Reading input

Learning Objectives

- Learn to write a program that reads text written by a user.
- Know what a "string" refers to in programming.
- Know how to join (i.e., "concatenate") strings together.

Input refers to text written by the user read by the program. Input is always read as a string. For reading input, we use the `Scanner` tool that comes with Java. The tool can be imported for use in a program by adding the command `import java.util.Scanner;` before the beginning of the main program's frame (`public class` ...). The tool itself is created with `Scanner scanner = new Scanner(System.in);`.

```java
import java.util.Scanner;

public class Program {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // We can now use the scanner tool.
        // This tool is used to read input.
    }
}
```

Below is an example of a program which asks for user input, reads the string entered by the user, and then prints it.

```java
// Introduce the scanner tool used for reading user input
import java.util.Scanner;

public class Program {

    public static void main(String[] args) {
        // Create a tool for reading user input and name it scanner
        Scanner scanner = new Scanner(System.in);

        // Print "Write a message: "
        System.out.println("Write a message: ");

        // Read the string written by the user, and assign it
        // to program memory "String message = (string that was given as input)
        String message = scanner.nextLine();

        // Print the message written by the user
        System.out.println(message);
    }
}
```

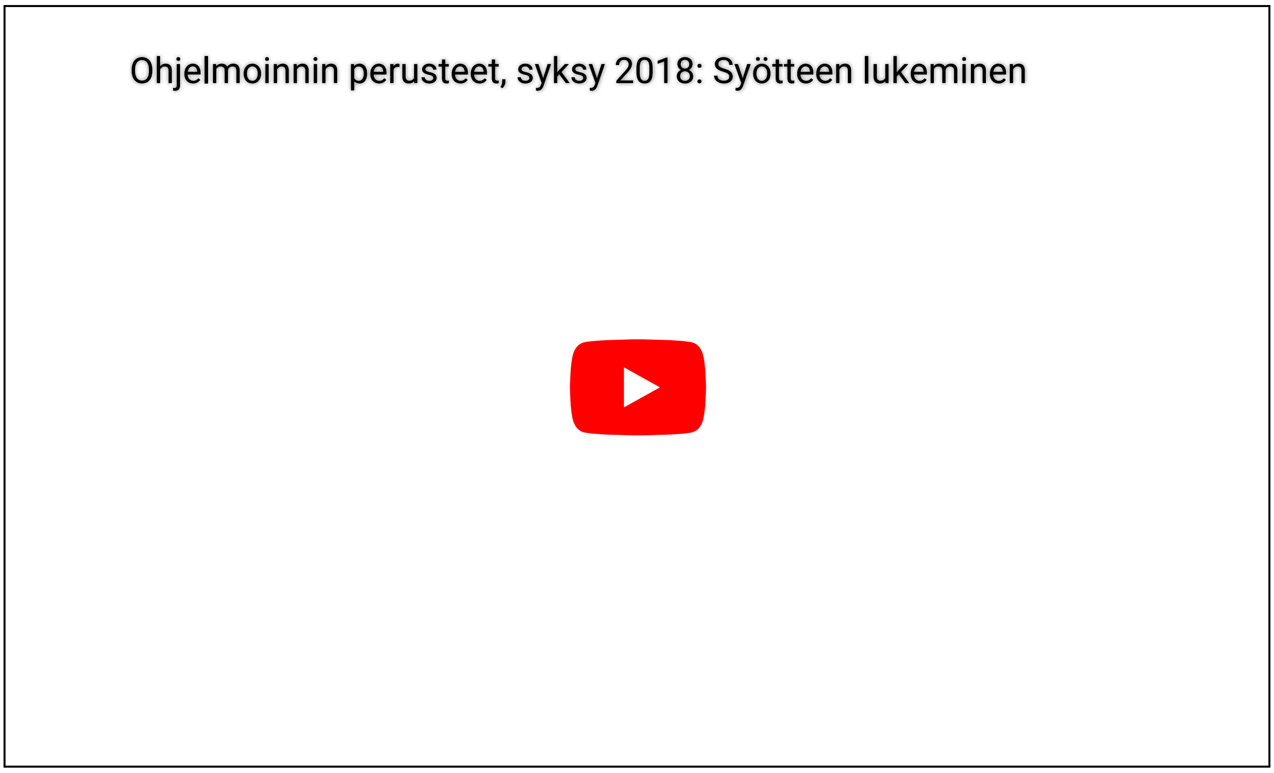
More precisely, input is read with the `scanner` tool's `nextLine()` method. The call `scanner.nextLine()` is left waiting for the user to write something. When user writes something and presses enter, the provided string is assigned to a **string variable** (in this instance `message`). The program is then able to reference the variable `message` later on — in the example above, the variable `message` is referenced in the print command.

