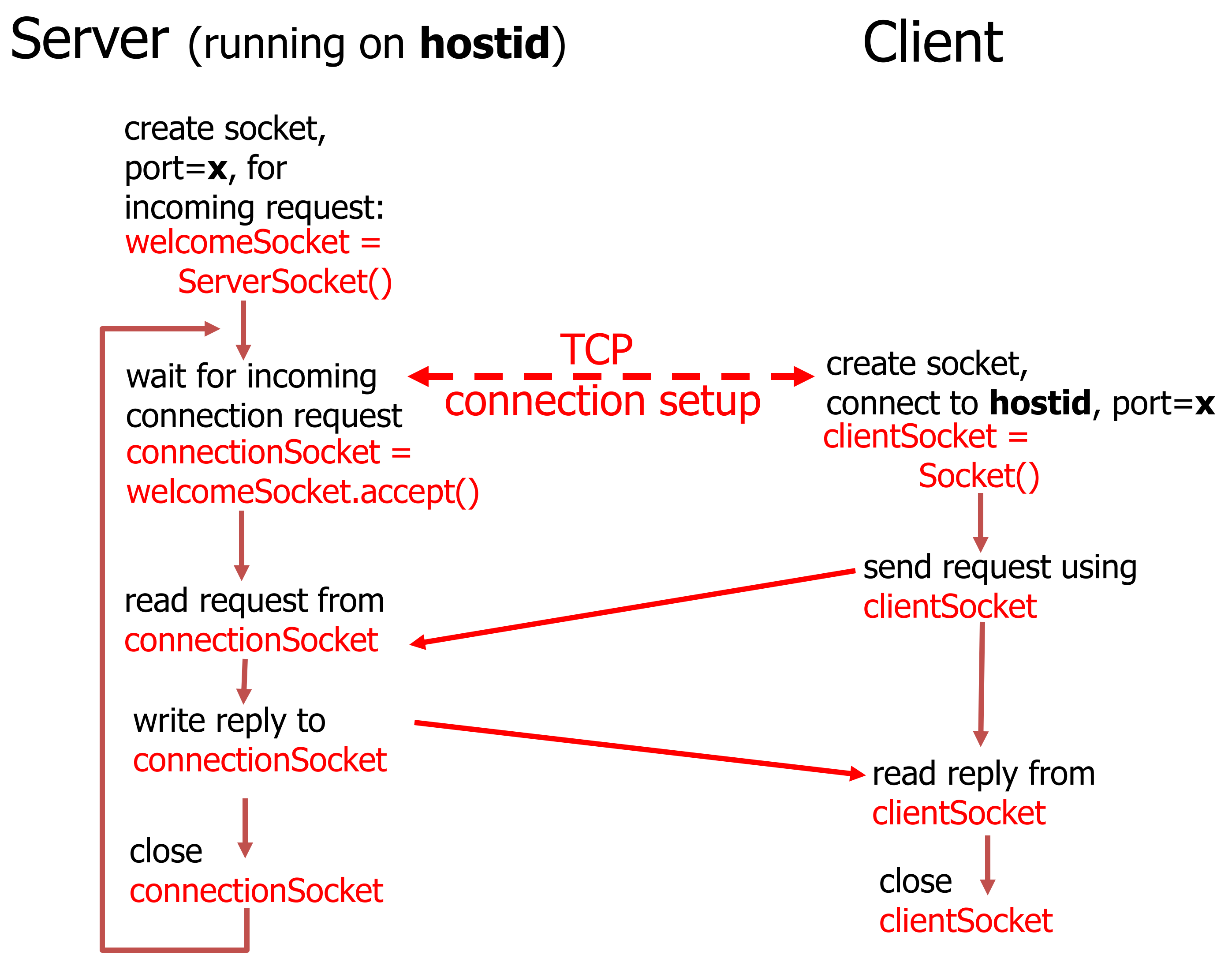
Creating a File through the keyboard

2. Get text and write it out to file

```
System.out.println( "Enter text ('stop' to quit)." ) ;
        do {
                System.out.print( ": " ) ;
                str = br.readLine( ) ;
                if (str.compareTo( "stop" ) == 0 ) break;
                str = str + "\r\n" ;
                fw.write( str ) ;
        } while ( str.compareTo( "stop" ) != 0 ) ;
        fw.close( ) ;
      System.exit(0);
   } // main( )
} // class
```

Remaining slides not part of evaluations

Client/server socket interaction: TCP



Server (running on **hostid**)
Client
create socket, port=**x**, for incoming request:
welcomeSocket = ServerSocket()
wait for incoming connection request
connectionSocket = welcomeSocket.accept()
TCP connection setup
create socket, connect to **hostid**, port=**x**
clientSocket = Socket()
send request using clientSocket
read request from connectionSocket
write reply to connectionSocket
read reply from clientSocket
close connectionSocket
close clientSocket

TCPClient.java

```
import java.io.*;
import java.net.*;

class TCPClient {
    public static void main(String argv[]) throws Exception
    {
          String sentence;
          String modifiedSentence;

         BufferedReader inFromUser =
             new BufferedReader(new InputStreamReader(System.in));

         Socket clientSocket = new Socket("hostname", 6789);

         DataOutputStream outToServer =
                   new DataOutputStream(clientSocket.getOutputStream());
```

TCPClient.java

```
        BufferedReader inFromServer =
          new BufferedReader(new
    InputStreamReader(clientSocket.getInputStream()));

    sentence = inFromUser.readLine();

    outToServer.writeBytes(sentence + '\n');

    modifiedSentence = inFromServer.readLine();

    System.out.println("FROM SERVER: " + modifiedSentence);

    clientSocket.close();

    }
}
```

TCPServer.java

```
import java.io.*;
import java.net.*;
class TCPServer {
    public static void main(String argv[]) throws Exception
        {
           String clientSentence;
           String capitalizedSentence;

           ServerSocket welcomeSocket = new ServerSocket(6789);

           while(true) {

              Socket connectionSocket = welcomeSocket.accept();

             BufferedReader inFromClient = new BufferedReader(new
                  InputStreamReader(connectionSocket.getInputStream()));
```

TCPServer.java

```
        DataOutputStream  outToClient =
            new DataOutputStream(connectionSocket.getOutputStream());

        clientSentence = inFromClient.readLine();

        capitalizedSentence = clientSentence.toUpperCase() + '\n';

        outToClient.writeBytes(capitalizedSentence);

     }
   }
}
```