CS1101 Programming and Problem Solving

Dr. Gina Bai

Spring 2023

Logistics

- **ZY-7A** on zyBook > Assignments
  - Due: **Wednesday, March 29**, at 11:59pm
- **PA09 –A, B** on zyBook > Chap 11
  - Due: **Thursday, March 30**, at 11:59pm

File Output

zyBook Chap 6.5

Recap – How do we read a file?

- **Step 1:** Specify the **file path** as a **String** object

```
String fileName = "data.txt";
```

- **Step 2:** Construct a **File** object to get the information about a file on the disk

```
import java.io.File;
File inputFile = new File(fileName);
```

- **Step 3:** Construct a **Scanner** object to read the file

```
import java.util.Scanner;
Scanner scnr = new Scanner(inputFile);
```

How do we write to a file?

- **Step 1:** Specify the **file path** as a **String** object

```java
String fileName = "output.txt";
```

- **Step 2:** Construct a **File** object to get the information about a file on the disk

```java
import java.io.File;
File outputFile = new File(fileName);
```

- **Step 3:** Construct a **PrintStream** object to print to the file

```java
import java.io.PrintStream;
PrintStream out = new PrintStream(outputFile);
```

```java
PrintStream <name> = new PrintStream(new File("<fileName>"));
```

PrintStream Objects

- The console output object, **`System.out`**, is a **`PrintStream`** object

```
// Prints the message to the console, equivalent to System.out.println()
PrintStream consoleOutput = System.out;
consoleOutput.println("Message in console!");

// Prints the message to the file result.txt
PrintStream fileOutput = new PrintStream(new File("result.txt"));
fileOutput.println("Message in file!");
```

- Any methods we have used with **`System.out`** (e.g., **`print`**, **`println`**, **`printf`**) also work with **`PrintStream`** objects

Write to a File

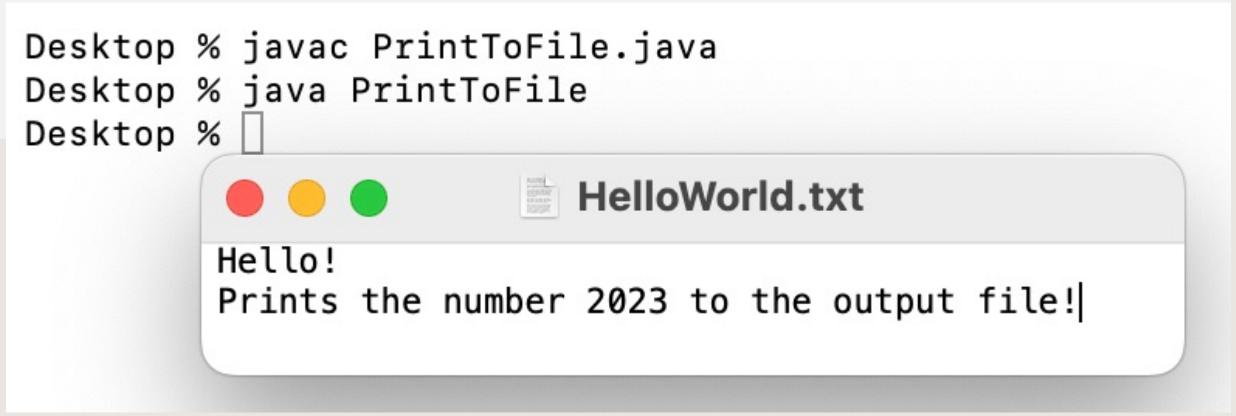
```
PrintStream <name> = new PrintStream(new File("<fileName>"));
```

