COSC 121: Computer Programming II

Today's Key Concepts

- A stream is a sequence of bytes representing data that flow from a source to a destination
  - Input streams for reading data
    - Ex: System.in, Scanner, BufferedReader (+ InputStreamReader or FileReader)
  - Output streams for writing data
    - Ex: System.out, System.err, PrintWriter
- Processing text data
  - Exception handling related to files
  - Scraping from web!

Text Input/Output Applications

- Previously, interaction was limited to the console. What if we can read from text files or write output as text files?
- Example apps:
  - Text transformer:
  - Data processing:
  - Comparing two files:
  - Activity logging:

```
java Uppercase < input.txt > output.txt
```

```
java CsvStats grades.csv
```

```
Line 12 differs:
- old text
+ new text
```

```
> hello
Hi! How can I help?
```

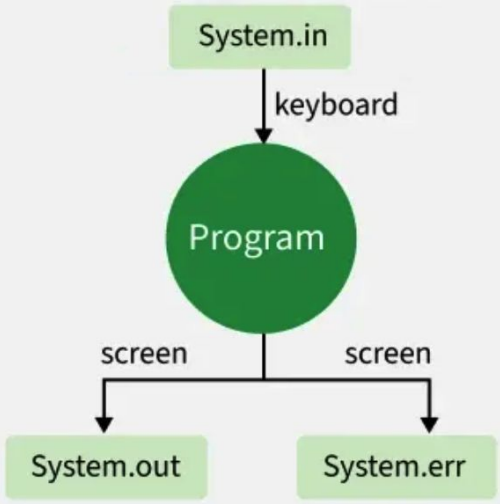
Working with Files

- Files may be of various types (note: file extensions)
- Two main operations on files:
  - Read: when the file is used as input
  - Write: when the file is used as output
- Reading involves:
  - Open file, get info from file, close file
- Writing involves:
  - May involve reading operations
  - Open file, add info to file, close file
  - Create new file, add info to file, close file

Input and Output Streams

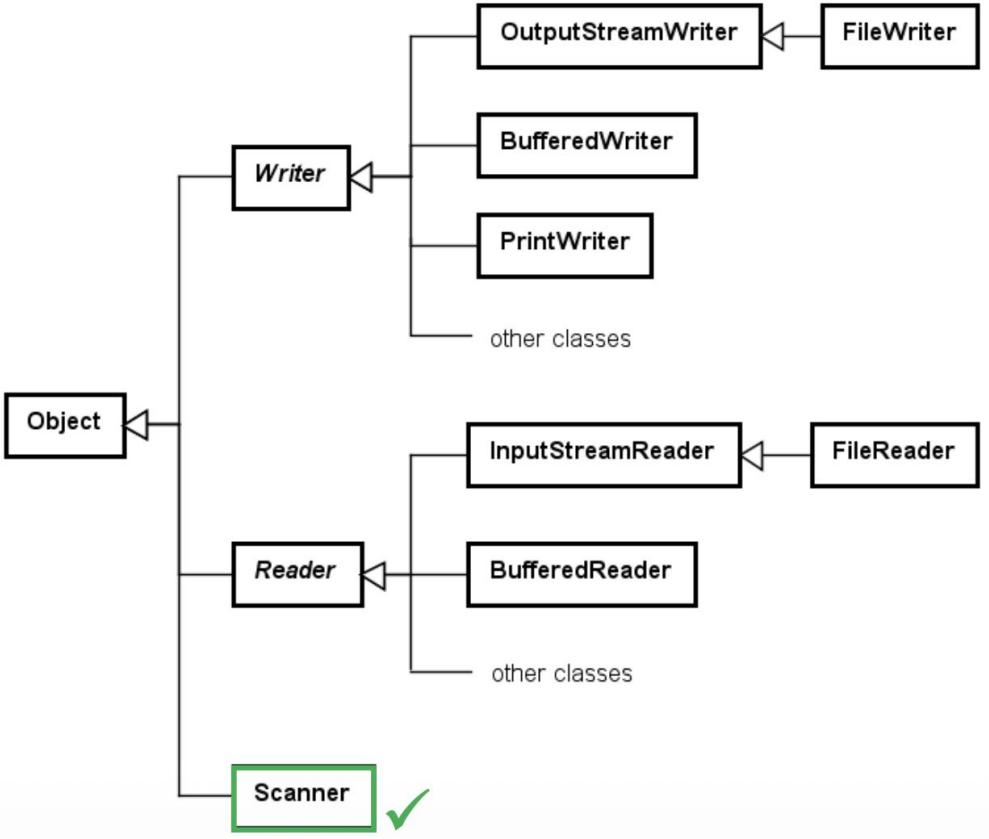
- A stream is a sequence of bytes representing data that flow from a source to a destination
  - Example sources and destinations:
    keyboard, screen, data files, networked computers
- In a program:
  - Read info from an input stream
  - Write info to an output stream
- A program can manage multiple streams simultaneously
  - E.g., read from multiple files and process data across all the files
  - E.g., write to both the screen and save contents to a file