When the program is run, its output can look like the example below. In this example, the user has written the text "Hello world" — user input is marked with red in the sample examples.

Sample output

```
Write a message:
Hello world
Hello world
```

The video below shows the process of making a program that reads user input. Watch the video before doing the next programming exercise. Take special notice of how the user input is provided to the Output window located at the bottom of TMC as the program is running.



Programming exercise:

Message

Points
1/1

Write a program that asks the user to write a string. When the user has provided a string (i.e., written some text and pressed the enter key), the program should print the string that was provided by the user.

The exercise template comes with a program template that includes the creation of a Scanner tool.

```java
import java.util.Scanner;

public class Message {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.println("Write a message: ");
        // Write your program here
    }
}
```

Example output for when the user writes "Bye".

Sample output

```
Write a message:
Bye
Bye
```

Example output for when the user writes "Once upon a time...".

Sample output

```
Write a message:
Once upon a time...
Once upon a time...
```

Exercise submission instructions

How to see the solution

Next up, let's take a step back, and examine what on earth `String message = ...` even means.

Fundamentals of Strings

As you might have noticed, in programming we refer to "strings" rather than "text". The term "string" is shorthand for "string of characters" which describes how the computer sees text on a more fundamental level: as a sequence of individual characters.

We've so far used strings in two ways. When practicing the print command, we passed the string to be printed to the print command in quotation marks, and when practicing reading input, we saved the string we read to a variable.

In practice, variables are named containers that contain information of some specified type and have a name. A string variable is declared in a program by stating the type of the variable (`String`) and its name (`myString`, for instance). Typically a variable is also assigned a value during its declaration. You can assign a value by following the declaration with an equals sign followed by the value and a semicolon.

A string variable called `message` that is assigned the value "Hello world!" is declared like this:

```java
String message = "Hello world!";
```

When a variable is created, a specific container is made available within the program, the contents of which can later be referenced. Variables are referenced by their name. For instance, creating and printing a string variable is done as shown below:

```java
String message = "Hello world!";
System.out.println(message);
```

Sample output

```
Hello world!
```

A string enclosed in a programming language's quotation marks is called a "string literal", i.e., a string with a specified value. A common programming mistake is trying to put quotation marks around variable names. If there were quotation marks around the string variable `message`, the program would print the text "message" instead of the "Hello world!" text held by the `message` variable.

```java
String message = "Hello world!";
System.out.println("message");
```

Sample output

```
message
```

Concatenation - Joining Strings Together

The string to be printed can be formed from multiple strings using the `+` operator. For example, the program below prints "Hello world!" on one line.

```java
public class Program {

    public static void main(String[] args) {
        System.out.println("Hello " + "world!");
    }
}
```

The same method can be used to join a string literal and the value of a string variable.

```java
public class Program {

    public static void main(String[] args) {
        String message = "Hello world!";

        System.out.println(message + " ... and the universe!");
    }
}
```

Sample output

```
Hello world! ... and the universe!
```

We can do the same with any number of strings.

```java
public class Program {

    public static void main(String[] args) {
        String start = "My name is ";
        String end = ", James Bond";

        System.out.println(start + "Bond" + end);
    }
}
```

Sample output

```
My name is Bond, James Bond
```

Programming exercise:

Hi Ada Lovelace!

