Keyboard input

1. Reading from the console using `System.in`

In the previous lessons, we got acquainted with the commands for displaying information on the screen. To do this, we have used the `System.out` object and its `print()` and `println()` methods. It's simple and convenient.

But, as you probably already guessed, displaying information on the screen isn't enough for us. The purpose of most programs is to do something useful for the user. That means that it's very often necessary for the user to be able to enter data from the keyboard.

Just as was the case for output, we also have a special object for data input — `System.in`. But, unfortunately for us, it isn't as convenient as we would like. It lets us read data from the keyboard one character at a time.

To improve on this, we will use another class that, when paired with the `System.in` object, will give us everything we need. For a long time now, Java has had classes to suit every occasion. And we'll get to know one of them now.

---

2. `Scanner` class

The `Scanner` class (full name: `java.util.Scanner`) can read data from different sources, e.g. the console, files, and the Internet. If we want it to read data from the keyboard, then we must pass in the `System.in` object as an argument that will serve as the data source. And then the Scanner object will figure out what to do with it.

Reading from the keyboard using a `Scanner` object would look something like this:

Code	Explanation

`Scanner console = new Scanner(System.in);`
`String name = console.nextLine();`
`int age = console.nextInt();`
	We create a `Scanner` object.
	We read *`a line of text`* from the keyboard.
	We read a *`number`* from the keyboard.

It looks easy, but is it really that simple?

I think you must have a bunch of questions, and now we will answer them.

But first, let's demonstrate an example of a complete program that uses the `Scanner` class:

```
import java.util.Scanner;
public class Solution {
   public static void main(String[] args)
   {
      Scanner console = new Scanner(System.in);
      String name = console.nextLine();
      int age = console.nextInt();

      System.out.println("Name: " + name);
      System.out.println("Age: " + age);
   }
}
```

3. Creating a `Scanner` object

The first question is what is this line `Scanner console = new Scanner (System.in);`?

This line may be confusing, but you will see similar things all the time. So we think it's important to explain what is going on here.

Recall how we usually create a variable with text:

```
String str = "text";
```

Declaring and initializing a string variable

First, we write the variable's type (`String`), then its name (`str`), and finally, after the equals sign, we write the value.

Our bewildering line is actually the same:

```
Scanner console = new Scanner(System.in);
```

Declaring and initializing a `Scanner` variable

Everything to the left of the *equals sign* is the declaration of a variable named `console` whose type is `Scanner`. You could instead call it `s` or`scanner` or even `keyboard`. Then the code would look like this:

```
Scanner s = new Scanner(System.in);
String name = s.nextLine();
int age = s.nextInt();
```

```
Scanner scanner = new Scanner(System.in);
String name = scanner.nextLine();
int age = scanner.nextInt();
```

```
Scanner keyboard = new Scanner(System.in);
String name = keyboard.nextLine();
int age = keyboard.nextInt();
```

I think that makes everything much clearer.

But the code to the right of the *equals sign* is a little more complicated. I'm now referring to `new Scanner(System.in);` That said, there is nothing supernatural happening here either.

In this code, we tell the Java machine: create a new object (the `new` keyword) whose type is `Scanner` , passing in the `System.in` object as the data source for the newly created `Scanner` object.

After executing this whole line, we will have a `Scanner` variable named `console` that our program will use to read data from the keyboard.

---

4. List of methods

In the example above, our `Scanner console` variable stores a reference to a `Scanner` object.

To call methods on an object referenced by a variable, you write a period after the name of the variable, followed by the method name and any arguments. The general appearance of the command is as follows:

```
variable.method(arguments);
```

Calling a method on an object referenced by a variable

Examples:

```
System.out.println("Hello");
```

```
System.out.println(1);
```

If you don't plan to pass arguments to a function, then you just write empty parentheses:

```
variable.method();
```

Calling a method without passing arguments

Example:

```
System.out.println();
```

5. Console input

When we have a `Scanner` object, it's easy to get data from the keyboard.

To *read a line from the keyboard*, you need this command:

```
String str = console.nextLine();
```

When execution of the program reaches this line, it will pause and wait for the user to enter data and press enter. Then everything that the user entered is stored in the variable `str`.

To *read a number from the keyboard*, you need this command:

```
int number = console.nextInt();
```

Everything here is like the previous command. When execution of the program reaches this line, it will pause and wait for the user to enter data and press enter. After that, everything that the user entered is converted to a number and stored in the `number` variable.

If the user entered data that cannot be converted to an integer, the program will generate an error and exit.

To *read a fractional number from the keyboard*, you need this command:

```
double number = console.nextDouble();
```

This statement is very similar to the one with the `nextInt()` method, only it checks that the entered data can be converted to a `double` number.

Example of a program that reads two numbers from the keyboard and displays their sum on the screen:

```
import java.util.Scanner;
public class Solution {
   public static void main(String[] args)
   {
      Scanner console = new Scanner(System.in);
      int a = console.nextInt();
      int b = console.nextInt();

      System.out.println(a + b);
   }
}
```

Note

The user can enter several numbers on a single line, separating them with spaces: the methods of the `Scanner` class know how to handle this. That said, the program reads the numbers only after the user presses `Enter`.

6. Other methods of the Scanner class

By the way, the methods above are not all that the `Scanner` class has to offer. The complete list of methods looks something like this:

Method	Description
`nextByte()`	Reads data and converts it to a `byte`
`nextShort()`	Reads data and converts it to a `short`
`nextInt()`	Reads data and converts it to an `int`
`nextLong()`	Reads data and converts it to a `long`
`nextFloat()`	Reads data and converts it to a `float`
`nextDouble()`	Reads data and converts it to a `double`
`nextBoolean()`	Reads data and converts it to a `boolean`

`next()`	Reads one "token". Tokens are separated by spaces or presses of the enter key
`nextLine()`	Reads an entire line

There are also methods that let you check the next token in the input without actually fetching it (in order to know which method to use to read it).

Method	Description
`hasNextByte()`	Is there a `byte`? Can the input be converted to a `byte`?
`hasNextShort()`	Is there a `short`? Can the input be converted to a `short`?
`hasNextInt()`	Is there an `int`? Can the input be converted to an `int`?
`hasNextLong()`	Is there a `long`? Can the input be converted to a `long`?

`hasNextFloat()`	Is there a `float`? Can the input be converted to a `float`?
`hasNextDouble()`	Is there a `double`? Can the input be converted to a `double`?
`hasNextBoolean()`	Is there a `boolean`? Can the input be converted to a `boolean`?

`hasNext()`	Is there another token?
`hasNextLine()`	Is there another line?

7. Reading data from a string

We previously mentioned that the `Scanner` class can read data from various sources. One of those sources is a *string of text*.

It looks something like this:

```
String str = "text";
Scanner scanner = new Scanner(str);
```

When creating the `Scanner` object, we pass in the string `str` instead of the `System.in` object. And now the `scanner` object will read data from the string. Example:

Program code:	Explanation:

`import java.util.Scanner;`
`public class Solution {`
`  public static void main(String[] args)`
`  {`
`  String str = "10 20 40 60";`
`  Scanner scanner = new Scanner(str);`
`  int a = scanner.nextInt();`
`  int b = scanner.nextInt();`
 	
`  System.out.println(a + b);`
`  }`
`}`
	`// a == 10;`
	`// b == 20;`
	The screen output will be:
	30