- If the given file **does not exist**, it is **created**

```java
import java.io.File;
import java.io.PrintStream;

public class PrintToFile {
    public static void main (String[] args) {
        // Specify the file that we would like to print to
        File outFile = new File("/Users/ginabai/Desktop/HelloWorld.txt");

        // Construct the PrintStream object to print to the specified output file
        PrintStream output = new PrintStream(outFile);
        output.println("Hello!");
        output.printf("Prints the number %d ", 2023);
        output.print("to the output file!");
        output.close();
    }
}
```

```
$ javac PrintToFile.java
PrintToFile.java:10: error: unreported exception FileNotFoundException;
must be caught or declared to be thrown
        PrintStream output = new PrintStream(outFile);
                             ^
1 error
```

```java
import java.io.File;
import java.io.PrintStream;
import java.io.FileNotFoundException;

public class PrintToFile {
    public static void main (String[] args) throws FileNotFoundException {
        // Specify the file that we would like to print to
        File outFile = new File("/Users/ginabai/Desktop/HelloWorld.txt");

        // Construct the PrintStream object to print to the specified output file
        PrintStream output = new PrintStream(outFile);
        output.println("Hello!");
        output.printf("Prints the number %d ", 2023);
        output.print("to the output file!");
        output.close();
    }
}
```



```
Desktop % javac PrintToFile.java
Desktop % java PrintToFile
Desktop %
```

HelloWorld.txt

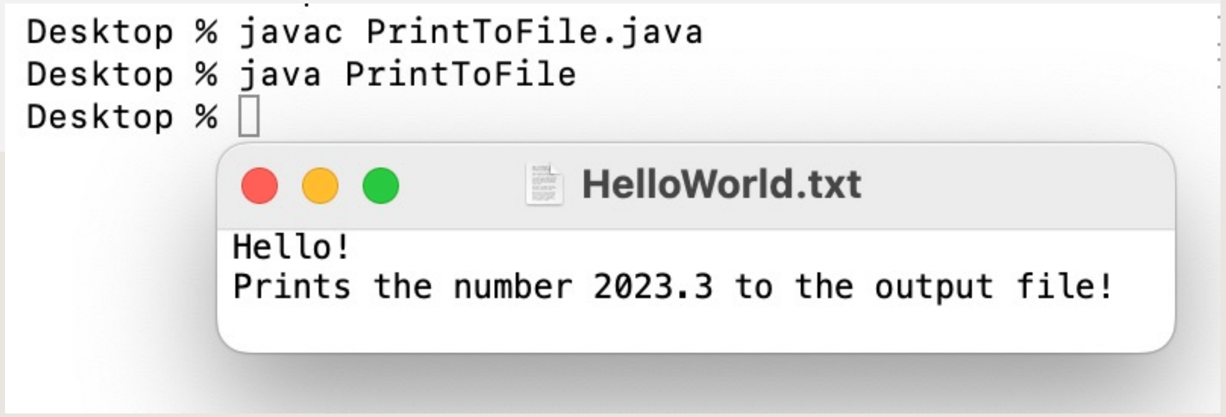
```
Hello!
Prints the number 2023 to the output file!
```

Write to a File

```
PrintStream <name> = new PrintStream(new File("<fileName>"));
```

- If the given file **does not exist**, it is **created**
- If the given file **already exists**, it is **overwritten**

```java
import java.io.File;
import java.io.PrintStream;
import java.io.FileNotFoundException;

public class PrintToFile {
    public static void main (String[] args) throws FileNotFoundException {
        // Specify the file that we would like to print to
        File outFile = new File("/Users/ginabai/Desktop/HelloWorld.txt");

        // Construct the PrintStream object to print to the specified output file
        PrintStream output = new PrintStream(outFile);
        output.println("Hello!");
        output.printf("Prints the number %.1f ", 2023.327);
        output.print("to the output file!");
        output.close();
    }
}
```



```
Desktop % javac PrintToFile.java
Desktop % java PrintToFile
Desktop %
```

```
Hello!
Prints the number 2023.3 to the output file!
```

Write to a File

```
PrintStream <name> = new PrintStream(new File("<fileName>"));
```

- If the given file **does not exist**, it is **created**
- If the given file **already exists**, it is **overwritten**
  - How to avoid overwriting a file?
    - <outputFile>.exists()
      - Common options for an output file that already exists: overwrite the file, stop the program, prompt for a new output file name, prompt user whether they would like to overwrite

```java
import java.io.File;
import java.io.PrintStream;
import java.io.FileNotFoundException;

public class PrintToFile {
    public static void main (String[] args) throws FileNotFoundException {
        // Specify the file that we would like to print to
        File outFile = new File("/Users/ginabai/Desktop/HelloWorld.txt");

        if(outFile.exists()){
            System.out.println("Output file exists. It was not overwritten.");
            System.exit(1); // Terminate the program execution
        }

        // Construct the PrintStream object to print to the specified output file
        PrintStream output = new PrintStream(outFile);
        output.println("Hello!");
        output.printf("Prints the number %.1f ", 2023.327);
        output.print("to the output file!");
        output.close();
    }
}
```

