CS1101 Programming and Problem Solving

Dr. Gina Bai

Spring 2023

Logistics

- **ZY-5B** on **zyBook > Assignments**
  - Due: **Wednesday, March 8**, at 11:59pm
- **PA07 – W, A, B** on **zyBook > Chap 11**
  - Due: **Thursday, March 9**, at 11:59pm
- **NO CLASS on Friday, March 10** (before Spring Break)

Recap – File Input

- **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 o read the file

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

Recap – Coding Practice

Write a program that reads in the file CourseDescription.txt, and

- counts the number of tokens in the file
- counts the number of integers in the file

CourseDescription.txt

```
CS 1101 - Programming and Problem Solving

An intensive introduction to algorithm development and problem solving
on the computer. Structured problem definition, top down and modular
algorithm design. Running, debugging, and testing programs. Program
documentation. Not open to students who have earned credit for CS 1104
without permission. Total credit for this course and CS 1104 will not
exceed 3 credit hours. Credit hours reduced from second course taken
(or from test or transfer credit) as appropriate. FALL, SPRING. [3]
```

```java
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;

public class CountTokensInFile {
    public static void main (String[] args) throws FileNotFoundException {

        String fileName = "CourseDescription.txt"; // Relative path
        Scanner input = new Scanner(new File(fileName));

        int countToken = 0, countInt = 0;

        while (__________________) {
            if (____________________) {
                ++countInt;
            }
            ________________; // Consume the token, so we can move to the next one
            ________________;
        }
        input.close(); // Close the scanner

        System.out.println(fileName + " has " + countToken + " token(s), including " +
                countInt + " integer(s).");
    }
}
```

```java
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;

public class CountTokensInFile {
    public static void main (String[] args) throws FileNotFoundException {

        String fileName = "CourseDescription.txt"; // Relative path
        Scanner input = new Scanner(new File(fileName));

        int countToken = 0, countInt = 0;

        while (input.hasNext()) {
            if (__________________) {
                ++countInt;
            }
            ________________; // Consume the token, so we can move to the next one
            ________________;
        }
        input.close(); // Close the scanner

        System.out.println(fileName + " has " + countToken + " token(s), including " +
                countInt + " integer(s).");
    }
}
```

```java
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;

public class CountTokensInFile {
    public static void main (String[] args) throws FileNotFoundException {

        String fileName = "CourseDescription.txt"; // Relative path
        Scanner input = new Scanner(new File(fileName));

        int countToken = 0, countInt = 0;

        while (input.hasNext()) {
            if (input.hasNextInt()) {
                ++countInt;
            }
            ______________; // Consume the token, so we can move to the next one
            ______________;
        }
        input.close(); // Close the scanner

        System.out.println(fileName + " has " + countToken + " token(s), including " +
                countInt + " integer(s).");
    }
}
```

```java
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;

public class CountTokensInFile {
    public static void main (String[] args) throws FileNotFoundException {

        String fileName = "CourseDescription.txt"; // Relative path
        Scanner input = new Scanner(new File(fileName));

        int countToken = 0, countInt = 0;

        while (input.hasNext()) {
            if (input.hasNextInt()) {
                ++countInt;
            }
            input.next(); // Consume the token, so we can move to the next one
            ________________;
        }
        input.close(); // Close the scanner

        System.out.println(fileName + " has " + countToken + " token(s), including " +
                countInt + " integer(s).");
    }
}
```

```java
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;

public class CountTokensInFile {
    public static void main (String[] args) throws FileNotFoundException {

        String fileName = "CourseDescription.txt"; // Relative path
        Scanner input = new Scanner(new File(fileName));

        int countToken = 0, countInt = 0;

        while (input.hasNext()) {
            if (input.hasNextInt()) {
                ++countInt;
            }
            input.next(); // Consume the token, so we can move to the next one
            ++countToken;
        }
        input.close(); // Close the scanner

        System.out.println(fileName + " has " + countToken + " token(s), including " +
                countInt + " integer(s).");
    }
}
```

Coding Practice – Grades from File (Token-Based)

Write a program that adds up the grades from a file via token-based processing.

grades.txt

```
StudentA 98 89.5              100
StudentB 100
StudentC
StudentD 100 89.5 92.5    97.5
```

```
$ javac GradeCalculator.java
$ java GradeCalculator
StudentA: 287.5
StudentB: 100.0
StudentC: 0.0
StudentD: 379.5
```

Sample Solution

```java
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;

public class GradeCalculator {
    public static void main (String[] args) throws FileNotFoundException {
        Scanner input = new Scanner(new File("grades.txt"));
        processGrade(input);
    }

    public static void processGrade(Scanner input) {
        // While there is one more token in the file
        while (input.hasNext()) {
            // The first token in each line is always a String (student name)
            String stuName = input.next();

            double total = 0;
            // Add up all doubles until the Scanner sees another student name
            while (input.hasNextDouble()) {
                total += input.nextDouble();
            }

            System.out.println(stuName + ": " + total);
        }
        input.close();  // Close the Scanner for the input file
    }
}
```

Brainstorm: Would the token-based processing work given the following input file?

gradesWithID.txt

```
123 StudentA 98 89.5            100
456 StudentB 100
789 StudentC
101 StudentD 100 89.5 92.5    97.5
```