Standard I/O



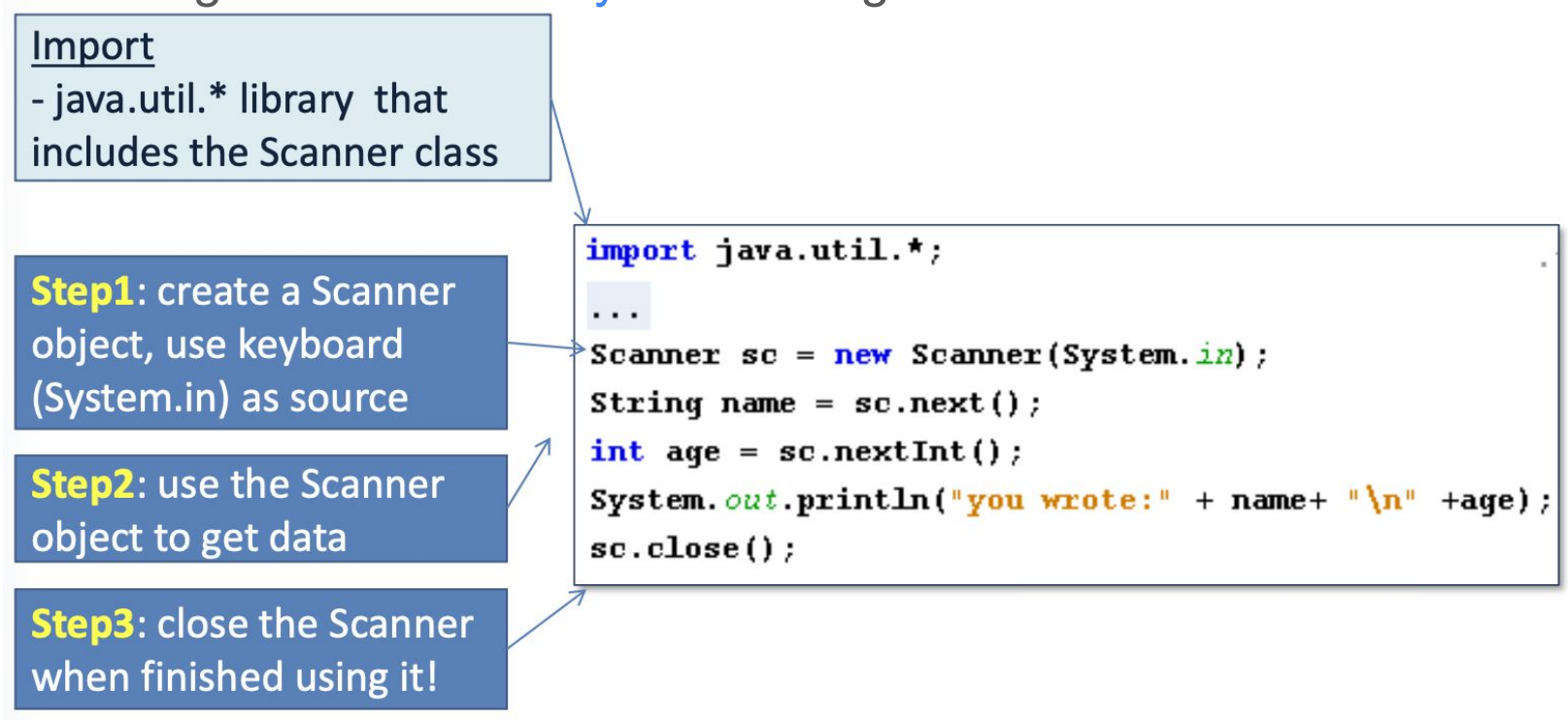
- There are three (default) standard I/O streams:
  - Standard output: defined by System.out
  - Standard input: defined by System.in
  - Standard error: defined by System.err
- We use System.out when we execute println() statements
- System.out and System.err typically represent the console window
  - These are separated so programs can distinguish normal vs. error output, and allow users/other programs to handle them differently
  - E.g., Errors are often logged and ignored
- System.in typically represents keyboard input, which we've used many times with Scanner
  - E.g., Scanner sysin = new Scanner(System.in);

