

CSE 2102: Introduction to Software Engineering

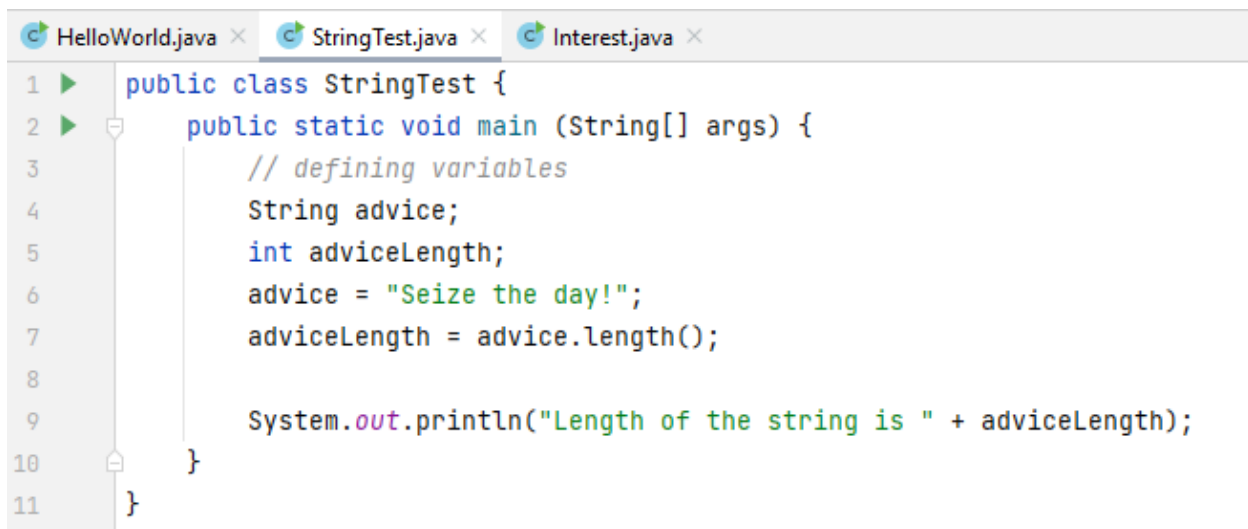
Lab #2: Feb 1, 2022

String Data Type & Scanner Class

Note: Lab assignments are intended for practice. These will not be graded, and need not be submitted. Students are expected to try these by themselves. If you need help, TAs will be available during the lab times, and office hours, which can be found on HuskyCT.

A. String Data Type

The string data type is not primitive, it is an object data type. A value of type `String`, which is an object, contains subroutines that can be used to manipulate that string. These functions are “built into” the Java language. Consider the following piece of code:



```
1 public class StringTest {
2     public static void main (String[] args) {
3         // defining variables
4         String advice;
5         int adviceLength;
6         advice = "Seize the day!";
7         adviceLength = advice.length();
8
9         System.out.println("Length of the string is " + adviceLength);
10    }
11 }
```

The function `.length()` is a built-in function and the `advice.length()` function call returns the number of characters in the string “Seize the day!”. There are many functions in the `String` class, and it would help to familiarize yourself with this class. An interesting fact about strings is that you can use the plus operator, `+`, to concatenate two strings. The concatenation of two strings is a new string consisting of all the characters of the first string followed by all the characters of the second string.

B. Scanner Class

`Scanner` is a class in `java.util` package used for obtaining the input of the primitive data types like `int`, `double`, etc. and `String`. It is the easiest way to read input in a Java program.

- To create an object of `Scanner` class, we usually pass the predefined object `System.in`, which represents the standard input stream. We may pass an object of class `File` if we want to read input from a file.
- To read numerical values of a certain data type for `xyz`, the function to use is `nextxyz()`. For example, to read a value of type `short`, we can use `nextShort()`.
- To read strings, we use `nextLine()`.

- To read a single character, we use `next().charAt(0)`. `next()` function returns the next token/word in the input as a string and `charAt(0)` function returns the first character in that string.

