Output Files

PrintStream

- System.out
- Files

Output Files

Input/Output File Example

- Scanner close() method
- PrintStream close() method

Gotcha: Mixing Tokens and Lines

PrintStream Class

- System.out is a PrintStream!
- We generally use it to output to the monitor:

```
System.out.println(“Hello World!”);
System.out.print(“Enter name: ”);
```

PrintStream Class (cont.)

- We can “redirect” System.out to a file.

```
eos% java HelloWorld > output
```

- We can “redirect” both System.in and System.out to files!:

```
eos% java CreditorCardValidator < testInput > output
```

PrintStream Class (cont.)

- We can use the PrintStream class to write to a file:

```
PrintStream output = new PrintStream(new File(“words.txt”));
output.println(“ant”);
output.print(“bug”);
output.println(“cat”);
output.print(“dog”);
output.println(“egg”);
```

Output Files

```
System.out.print("Enter filename for output data: ");
String filename = console.next();
PrintStream output = new PrintStream(new File(filename));
```

- If the file does not exist, a new file will be created.
- If the file *does* exist, it will be overwritten!!!!

```
    //Check with user if file exists!
    File file = new File(filename);
    if (file.exists()) {
        System.out.print("OK to overwrite file? (y/n): ");

    }
```

Output Files (cont.)

- Whether or not the output file exists, we must still handle the FileNotFoundException!
  - throws clause
  - try/catch block

```
import java.io.*;
public class AlmaMater {
   public static void main(String[] args) {

     try {
       File file = new File(“AlmaMater.txt");
       PrintStream output = new PrintStream(file);
       output.println("Where the winds of Dixie softly blow");
       output.println("o'er the fields of Caroline,");
       output.println("There stands ever cherished, N.C. State,");
       output.println("as thy honored shrine.");
     }
     catch (FileNotFoundException e) {
        System.out.println("Problem creating file!");
   }
  }
}
```

Input/Output File Example

```
import java.util.*;
import java.io.*;
/**
 * Copies the input file to the output file
 * converting all spaces to dashes
 * @author Suzanne Balik
 */
public class SpacesToDashes {
```

```
/**
 *  Prompts the user for the name of an  input file and an output file.
 *  If the output file does not exist,  it is created and the input file is
 *  copied to it, converting all spaces to dashes.
 *  If the output file does exist or if  there is a problem creating the
 *  output  file, an error message is output and the program exits.
 */
 public static void main (String[] args) {
     // See code on next slide...
```

```
public static void main(String[] args) {
     Scanner console = new Scanner(System.in);
     Scanner in = getInput(console);
     System.out.print("output file name? ");
     String filename = console.next();
     File f = new File(filename);
     PrintStream out = null;
     if (!f.exists()) {
        try {
           out = new PrintStream(f);
        }
        catch (FileNotFoundException e) {
            System.out.println (  e.getMessage() );
            System.exit(1);
       }
       while (in.hasNextLine()) {
            String line = in.nextLine();
            String dashLine = convertSpacesToDashes(line);
            out.println(dashLine);
       }
       in.close();       // Close the Scanner
       out.close();     // Close the PrintStream
    }
    else {
       System.out.println ("File already exists!");
    }
  }
```

```
/**
  * Prompts the user for an input file name,
  * then creates and returns a Scanner tied to the file
  * @param console console input scanner
  * @return scanner for input file
  */
public static Scanner getInput(Scanner console) {
    Scanner result = null; //null signifies NO object reference
    while (result == null) {
        System.out.print("input file name? ");
        String name = console.next();
        try {
          result = new Scanner(new File(name));
        }
        catch (FileNotFoundException e) {
            System.out.println("File not found. Please try again.");
         }
     }
     return result;
}
```

```
 /**
  *  Creates a copy of a String in which each space has been
  *  replaced with a dash (“-”)
  *  Illustrates character by character processing of a line –
  *  could be done with String replace() instead
  *  @param line string to be copied
  *  @return copy of string with spaces replaced by dashes
  */
 public static String convertSpacesToDashes(String line) {
     String s = "";
     for (int i = 0; i < line.length(); i++) {
        char ch = line.charAt(i);
        if (ch == ‘ ’) {
           s += ‘-’;
        }
        else {
           s += ch;
      }
      return s;
  }
}
```

Mixing tokens and lines

- Using `nextLine` in conjunction with the token-based methods on the same `Scanner` can cause bad results.

```
23   3.14
Joe    "Hello" world
         45.2  19
```

  - You'd think you could read `23` and `3.14` with `nextInt` and `nextDouble`, then read `Joe  "Hello"  world` with `nextLine` .

```
System.out.println(input.nextInt());       // 23
System.out.println(input.nextDouble());    // 3.14
System.out.println(input.nextLine());      //
```

  - But the `nextLine` call produces no output! Why?

Mixing lines and tokens

- Don't read both tokens and lines from the same `Scanner`:

```
23   3.14
Joe    "Hello world"
              45.2   19



input.nextInt()                                   // 23
23\t3.14\nJoe\t"Hello" world\n\t\t45.2  19\n
  ^

input.nextDouble()                                // 3.14
23\t3.14\nJoe\t"Hello" world\n\t\t45.2  19\n
        ^

input.nextLine()                                  // "" (empty!)
23\t3.14\nJoe\t"Hello" world\n\t\t45.2  19\n
          ^

input.nextLine()                         // "Joe\t\"Hello\" world"
23\t3.14\nJoe\t"Hello" world\n\t\t45.2  19\n
                              ^
```

Line-and-token example

```
Scanner console = new Scanner(System.in);
System.out.print("Enter your age: ");
int age = console.nextInt();

System.out.print("Now enter your name: ");
String name = console.nextLine();
System.out.println(name + " is " + age + " years old.");
```

Log of execution (user input underlined):

```
Enter your age: 12
Now enter your name: Sideshow Bob
 is 12 years old.
```

- Why?
  - Overall input: `12\nSideshow Bob`
  - After `nextInt()`:
    ```
    12\nSideshow Bob
      ^
    ```
  - After `nextLine()`:
    ```
    12\nSideshow Bob
        ^
    ```

Line-and-token example

```
Scanner console = new Scanner(System.in);
System.out.print("Enter your age: ");
int age = console.nextInt();

//Add an extra console.nextLine() to read blank line
console.nextLine();


System.out.print("Now enter your name: ");
String name = console.nextLine();
System.out.println(name + " is " + age + " years old.");
```

Log of execution (user input underlined):

```
Enter your age: 12
Now enter your name: Sideshow Bob
Sideshow Bob is 12 years old.
```

^