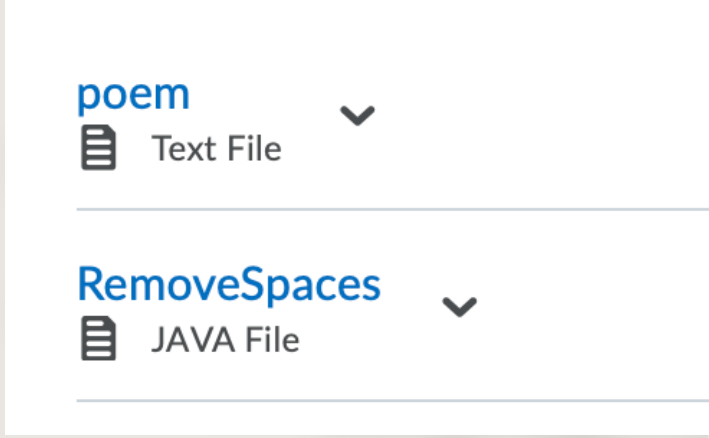
```
$ javac PrintToFile.java
$ java PrintToFile
Output file exists. It was not overwritten.
```

Write to a File

```
PrintStream <name> = new PrintStream(new File("<fileName>"));
```

- If the given file **does not exist**, it is **created**
- If the given file **already exists**, it is **overwritten**
- **Do not open a file** for **reading** (i.e., Scanner) and **writing** (i.e., PrintStream) **at the same time**
  - You could overwrite your input file by accident! The result can be an empty file (size 0 bytes).

Coding Practice – File I/O


poem
Text File
RemoveSpaces
JAVA File

- Write a program called **RemoveSpaces** that
  - Prompt the user for an input file name
  - If the input file does not exist, re-prompt the user for an input file name
  - Otherwise,
    - If the output file does not exist, it **copies each line of the input file to the output file with no whitespace before or after each token**.
    - If the output file does exist, it outputs "<fileName> already exists!" and **does not overwrite** the file.

poem.txt

```
Still I Rise

Maya Angelou

You may write me down in history
With your bitter, twisted lies,
You may trod me in the very dirt
But still, like dust, I'll rise.
```

poemNoSpace.txt

```
StillIRise

MayaAngelou

Youmaywritemedowninhistory
Withyourbitter,twistedlies,
Youmaytrodmeintheverydirt
Butstill,likedust,I'llrise.
```

Sample Solution

```java
import java.util.Scanner;
import java.io.*;

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

        System.out.print("Enter file name: ");
        String fileName = console.nextLine().trim();

        File inputFile = new File(fileName);

        // Use a while loop to validte the input file name
        while(!inputFile.exists()){
            System.out.print("Input file does not exist, try again: ");
            fileName = console.nextLine().trim();
            inputFile = new File(fileName);
        }
        // Construct a Scanner to read the input file
        Scanner input = new Scanner(inputFile);

        // Specify the output file name
        File outputFile = new File("poemNoSpace.txt");

        // Use an if statement to check whether the output file already exists
        if (outputFile.exists()) {
            System.out.println("poemNoSpace.txt already exists!");
            System.exit(1);
        }
        // Construct a PrintStream to print to the output file
        PrintStream output = new PrintStream(outputFile);

        // Line-based processing
        while (input.hasNextLine()) {
            String line = input.nextLine();
            Scanner lineScnr = new Scanner(line);
            while (lineScnr.hasNext()) {
                // remove the spaces by reading and printing the tokens
                output.print(lineScnr.next());
            }
            output.println();
            lineScnr.close(); // Close the Scanner for the line
        }

        input.close();  // Close the Scanner for the input file
        output.close(); // Close the PrintStream for the output file
    }
}
```

Array Basics

zyBook Chap 7.1, 7.2, 7.3, 7.4

Recap – Array Basics

- An array object is an **indexed collection of data** of the **same type**
  - `<type>[] <arrayName> = new <type>[<arraySize>];`
- **Zero-based** indexing
- Accessing an array element with **`<arrayName>[<index>]`**
- Getting the length of an array with **`<arrayName>.length`**
- Initialize via two ways,
  - a for loop, or
  - `<type>[] <arrayName> = { val1, val2, …, valN };`

Recap

- Q: (T/F) Each data element in an array can be accessed by an index.