Consider the following code snippet that reads various types of data using the `Scanner` class.

```
ScannerDemo.java x
1  import java.util.Scanner;
2  public class ScannerDemo {
3      public static void main(String[] args) {
4          // Declare the object and initialize with predefined standard input object
5          Scanner sc = new Scanner(System.in);
6
7          // These prompts ("Enter xyz: ") are added for the ease of understanding code flow.
8          // This print statements are not necessary to take input.
9          System.out.print("Enter name: ");
10
11         // String input
12         String name = sc.nextLine();
13
14         // Character input
15         System.out.print("Enter gender (M or F): ");
16         char gender = sc.next().charAt(0);
17
18         // Numerical data input byte, short, and float can be read using similar-named functions.
19         System.out.print("Enter age: ");
20         int age = sc.nextInt();
21     }
```

B. File I/O

So far, you have read input from the command line and written output to a command line. The keyboard and screen input deal with temporary data. When the program ends, the data typed at the keyboard and shown on the screen go away. Files provide you with a way to store data permanently. Therefore, more often than not, you will need to read input from a file, and write output to a file as well. Therefore, in this lab, you will learn to read input and write output to a text file.

The following piece of code can be used to write output that is provided on the screen to a text file. In Java, input and output from a file, keyboard or a screen are all handled by a stream abstraction. The class `PrintWriter` in the Java class library defines the methods needed to create and write to a text file. It is in the package `java.io`, so we will need to begin our program with an import statement. Before we can write to the file, we must open the file, and in this code snippet the name of the file to be opened is hard coded into the text. However, it can be accepted as a command line argument, or using the `Scanner` class after running the program. Another concept you will see here is exception handling. The program may not be able to open and create a file, in which case it will throw a `FileNotFoundException`.

```

import java.io.FileNotFoundException;
import java.io.PrintWriter;
import java.util.Scanner;
public class TextFileOutput
{
    public static void main(String[] args)
    {
        String fileName = "out.txt";
        PrintWriter outputStream = null;
        try
        {
            outputStream = new PrintWriter(fileName);
        }
        catch(FileNotFoundException e)
        {
            System.out.println("Error opening the file" + fileName);
            System.exit(status: 0);
        }
        System.out.println("Enter three lines of text:");
        Scanner keyboard = new Scanner(System.in);
        for (int count = 1; count <= 3; count++)
        {
            String line = keyboard.nextLine();
            outputStream.println(count + " " + line);
        }
        outputStream.close();
        System.out.println("Those lines were written to " + fileName);
    }
}

```

It is important to remember that when you open a file for writing, the program starts with an empty file. Any old contents in the file will be lost. You can open the file for appending by replacing the line:

```
outputStream = new PrintWriter(fileName);
```

with

```
PrintWriter outputStream = new PrintWriter(new FileOutputStream("out.txt", true));
```

Please do not forget to import `java.io.FileOutputStream`.

The following piece of code is used to read input from a file.

```
import java.util.Scanner;
import java.io.File;
import java.io.FileNotFoundException;
public class TextFileInput
{
    public static void main(String[] args)
    {
        String fileName = "out.txt";
        Scanner inputStream = null;
        System.out.println("The file " + fileName +
            "\ncontains the following lines:\n");
        try
        {
            inputStream = new Scanner(new File(fileName));
        }
        catch(FileNotFoundException e)
        {
            System.out.println("Error opening the file " + fileName);
            System.exit(0);
        }
        while (inputStream.hasNextLine())
        {
            String line = inputStream.nextLine();
            System.out.println(line);
        }
        inputStream.close();
    }
}
```

In this code, the `hasNextLine()` method from the `Scanner` class is used. This method returns true if more input data is available to be read by the method `nextLine()`.

Note – Although you will learn and practice conditionals, loops and branching in the next lab, it is assumed that you have basic knowledge of simple programming constructs such as while, if, for, etc.