Text I/O Classes



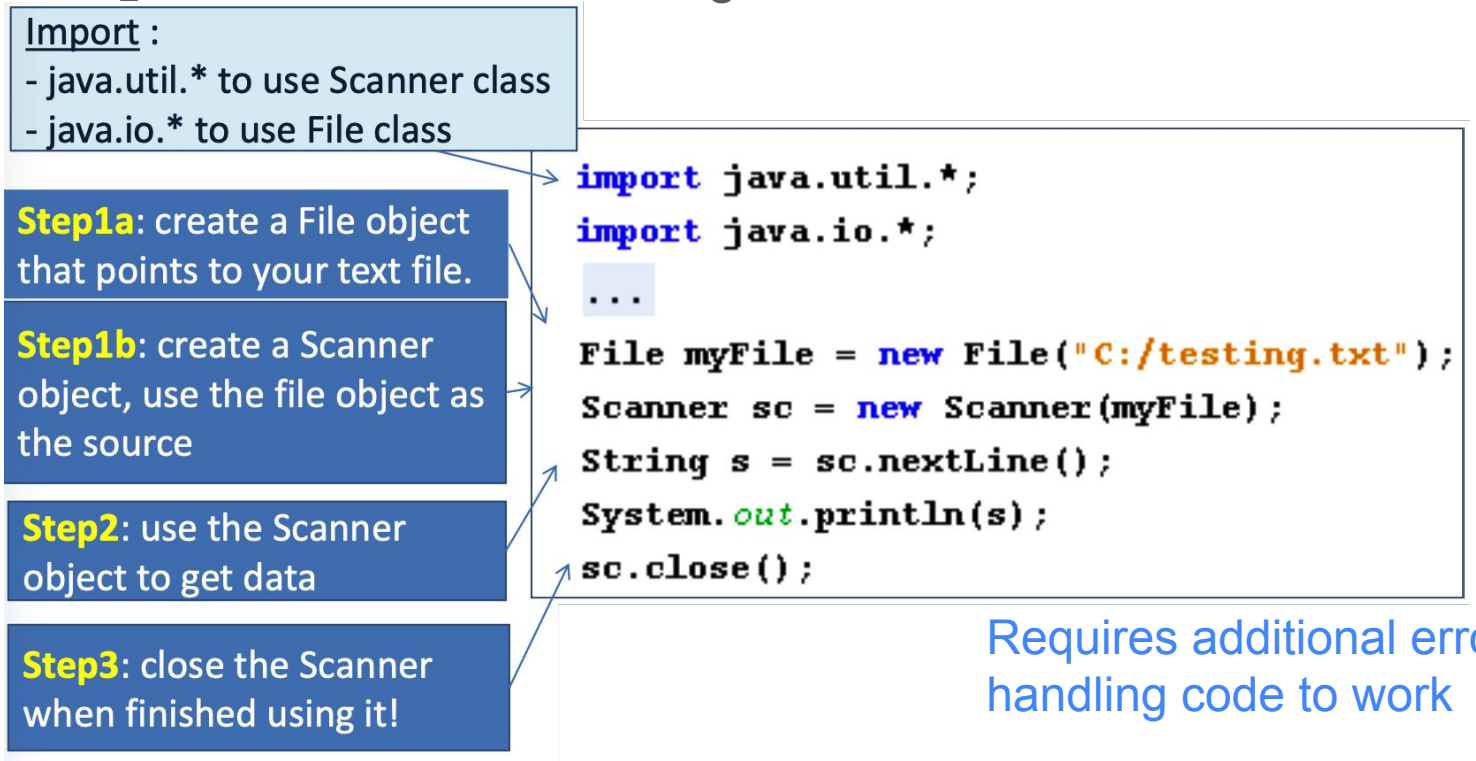
The Scanner Class

- Reading text from the keyboard using the Scanner class



Using Scanner Class with File Class

- Reading text from a text file using the Scanner class



Requires additional error handling code to work

Common Scanner Input Methods

- next() returns the next "token"
- nextLine() returns entire input line as a String
- nextInt() returns next value as an integer
- nextDouble() returns next value as a double
- nextXYZ() returns value of type "XYZ" (variable to represent one of float, short, boolean, byte, BigInteger, BigDecimal)
- useDelimiter(pattern: String)
  sets the scanner object's delimiter to use the given pattern
  - delimiting pattern is applied when next() is used to identify tokens
  - default limiter is space " "

Common Scanner Test Methods

- hasNext() returns true if another token is available to be read
- hasNextLine() returns true if another line is available to be read
- hasNextInt() returns true if another int is available to be read
- hasNextDouble() returns true if another double is available to be read
- hasNextXYZ() returns true if another token of type XYZ is available to be read

Scanner Example

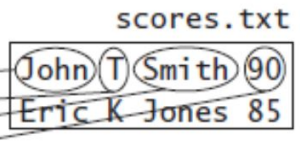
```
import java.util.Scanner;

public class ReadData {
  public static void main(String[] args) throws Exception {
    // Create a File instance
    java.io.File file = new java.io.File("scores.txt");

    // Create a Scanner for the file
    Scanner input = new Scanner(file);

    // Read data from a file
    while (input.hasNext()) {
      String firstName = input.next();
      String mi = input.next();
      String lastName = input.next();
      int score = input.nextInt();
      System.out.println(
        firstName + " " + mi + " " + lastName + " " + score);
    }

    // Close the file
    input.close();
  }
}
```



iClicker Question

Which Scanner method is appropriate to read a whole line of text from a file?