  True, via <arrayName>[<index>]

- Q: (T/F) A single array can store data elements of different data classes (int, double, char, etc.)

  False, an array is a collection of data of the same type

- Q: (T/F) All elements of the array `int[] arr = new int[5]` are initialized to 0.

  True, auto-initialization

- Q: Given the definition `int[] arr = {1 ,2, 3, 4, 5};`, arr [3] = __________?

  4, array index starts at 0

Recap

Q: What is the resulting array?

```
int[] arr = new int[4];
for (int i = 0; i < 4; i++) {
    if ( i % 2 == 0) {
        arr[i] = 2 * i;
    } else{
        arr[i] = i;
    }
}
```

[0, 1, 4, 3]

Q: Fill in the blanks.

```java
public class Gradebook {
    public static void main (String[] args) {
        double[] exam1 = {90.0, 87.5, 95.5, 88.0, 79.5, 91.5};

        System.out.println("First element: "  + exam1[__________]);
        System.out.println("Middle element: " + exam1[__________]);
        System.out.println("Last element: "   + exam1[__________]);
    }
}
```

```
$ javac Gradebook.java
$ java Gradebook
First element: 90.0
Middle element: 95.5
Last element: 91.5
```

Q: Fill in the blanks.

```
public class Gradebook {
    public static void main (String[] args) {
        double[] exam1 = {90.0, 87.5, 95.5, 88.0, 79.5, 91.5};

        System.out.println("First element: "  + exam1[0]);
        System.out.println("Middle element: " + exam1[__________]);
        System.out.println("Last element: "   + exam1[__________]);
    }
}
```

```
$ javac Gradebook.java
$ java Gradebook
First element: 90.0
Middle element: 95.5
Last element: 91.5
```

The middle of an **even length** array is considered **the left of center**

Q: Fill in the blanks.

```java
public class Gradebook {
    public static void main (String[] args) {
        double[] exam1 = {90.0, 87.5, 95.5, 88.0, 79.5, 91.5};

        System.out.println("First element: "  + exam1[0]);
        System.out.println("Middle element: " + exam1[(exam1.length – 1) / 2]);
        System.out.println("Last element: "   + exam1[__________]);
    }
}
```

This works for both **even** and **odd** length arrays

```
$ javac Gradebook.java
$ java Gradebook
First element: 90.0
Middle element: 95.5
Last element: 91.5
```

The middle of an **even length** array is considered **the left of center**

Q: Fill in the blanks.

```java
public class Gradebook {
    public static void main (String[] args) {
        double[] exam1 = {90.0, 87.5, 95.5, 88.0, 79.5, 91.5};

        System.out.println("First element: "  + exam1[0]);
        System.out.println("Middle element: " + exam1[(exam1.length – 1) / 2]);
        System.out.println("Last element: "   + exam1[exam1.length – 1]);
    }
}
```

This works for both **even** and **odd** length arrays

```
$ javac Gradebook.java
$ java Gradebook
First element: 90.0
Middle element: 95.5
Last element: 91.5
```

The middle of an **even length** array is considered **the left of center**

Arrays Class

Arrays Class

- `import java.util.Arrays;`
- Contains methods to manipulate arrays
- Syntax: <ClassName>.<methodName>(<parameter(s)>)
  - `Arrays.<methodName>(arrayName);`
  - Recap on similar ones: Math.sqrt(5), Character.toUpperCase('a')

Print an array

- Approach 1:
  - Print each element with a for loop

- Approach 2:
  - `Arrays.toString(arrayName)`

```java
import java.util.Arrays;

public class ArraysMethodDemo {
    public static void main(String[] args) {
        int[] arr = { 1, 2, 3 };
        System.out.println(Arrays.toString(arr));
    }
}
```

```
$ javac ArraysMethodDemo.java
$ java ArraysMethodDemo
[1, 2, 3]
```

Compare arrays (if two arrays are equal)

