

Lecture 9

The Input Part of I/O

Chapter 9.4-9.5

'print'

The **print** statement works very similarly to the **println** statement. However, the **print** statement does not put a newline character at the end of the output.

The lines:

```
System.out.print("These lines will be");  
System.out.print("printed on");  
System.out.println("the same line.");
```

That's a 'string'



Will output:

These lines will beprinted onthe same line.

The Scanner Class (input)

To read input from the keyboard we can use the **Scanner** class.

Like **Random**, the **Scanner** class is defined in **java.util**, so again we will use the following statement at the top of our programs:

```
import java.util.*;
```

OR

```
import java.util.Scanner;
```

Scanner is a class

Scanner objects work with **System.in**

To create a **Scanner** object:

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

NOTE:

Like any other object, **keyboard** here is a name “made up” by the coder and can be called anything—**input, feedline, keyIn, data, stuffComingFromTheUser**, etc. —although it should represent a word most apt to its purpose.

In this case we are using ***keyboard*** since it seems most apt.

Example

```
import java.util.Scanner; // Or import java.util.*;
                          // First way is preferred

public class ReadConsole {
    public static void main(String[] args) {
        Scanner cin = new Scanner(System.in); //instantiate

        System.out.print("Enter an integer: ");
        int a = cin.nextInt();
        System.out.print("Enter an integer: ");
        int b = cin.nextInt();

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

Scanner methods

These are for **ints** (integers). There are also Scanner methods available for floats, etc, which we'll see later on in the quarter.

nextInt ()

Does something with the int

hasNextInt ()

Checks to see if there is an int

nextLine ()

Scans all the content until the next line (“clears the buffer”)

```

import becker.robots.*;
import java.util.*;

public class Basic_Keyboard_IO extends Object
{
    public static void main(String[] args)
    {
        System.out.println("THE PROGRAM STARTS HERE!!");

        int numMoves = 0; // Since the loop below stops when numMoves == -1,
                          // it is important that this be anything EXCEPT -1
                          // Setting numMoves to zero works just fine
        int counter = 0;

        Scanner keyboard = new Scanner(System.in);

        City seattle = new City();
        Robot mary = new Robot(seattle, 1, 1, Direction.EAST, 0);

        System.out.println("How many intersections forward would you like the robot to go?");
        if( keyboard.hasNextInt() )
        {
            numMoves = keyboard.nextInt(); // nextInt actually gets the input
            System.out.println ("You asked to move " + numMoves + " spaces");
            counter = 0;
            while( counter < numMoves)
            {
                mary.move();
                counter = counter + 1;
            }
        }
        else
        {
            System.out.println ("You did NOT type in a whole number!");
        }

        keyboard.nextLine(); // DON'T FORGET TO CLEAR ANY REMAINING INPUT!!

        System.out.println ("THE PROGRAM ENDS HERE!!");
    }
}

```

```

import java.util.Scanner; // Or import java.util.*;

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

        int a = 0;
        while (true) // <-- A new kind of while loop we haven't talked about yet
        {
            System.out.print("Enter an integer: ");
            if (keyboard.hasNextInt()) // Checks to see whether an int has been typed in keyboard
            {
                a = keyboard.nextInt();
                keyboard.nextLine(); // newline flush to "clear the buffer"
                break; // <-- We haven't talked about break yet either
            } else
            {
                String next = keyboard.nextLine(); // newline flush
                System.out.println(next + " is not an integer such as 10 or -3.");
            }
        }

        int b = 0;
        while (true) // <-- A new kind of while loop
        {
            System.out.print("Enter an integer: ");
            if (keyboard.hasNextInt())
            {
                b = keyboard.nextInt();
                keyboard.nextLine(); // newline flush
                break;
            } else
            {
                String next = keyboard.nextLine(); // newline flush
                System.out.println(next + " is not an integer such as 10 or -3.");
            }
        }

        System.out.println(a + " * " + b + " = " + a * b);
    }
}

```


A Note on Integer Operations

Division can be tricky.

In a Java program, what is the value of $X = 1 / 2$?

You might think the answer is 0.5...

But, that's wrong.

The answer is simply 0.

Integer division will **truncate** any decimal remainder.

If you are going to divide and need a decimal, then you must use either the **float** or **double** types.

Answering a Question

stackoverflow.com/questions/2315705/what-is-the-difference-between-i-i-in-for-loop-java

2ac63ec3.jpg picture... Ricci Adams' Musict... A Toy Garden: -- Be... Wild Horse Renewa... Shop - ZIIIRO Our Absolute Favori... Washir

related: [stackoverflow.com/questions/1756015/...](#) - jdupont Feb 23 '10 at 2:19

add a comment

7 Answers


active oldest votes

▲ 24 ▼ They both increment the number. `++i` is equivalent to `i = i + 1`.

`i++` and `++i` are very similar but not exactly the same. Both increment the number, but `++i` increments the number before the current expression is evaluated, whereas `i++` increments the number after the expression is evaluated.

```
int i = 3;
int a = i++; // a = 3, i = 4
int b = ++a; // b = 4, a = 4
```

share improve this answer edited Feb 23 '10 at 3:16 answered Feb 23 '10 at 2:16

 David Johnstone 11.4k ● 8 ● 45 ● 63

6 To answer the actual question, however, they're essentially identical within the context of typical `for` loop usage. - Amber Feb 23 '10 at 2:16

Point of pedantry: `i = i + 1` is an expression with a value one more than the initial value of `i`, which would make it more like `++i`. - Tom Hawtin - tackline Feb 23 '10 at 2:48

add a comment