A. next()
B. nextLine()
C. nextByte()
D. nextInt()
E. nextDouble()

Possible Parsing Errors

- Not having the right format in the text file
- Why doesn't this example work?

```
File source = new File("test1.txt");
Scanner in = new Scanner(source);
while(in.hasNext()) {
    System.out.println(in.next());
    System.out.println(in.nextInt());// Exception
    System.out.println(in.nextInt());
}
```

test1.txt

```
John 23 10
Lili 26 15
Mark 12 1
Bill
```

Possible Parsing Errors

- Not having the right format in the text file
- Why doesn't this example work?

```
File source = new File("test1.txt");
Scanner in = new Scanner(source);
while(in.hasNextLine()) {
    System.out.println(in.next()); // Exception
    System.out.println(in.nextInt());
    System.out.println(in.nextInt());
}
```

test2.txt

```
John 23 10
Lili 26 15
Mark 12 1

```

there is a newline here!

Solutions

- 1. Make sure you have the right file format
  - Not good enough

    You need to protect your program from crashing
- 2. Use defensive programming
- 3. Use exception handling

Defensive Programming Solution

- From:

```
File source = new File("test1.txt");
Scanner in = new Scanner(source);
while(in.hasNextLine()) {
    System.out.println(in.next()); // Exception
    System.out.println(in.nextInt());
    System.out.println(in.nextInt());
}
```

- To:

```
File source = new File("test1.txt");
Scanner in = new Scanner(source);
while(in.hasNextLine()) {
    if(in.hasNext())
        System.out.println(in.next());
    if(in.hasNextInt())
        System.out.println(in.nextInt());
    if(in.hasNextInt())
        System.out.println(in.nextInt());
}
```

Exception Handling Solution

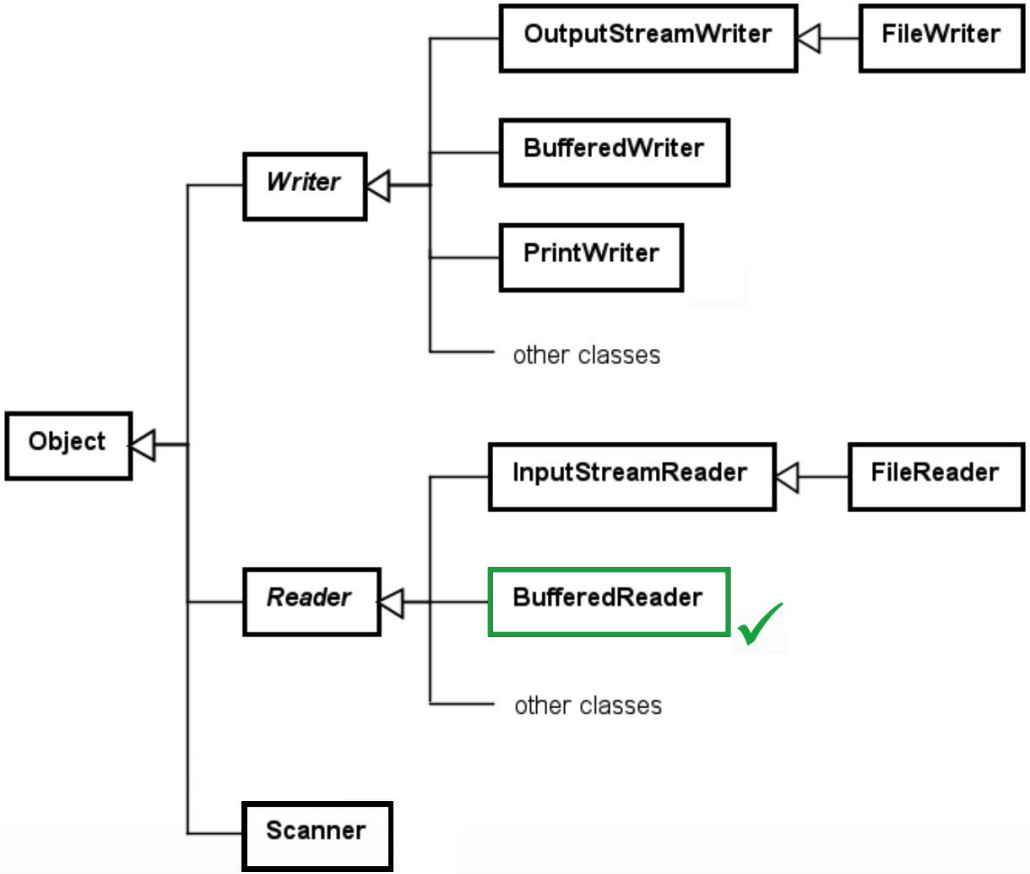
- From:

```
File source = new File("test1.txt");
Scanner in = new Scanner(source);
while(in.hasNextLine()) {
    System.out.println(in.next()); // Exception
    System.out.println(in.nextInt());
    System.out.println(in.nextInt());
}
```