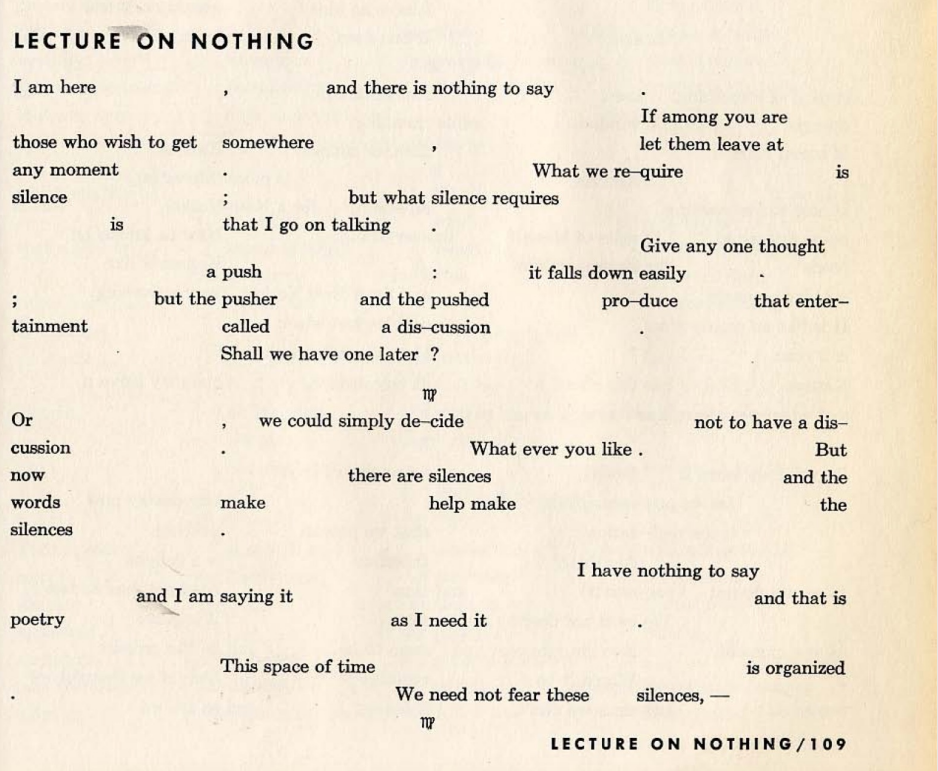
```
$ javac GradeCalculator.java
$ java GradeCalculator
123: 0.0
StudentA: 743.0
StudentB: 889.0
StudentC: 101.0
StudentD: 381.0
```

File Input
(Line-Based Processing)

zyBook Chap 6.4

Line-Based Processing

- **Keeps white space and line breaks** of text being processed
  - Important for poems or formatted text



LECTURE ON NOTHING

I am here , and there is nothing to say .
If among you are
those who wish to get somewhere , let them leave at
any moment . What we re–quire is
silence ; but what silence requires
is that I go on talking .
Give any one thought
a push : it falls down easily .
; but the pusher and the pushed pro–duce that enter–
tainment called a dis–cussion .
Shall we have one later ?

Or , we could simply de–cide not to have a dis–
cussion . What ever you like . But
now there are silences and the
words make help make the
silences .

I have nothing to say
and I am saying it and that is
poetry as I need it .

This space of time is organized
. We need not fear these silences, —

LECTURE ON NOTHING/109

"Functional White"

Line-Based Processing

- **Step 1:** Construct a Scanner for the input file

```
Scanner fileScnr = new Scanner(new File(fileName));
```

- **Step 2:** While there's a line, read the entire line as a String

```
String line = fileScnr.nextLine();
```

- **Step 3:** Construct a Scanner to **tokenize the String** (each line)

```
Scanner lineScnr = new Scanner(line);
```

Line-Based Processing

```java
// A scanner to process the file
Scanner fileScnr = new Scanner(new File(fileName));

// Check if there is one more line in the file
while (fileScnr.hasNextLine()) {
    // Scan in the line as a String
    String line = fileScnr.nextLine();

    // Tokenize a String with Scanner
    Scanner lineScnr = new Scanner(line);

    // Check if there is one more token in the line
    while (lineScnr.hasNext()) {
        // Consume the token
        lineScnr.next();
    }
    lineScnr.close(); // Close the scanner for the line
}
fileScnr.close(); // Close the scanner for the file
```

Sample Solution

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

public class GradeIDCalculator {
    public static void main (String[] args) throws FileNotFoundException {
        Scanner fileScnr = new Scanner(new File("gradesWithID.txt"));
        processGrade(fileScnr);
    }

    public static void processGrade(Scanner fileScnr) {
        // While there is one more line in the file
        while (fileScnr.hasNextLine()) {

            // Read the entire line as a String, where each line contains the info for one student
            String line = fileScnr.nextLine();

            // Tokenize the String by passing it as a parameter to a Scanner
            Scanner lineScnr = new Scanner(line);

            // The first token in each line is an int (student id)
            int stuID = lineScnr.nextInt();
            // The second token in each line is a String (student name)
            String stuName = lineScnr.next();

            double total = 0;
            // Add up all doubles in this line
            while (lineScnr.hasNextDouble()) {
                total += lineScnr.nextDouble();
            }
            lineScnr.close(); // Close the Scanner for the line

            System.out.println(stuName + " (id: " + stuID + "): " + total);
        }
        fileScnr.close();  // Close the Scanner for the input file
    }
}
```

```
$ javac GradeIDCalculator.java
$ java GradeIDCalculator
StudentA (id: 123): 287.5
StudentB (id: 456): 100.0
StudentC (id: 789): 0.0
StudentD (id: 101): 379.5
```