COURSEWARE

Input With Scanner

Contents

Overview

We can get input from the user through the console with the use of the Scanner class in the java.util package.
To use the Scanner class we will need to instantiate an object of the class, then we are free to use any of the methods in it's class to take user input, we will go over how to instantiate the Scanner class in the tutorial.
The Scanner class has different methods for user input and allows us to specify the type of variable we want to get from the user, below is a table of the methods.

Method	Description
next()	Reads a String value from the user, but only the first word of the line
nextBoolean()	Reads a boolean value from the user
nextByte()	Reads a byte value from the user
nextDouble()	Reads a double value from the user
nextFloat()	Reads a float value from the user
nextInt()	Reads an int value from the user
nextLine()	Reads a String value from the user
nextLong()	Reads a long value from the user
nextShort()	Reads a short value from the user

There are more methods available within the Scanner class that you may find useful such as the `close()` method which closes the Scanner, however the methods in the above table are the most useful methods for taking input from the user.
If you want to see which other methods are available in the Scanner class, feel free to check out the JavaDoc on it.

Tutorial

```java
import java.util.Scanner;

public class InputWithScanner {

    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        System.out.println("Enter your username");

        String user = scan.nextLine();
    }
}
```

The above example imports the Scanner class from the java.util package, then in the main method we are instantiating the Scanner object passing it an input stream as a parameter, in this case `System.in`.
Then we are printing to the console to tell the user what input we are expecting, and then using the `nextLine()` method to get the String entered by the user.

A Known Issue

When using any of the methods above for reading input such as `next()` or `nextInt()`, these target specific inputs from the user such as one word of a string, an int value, a double value etc, rather than reading the entire line. As such, these methods **do not** read the newline character in the input that is created when the user hits the enter key, as it will grab only a value matching the type it's looking for such as int, long, double etc. `nextLine()` however, will read the **entire** line including the newline character at the end.

This means that when you use `nextLine()` immediately after any of the more specific methods like `next()` or `nextInt()` you will notice that it appears to be skipped. This is because the newline character left behind by the `nextInt()` or `nextLong()` etc is then read by the `nextLine()` which follows it, causing it to return as if you'd hit enter.

```java
import java.util.Scanner;

public class InputWithScanner {

    public static void main(String[] args) {
        // Set up scanner object
        Scanner scan = new Scanner(System.in);

        // Read an int from the user
        System.out.println("Please enter a number: ");
        int first = scan.nextInt();

        // Read a whole line from the user
        System.out.println("Please enter a message: ");
        String second = scan.nextLine(); // <- This will appear to be skipped

        // Print the responses
        System.out.println("First value = " + first);
        System.out.println("Second value = " + second);
    }
}
```

This example would give us the following output in the console:

```
Please enter a number:
5
Please enter a message:
First value = 5
Second value =
```

You can see the user had no opportunity to enter a second value. It appears to skip that line because the `nextLine()` reads the newline character left behind and immediatley returns. To solve this issue, you can simply add an extra

`nextLine()` call between the first and second, to clear the newline character from the scanner:

```
import java.util.Scanner;

public class InputWithScanner {

    public static void main(String[] args) {
        // Set up scanner object
        Scanner scan = new Scanner(System.in);

        // Read an int from the user
        System.out.println("Please enter a number: ");
        int first = scan.nextInt();

        scan.nextLine(); // <- This extra call will consume rest of the previous
line including the newline character

        // Read a whole line from the user
        System.out.println("Please enter a message: ");
        String second = scan.nextLine(); // <- This will no longer be skipped

        // Print the responses
        System.out.println("First value = " + first);
        System.out.println("Second value = " + second);
    }
}
```

An improved solution would be to always read your input using `nextLine()` and convert your input to the proper format you need. For example, you can convert to an integer using the Integer.parseInt(String) method.

Exercises

Calculator

1. Create a project that has four methods; addition, subtraction, multiplication, and division.
   All of these methods should take two numbers as parameters.
2. Create a method to ask the user which of the four methods they wish to use, then take the numbers as user input for the operation and output the result back to the user.

Person

If you have already done this project in the Input and Output module, move onto the Person Extended With Menu exercises.

1. Create a Person class that contains the following attributes:
   - Name
   - Age
   - Job Title
2. Create a method to return all three of these attributes in a formatted String.
   HINT: Override the toString() method.
3. Create some example objects with this class.
4. Create a List implementation and store those objects inside it.
5. Use a stream to output all of your people to the console.
6. Create a method that can search for a specific Person by their name.

Person Extended With Menu

1. Add a menu that prints options to the user for each of the following functionalities; create person, output all people to console, and search for a specific person.

2. Take user input from the menu and run the functionality they have chosen, taking user input where necessary.
   For example when the user wants to create a person, take user input for name, age, and job title.