- To:

```
File source = new File("test1.txt");
Scanner in = new Scanner(source);
try{
  while(in.hasNextLine()) {
      System.out.println(in.next());
      System.out.println(in.nextInt());
      System.out.println(in.nextInt());
  }
} catch(NoSuchElementException e) {...}
```

Recall: Text I/O Classes



Alternative: Using BufferedReader

- The BufferedReader wraps another reader class and improves performance
  - InputStreamReader to get input from keyboard as bytes and converts it to characters
  - FileReader to get input from character files
- The BufferedReader as a wrapper
  - Provides a readLine method to read a line of text
  - Improves efficiency (faster than Scanner)
- Java uses similar “writer” classes for writing data
  - OutputStreamWriter, FileWriter, BufferedWriter

BufferedReader with InputStreamReader

- Example: reading input from keyboard

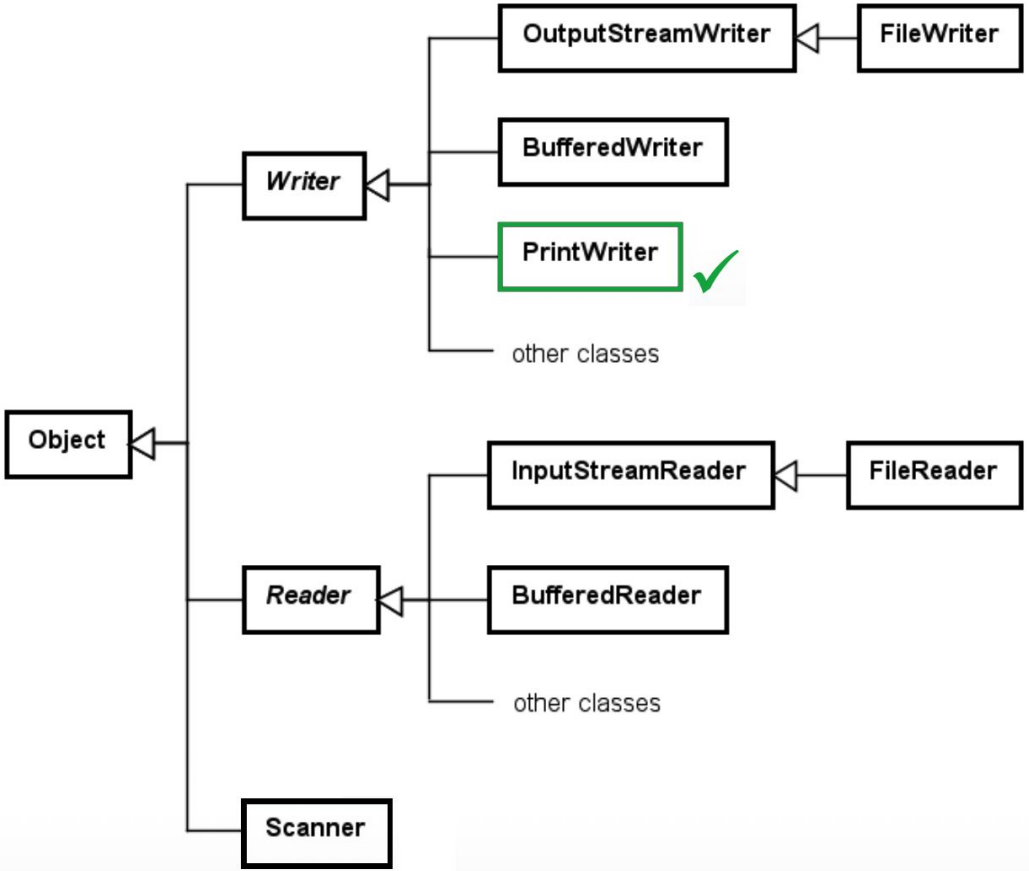
```
import java.io.*;
public class ex2 {
    public static void main(String[] args) throws IOException{
        InputStreamReader isr = new InputStreamReader(System.in);
        BufferedReader in = new BufferedReader(isr);
        System.out.print("What is your name? ");
        String name = in.readLine();
        System.out.println("Welcome " + name);
        in.close();
    }
}
```

BufferedReader with FileReader

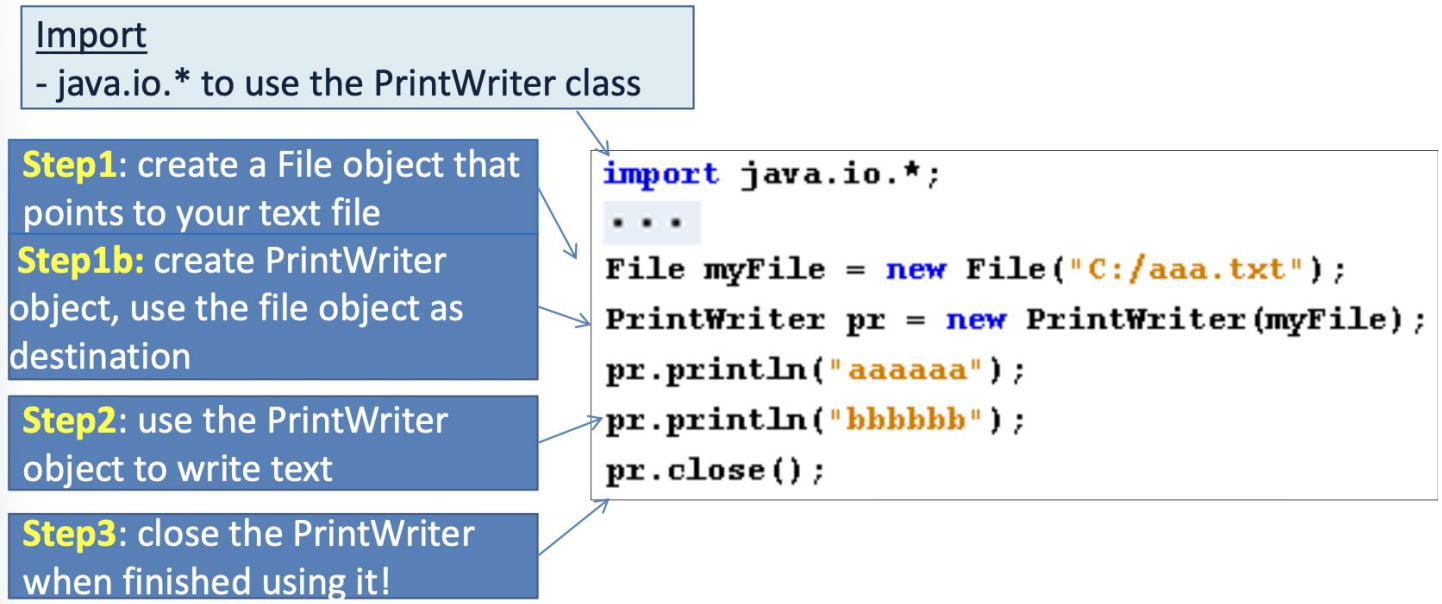
- Example: reading input from file

```
import java.io.*;
public class ex2 {
    public static void main(String[] args) throws IOException{
        FileReader fr = new FileReader("c://abc.txt");
        BufferedReader in = new BufferedReader(fr);
        String s;
        while((s = in.readLine()) != null)
            System.out.println(s);
        in.close();
    }
}
```

Recall: Text I/O Classes



Using PrintWriter Class to Write Text to File

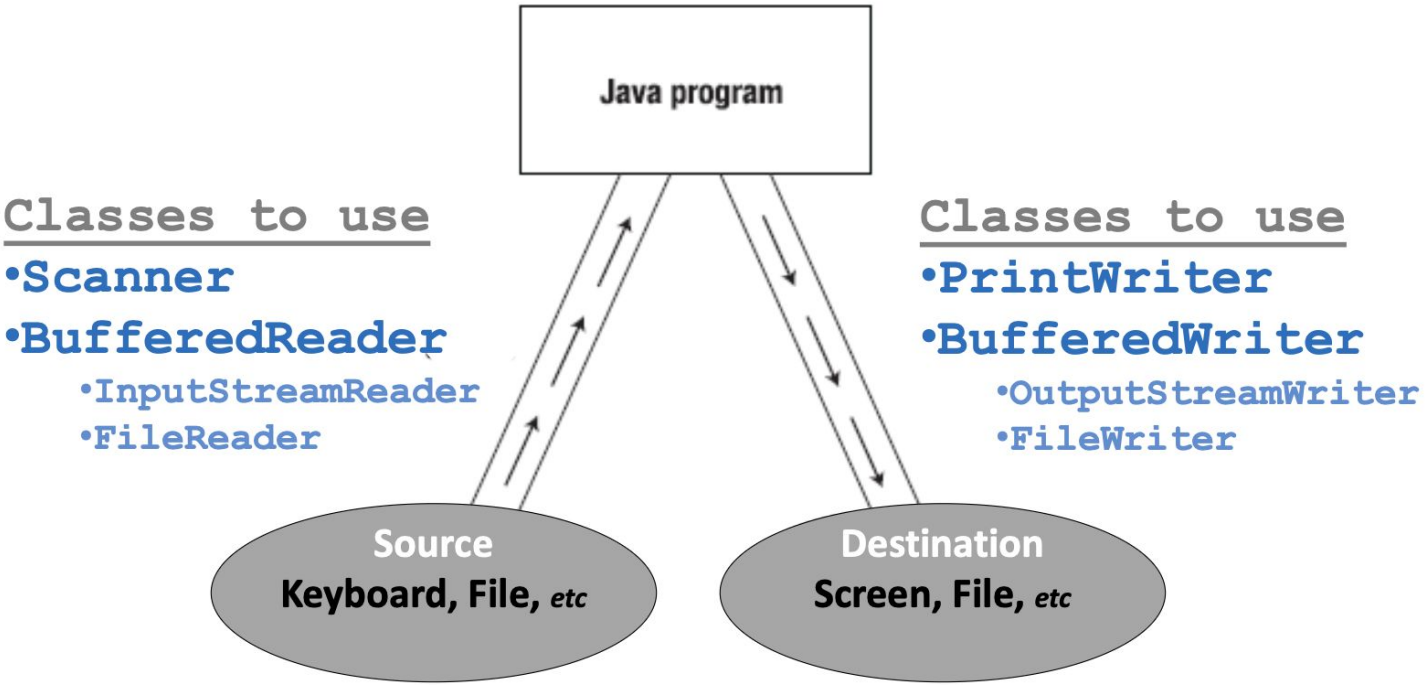


- You can use print(), println(), and printf() methods
  - Data in aaa.txt will be overwritten
- In the case the file cannot be created, error handling code is needed

PrintWriter Examples

- Print into a file:
  ```
  PrintWriter out = new PrintWriter( new File( "test.txt" ));
  out.println( "ABC" );
  out.close();
  ```
- Print into a file (same as above):
  ```
  PrintWriter out = new PrintWriter( "test.txt" );
  out.println( "ABC" );
  out.close();
  ```
- Print to screen (standard output):
  ```
  PrintWriter out = new PrintWriter( System.out );
  out.println( "ABC" );
  out.close();
  ```

Summary


Java program
Classes to use
•Scanner
•BufferedReader
•InputStreamReader
•FileReader
Classes to use
•PrintWriter
•BufferedWriter
•OutputStreamWriter
•FileWriter
Source
Keyboard, File, etc
Destination
Screen, File, etc

For Reading

READING	From Keyboard	From File

Scanner	`Scanner in = new Scanner(X);`
	`myString = in.next(); myInt = in.nextInt(); //etc`
	`in.close();`	

Replace X with:	`System.in`	`new File("C:/filename")`

BufferedReader	`BufferedReader in = new BufferedReader(X);`
	`myString = in.readLine();`
	`in.close();`	

Replace X with:	`new InputStreamReader(System.in)`	`new FileReader("C:/filename")`

For Writing

WRITING	To Screen	To File

PrintWriter	`PrintWriter out = new PrintWriter(X);`
	`out.println("some text here");`
	`out.close();`	

Replace X with:	`System.out`	`new File("C:/filename")`

BufferedWriter	`BufferedWriter out = new BufferedWriter(X);`
	`out.write("some text here");`
	`out.close();`	

Replace X with:	`new outputStreamWriter(System.out)`	`new FileWriter("C:/filename")`
	OR `System.out.println("text");`	

iClicker Question

What are the contents of temp.txt after this code:

```
PrintWriter out = new PrintWriter("temp.txt");
out.printf("x:%3.3f ", 32.32);
out.printf("%6b ", (1>2));
out.printf("%6s ", "Java");
out.close();
```

A. x:32.320 false Java
B. x:32.32 false Java
C. x:32.320 1>2 Java
D. x:32 1>2 Java
E. Something else

The File Class

- Recall: `File source = new File( "filename.txt" );`
- Provides an abstraction that deals with most of the machine-dependent complexities of files and path names
- The File class is a wrapper class for the file name and its directory path
- Some useful File methods:
  - exists(): boolean
    Returns true if the file or the directory exists.
  - canRead(): boolean
    Returns true if the file exists and can be read.
  - isDirectory(): boolean
    Returns true if the File object represents a directory.
  - isFile(): boolean
    Returns true if the File object represents a file