Points

1/1

The exercise template contains the following program.

```
public class HiAdaLovelace {

    public static void main(String[] args) {
        String name = "Ada Lovelace";

    }
}
```

Modify the program so that it prints the contents of the variable `name`, and the printed text is the following in its full form:

Sample output

```
Hi Ada Lovelace!
```

NB! When using the `System.out.println` command, do not pass in the string "Ada Lovelace" as a parameter. Instead, use the existing variable `name`: `System.out.println("Hi " + ...)`

Exercise submission instructions

How to see the solution

Reading Strings

The `reader.nextLine();` command reads the user's input and *returns* a string. If we then want to use the string in the program, it must be saved to a string variable — `String message = scanner.nextLine();`. A value saved to a variable can be used repeatedly. In the example below, the user input is printed twice.

```java
//Introduce the Scanner tool used for reading
import java.util.Scanner;

public class Program {

    public static void main(String[] args) {

        //Create the tool for reading, assign it to variable caller "scanner
        Scanner scanner = new Scanner(System.in);

        //Print user a message "Write a message: "
        System.out.println("Write a message: ");

        //Read the user's string input, save it to program memory
        //"String message = (user input)"
        String message = scanner.nextLine();

        // Print the user input twice
        System.out.println(message);
        System.out.println(message);
    }
}
```

Sample output

```
Write a message:
This will be printed twice...
This will be printed twice...
This will be printed twice...
```

Programming exercise:

Message Three Times

Points

1/1

Write a program that asks the user to write a string. When the user has given a string (that is, written some text and pressed enter), the program must print the user's string three times (you can use the `System.out.println` command multiple times).

The exercise template already includes the code that creates the `Scanner` tool.

```java
import java.util.Scanner;

public class MessageThreeTimes {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.println("Write a message: ");
        // Write your program here
    }
}
```

Example output for when the user writes the string "Hi".

Sample output

```
Write a message:
Hi
Hi
Hi
Hi
```

Example output when the user writes "Once upon a time...".

Sample output

```
Write a message:
Once upon a time...
Once upon a time...
Once upon a time...
Once upon a time...
```

Exercise submission instructions

How to see the solution

Input String as a Part of Output

We noticed in the "Hi Ada Lovelace!" exercise that string literals and string variables can be joined using the `+` operator. The example below demonstrates a program that takes user input and prints it concatenated with a string literal.

```java
import java.util.Scanner;

public class Program {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.println("Write something: ");

        String message = scanner.nextLine();

        System.out.println("You wrote " + message);
    }
}
```

Sample output

```
Write something:
this
You wrote this
```

Programming exercise:
Greeting

Points
1/1

Write a program that prompts the user for their name with the message "What's your name?". When the user has written their name, the program has to print "Hi " followed by the user's name.

The exercise template already includes the code that creates the `Scanner` tool.

```java
import java.util.Scanner;

public class Greeting {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Write your program here
    }
}
```

Example output when user gives the name Ada.

Sample output

```
What's your name?
Ada
Hi Ada
```

Example output when user gives the name Lily.

Sample output

```
What's your name?
Lily
Hi Lily
```

Exercise submission instructions

How to see the solution

Program Execution Waits for Input

When the program's execution comes a statement that attempts to read input from the user (the command `reader.nextLine()`), the execution stops and waits. The execution continues only after the user has written some input and pressed enter.