- Approach 1:
  - Compare size, if same, compare each element with a for loop

- Approach 2:
  - `Arrays.equals(array1, array2)`

```java
import java.util.Arrays;

public class ArraysMethodDemo {
    public static void main(String[] args) {
        int[] a1 = { 1, 2, 3 };
        int[] a2 = { 1, 2, 3 };
        System.out.println(Arrays.equals(a1, a2));
    }
}
```

```
$ javac ArraysMethodDemo.java
$ java ArraysMethodDemo
true
```

```java
import java.util.Arrays;

public class ArraysMethodDemo {
    public static void main(String[] args) {
        int[] a1 = { 1, 2, 3 };
        int[] a2 = { 1, 2, 3 };
        int[] a3 = a1;

        // == is used to compare referential equality
        System.out.println("a1 == a2 : " + (a1 == a2));   // false
        System.out.println("a1 == a3 : " + (a1 == a3));   // true
        System.out.println("a2 == a3 : " + (a2 == a3));   // false

        // By defualt, Object.equals() compares object memory addresses
        // Hence, it works the same as the == operator for Arrays
        System.out.println("a1.equals(a2) : " + a1.equals(a2));  // false
        System.out.println("a1.equals(a3) : " + a1.equals(a3));  // true
        System.out.println("a2.equals(a3) : " + a2.equals(a3));  // false

        // MUST use Arrays.equals(array1, array2) to compare arrays
        System.out.println("Arrays.equals(a1, a2) : " + Arrays.equals(a1, a2)); //true
        System.out.println("Arrays.equals(a1, a3) : " + Arrays.equals(a1, a3)); //true
        System.out.println("Arrays.equals(a2, a3) : " + Arrays.equals(a2, a3)); //true
    }
}
```

Resize an existing array

- Approach 1:
  - Construct another array, copy the elements from the old array to the new array with a for loop

- Approach 2:
  - `Arrays.copyOf(arrayName, newSize)`

```java
import java.util.Arrays;

public class ArraysMethodDemo {
    public static void main(String []args) {
        int[] arr = { 1, 2, 3 };

        // Expand the size to 5
        arr = Arrays.copyOf(arr, 5);
        System.out.println(Arrays.toString(arr));

        // Shrink the size to 2
        arr = Arrays.copyOf(arr, 2);
        System.out.println(Arrays.toString(arr));

        // Expand the size to 4
        arr = Arrays.copyOf(arr, 4);
        System.out.println(Arrays.toString(arr));
    }
}
```

```
$ javac ArraysMethodDemo.java
$ java ArraysMethodDemo
[1, 2, 3, 0, 0]
[1, 2]
[1, 2, 0, 0]
```

Sort an array

`void sort(arrayName)`

- Sorts the array into ascending order

```java
import java.util.Arrays;

public class ArraysMethodDemo {
    public static void main(String []args) {
        int[] arr = { 3, 5, 1, 4, 2 };

        Arrays.sort(arr);
        System.out.println(Arrays.toString(arr));

    }
}
```

```
$ javac ArraysMethodDemo.java
$ java ArraysMethodDemo
[1, 2, 3, 4, 5]
```

Search an element in an array

`int binarySearch(arrayName, value)`

- Returns the index of the given value in a **sorted** array if exists
- Returns a negative number if the value doesn't exist

```java
import java.util.Arrays;

public class ArraysMethodDemo {
    public static void main(String []args) {
        int[] arr = { 3, 5, 1, 4, 2 };
        Arrays.sort(arr);

        int loc0 = Arrays.binarySearch(arr, 0);
        System.out.println("0 is found at " + loc0);

        int loc2 = Arrays.binarySearch(arr, 2);
        System.out.println("2 is found at " + loc2);
    }
}
```

```
$ javac ArraysMethodDemo.java
$ java ArraysMethodDemo
0 is found at -1
2 is found at 1
```

Fill an array with a same value

```
void fill(arrayName, value)
```

- Sets every element in the array to the given value

```java
import java.util.Arrays;

public class ArraysMethodDemo {
    public static void main(String []args) {
        int[] arr = { 3, 5, 1, 4, 2 };

        System.out.println("Before: " + Arrays.toString(arr));

        Arrays.fill(arr, 2);
        System.out.println("After: " + Arrays.toString(arr));
    }
}
```

```
$ javac ArraysMethodDemo.java
$ java ArraysMethodDemo
Before: [3, 5, 1, 4, 2]
After: [2, 2, 2, 2, 2]
```