Try with Resources

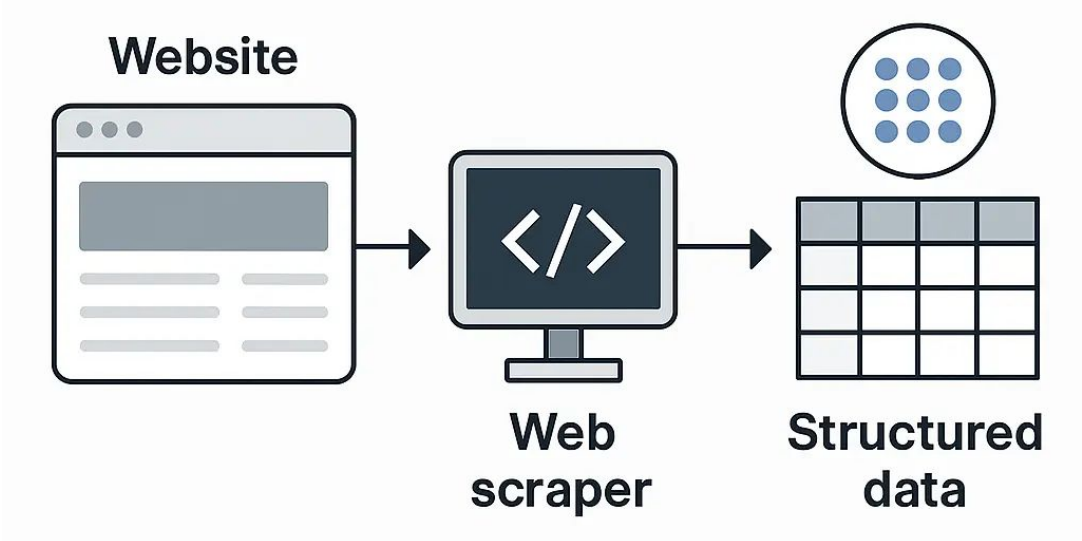
- JDK 7 provides try-with-resources syntax that automatically closes files because programmers often forgot to close files
- Example:

```
import java.io.*;
public class WriteDataWithAutoClose {
    public static void main(String[] args) throws Exception {
        File file = new File("scores.txt");
        if (file.exists()) {
            System.out.println("File already exists");
            System.exit(0);
        }
        try (PrintWriter output = new PrintWriter(file);) {
            output.print("John T Smith ");
            output.println(90);
            output.print("Eric K Jones ");
            output.println(85);
        }
    }
}
```

declare and create resources in "try"

use resource as usual

Reading from the Web



How to write such programs?

General Idea: URL File Object

```
import java.net.URL;
import java.io.BufferedReader;
import java.io.InputStreamReader;

public class ReadWeb {
    public static void main(String[] args) throws Exception {
        URL url = new URL("https://cmps-people.ok.ubc.ca/bowenhui/121");

        BufferedReader in = new BufferedReader(
            new InputStreamReader(url.openStream())
        );

        String line;
        while ((line = in.readLine()) != null) {
            System.out.println(line);
        }

        in.close();
    }
}
```

// add imports

// create URL object

// open data stream

// read each line

deprecated!

Suppressing Warnings

can add warning

```
import java.net.URL;
import java.io.BufferedReader;
import java.io.InputStreamReader;

public class ReadWeb {
    @SuppressWarnings("deprecation")
    public static void main(String[] args) throws Exception {
        URL url = new URL("https://cmps-people.ok.ubc.ca/bowenhui/121");

        BufferedReader in = new BufferedReader(
            new InputStreamReader(url.openStream())
        );

        String line;
        while ((line = in.readLine()) != null) {
            System.out.println(line);
        }

        in.close();
    }
}
```

Modern Approach: Use URI then Convert to URL Object

```java
import java.net.URI;
import java.net.URL;
import java.io.BufferedReader;
import java.io.InputStreamReader;

public class ReadWeb {
    public static void main(String[] args) throws Exception {
        // previously: URL url = URL( "webaddress" );
        URI uri = URI.create("https://cmps-people.ok.ubc.ca/bowenhui/121");
        URL url = uri.toURL();
        BufferedReader in = new BufferedReader(
            new InputStreamReader(url.openStream())
        );

        String line;
        while ((line = in.readLine()) != null) {
            System.out.println(line);
        }

        in.close();
    }
}
```

// new import

// recommended

Adding Exception Handling

```
import java.net.URI;
import java.net.URL;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

public class ReadWeb {
    public static void main(String[] args) throws Exception {
        try {
            // previously: URL url = URL( "webaddress" );
            URI uri = URI.create("https://cmps-people.ok.ubc.ca/bowenhui/121");
            URL url = uri.toURL();
            BufferedReader in = new BufferedReader(
                new InputStreamReader(url.openStream())
            );
        } catch (IOException e) {
            System.err.println("Failed to read from web");
        }

        String line;
        while ((line = in.readLine()) != null) {
            System.out.println(line);
        }

        in.close();
    }
}
```

why a compilation error?

With Proper Scoping


```
import java.net.URI;
import java.net.URL;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

public class ReadWeb {
    public static void main(String[] args) throws Exception {
        try {
            // previously: URL url = URL( "webaddress" );
            URI uri = URI.create("https://cmps-people.ok.ubc.ca/bowenhui/121");
            URL url = uri.toURL();
            BufferedReader in = new BufferedReader(
                new InputStreamReader(url.openStream())
            );
            String line;
            while ((line = in.readLine()) != null) {
                System.out.println(line);
            }

            in.close();
        } catch (IOException e) {
            System.err.println("Failed to read from web");
        }
    }
}
```