In the example below, the program prompts the user for three strings. First, the program prints `Write the first string:` , and then waits for user input. When the user writes some text, the program prints `Write the second string:` , and then waits for user input again. This continues for a third time, after which the program prints all three strings.

```
import java.util.Scanner;

public class Program {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.println("Write the first string:");
        String first = scanner.nextLine();
        System.out.println("Write the second string:");
        String second = scanner.nextLine();
        System.out.println("Write the third string:");
        String third = scanner.nextLine();

        System.out.println("You wrote:");
        System.out.println(first);
        System.out.println(second);
        System.out.println(third);
    }
}
```

Sample output

```
Write the first string:
The first
Write the second string:
second
Write the third string:
third
You wrote:
The first
```

```
second
third
```

Programming exercise:

Conversation

Points

1/1

Write a program that works as follows:

Sample output

```
Greetings! How are you doing?
Good thank you!
Oh, how interesting. Tell me more!
Well, there's really nothing to tell.
Thanks for sharing!
```

Sample output

```
Greetings! How are you doing?
Nice and dandy like cotton candy!
Oh, how interesting. Tell me more!
Just went shopping.
Thanks for sharing!
```

The exercise template already includes the code that creates the `Scanner` tool.

```java
import java.util.Scanner;

public class Conversation {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        //Write your program here
    }
}
```

Exercise submission instructions

How to see the solution

In the previous example, we saved the user input to three different string variables. This can be done as long as the variables all have different names (in the example, the names are `first`, `second` and `third`)

```java
import java.util.Scanner;

public class Program {

    public static void main(String[] args) {
        Scanner reader = new Scanner(System.in);

        System.out.println("Write the first string:");
        String first = reader.nextLine();
        System.out.println("Write the second string:");
        String second = reader.nextLine();
        System.out.println("Write the third string:");
        String third = reader.nextLine();

        System.out.println("You wrote:");
        System.out.println(first);
        System.out.println(second);
        System.out.println(third);
    }
}
```

We can form more complicated texts whose content changes depending on the user's input by using more strings. In the example below, the user is told a bit more about the texts they wrote — notice that the order in which the strings are printed can be changed. In the example below, the third input string is printed first.

```java
import java.util.Scanner;

public class Program {
    public static void main(String[] args) {
        Scanner reader = new Scanner(System.in);

        System.out.println("Write the first string:");
        String first = reader.nextLine();
        System.out.println("Write the second string:");
        String second = reader.nextLine();
        System.out.println("Write the third string:");
        String third = reader.nextLine();

        System.out.println("Last string you wrote was " + third + ", which ");
        System.out.println("was preceded by " + second+ ".");
        System.out.println("The first string was " + first + ".");

        System.out.println("All together: " + first + second + third);
    }
```

Sample output

```
Write the first string:
One
Write the second string:
two
Write the third string:
three
Last string you wrote was three, which
was preceded by two.
The first string was one.
All together: onetwothree
```

Programming exercise:

Story

Points

1/1

NB! The example output might align wrong on narrow displays. If you're using only a limited portion of the browser window, or your display is otherwise very narrow, try to stretch the display horizontally to see if the text aligns differently. The exercise expects the text to align as it does on wider displays.

Write a program that asks the user for a character's name and their job. The program then prints a short story.

The output must be as shown below — note, the name and job depend on the user's input.

Sample output

```
I will tell you a story, but I need some information first.
What is the main character called?
Bob
What is their job?
a builder
Here is the story:
Once upon a time there was Bob, who was a builder.
On the way to work, Bob reflected on life.
Perhaps Bob will not be a builder forever.
```

The exercise template already includes the code that creates the `Scanner` tool.

```java
import java.util.Scanner;

public class Story {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Write your program here
    }
}
```

Here's another example output:

Sample output

```
I will tell you a story, but I need some information first.
What is the main character called?
Ada
What is their job?
a Data scientist
Here is the story:
```

```
Once upon a time there was Ada, who was a Data scientist.
On the way to work, Ada reflected on life.
Perhaps Ada will not be a Data scientist forever.
```

Exercise submission instructions

How to see the solution

You have reached the end of this section! Continue to the next section:

→ 4. Variables

Remember to check your points from the ball on the bottom-right corner of the material!

In this part: