Download

Day 3: Scanner

Download

Agenda

- Quiz 1
- Census Day Reminder
- Lab 1 Q5 Review
- Input
- Scanner
- Closing the Scanner
- Review of Operators
- In Class: Input Practice

Download

Quiz 1

You have 15 minutes to finish the Quiz. When you're done, you can bring it to me at the front or give it to your TA.

Closed note quiz, try your best!

Download

Quiz 1 Review

Do we have any questions to go over?

Download

Friday, Sep 5

Last day to add/drop classes

Download

Lab 1 Quick Debrief

Download

Q3

Int * String doesn't work in Java

Download

Q5

Download

Let's do it live!

How does 127*3 give us 125?

Day 3: Scanner

127 * 3 = 381

381 in binary = 256 + 64 + 32 + 16 + 8 + 4 + 1

This is also equal to:

$$1 * 2^{8} + 0 * 2^{7} + 1 * 2^{6} + 1 * 2^{5} + 1 * 2^{4} + 1 * 2^{3} + 1 * 2^{2} + 0 * 2^{1} + 1 * 2^{0}$$

By removing the powers of 2 and only taking the 1s and 0s

we get: 101111101

Byte takes 8 bits, so if we only look at the last 8 bits, we get 01111101

01111101 is the binary for 125

Download

Test code. Don't assume

Variable Creation Overview

```
{Data type} {Variable Name} = {Value};
```

For example: `byte n = 100;`

If we don't have the n part, what is the variable?

Download

Variable Changing

`{variable} += {value}` may not always print something. Do not print this value out. Instead, change the variable first, then print out the value.

```
int x = 1;

x += 2;

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

Download

Input

In Java, taking input is more complicated than the `input(prompt)` that we saw in Python.

Day 3: Scanner

Download

In order to get input, we must use the `Scanner` class. This requires us to `import` the Scanner class into our program.

At the top of your program, we will include the following code.

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

Download

This will allow us to bring in the Scanner into our code, using Scanner to take the input from the user.

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

Download

The next step is to use our Scanner in our code.

We will create a new Scanner **object**, which effectively creates a new instance or copy of the Scanner.

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

What is this new part?

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

Download

new is a keyword that we use to create a new object. We will see what these objects mean later when we dive deeper into Classes and Objects.

For now, think of it as making a new copy or instance of the Scanner

Download

But what about the `System.in`?

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

Day 3: Scanner

Scanner can take in different types of input locations.

`System.in` simply tells Java to use the Standard Input from system.

This is the same as the user input we used in Python.

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

Download

In this case, we create the variable `input` to store our scanner, allowing us to call various functions connected to the scanner through input.

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

Download

Print then input

To use Scanner, we first must print out the prompt, then on the next line, we can call our input.

The example to the right simply prints the user to enter their name. On the next line, we read what the user typed in.

```
System.out.println("Enter your name
");

String name = input.nextLine();
```

Day 3: Scanner

Download

In the previous slide, we saw
`input.nextLine();`

This will read the data input on the line after the prompt. However, there are other functions we can call from the input.

1. `nextLine()`
2. `next()`
3. `nextInt()`
4. `nextDouble()`
5. `nextLong()`
6. `nextShort()`
7. `nextByte()`
8. `nextBoolean()`

nextLine()

`nextLine()` gives us a String of what was inputted by the user on the line after the new line character.

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

System.out.println("Enter your
name: ");

String name = input.nextLine();
```

Download

next()

`next()` gives us a String of what was inputted by the user, following the print statement.

If we want to output a prompt and take the input on the same line, this is where the regular `System.out.print` comes in handy.

The difference between `next` and `nextLine` is, `next` reads up to the first space character, while `nextLine` will read until the new line character. (\n)

An important note is: We may have to scan in the new line character afterwards, so we often simply have a `input.nextLine();`

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

System.out.print("Enter your name:
");

String name = input.next();

input.nextLine();
```

Download

nextInt()

`nextInt()` will give us the value entered by the user as an integer.

An important note is: We may have to scan in the new line character afterwards, so we often simply have a `input.nextLine();`

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

System.out.print("Enter your age:
");

int age = input.nextInt();

input.nextLine();
```

Download

nextDouble()

`nextInt()` will give us the value entered by the user as a double.

An important note is: We may have to scan in the new line character afterwards, so we often simply have a `input.nextLine();`

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

System.out.print("Enter your age:
");

double age = input.nextDouble();

input.nextLine();
```

We won't go over all of them, as we will explore them during the in-class coding exercise later.

Download

Closing the Scanner

In order to ensure we don't lock the system input to our program, at the end of our code, we should `close` the input for the scanner

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

//Code

input.close();
```

Operators in Java

As we saw a little in the Lab last week, we have very similar operators as in Python, with some slight differences.

Download

Operator	Use	Differences/Limitations
+	Adds two values together	If strings are present, concatenate. Otherwise, attempts to add two values. Some types, such as booleans cannot be added together.
-	Finds the differences of two values	Cannot subtract from a string (same reason as Python)
*	Multiples two values together	Must be the same type/"multiply-able"
/	Divides two values	Must be the same type/"divide-able"; No such thing as // for integer division. / is the primary method of division in Java. If they're both integers, it'll integer divide. If one is a floating point, it'll floating point divide.
%	Divides two values, returns remainder (mod)	Must be the same type/"divide-able"

Day 3: Scanner

Download

Operator	Use	Example

+=	Updates the variable with the added value to it.	`int x = 0;`
		`x += 3;`
		`x -> 3;`

-=	Updates the variable with the subtracted value.	`int x = 10;`
		`x -= 3;`
		`x -> 7;`

*=	Updates the variable with the multiplied value.	`int x = 25;`
		`x *= 20;`
		`x -> 500;`

/=	Updates the variable with the divided value.	`int x = 24;`
		`x /= 3;`
		`x -> 8;`

%=	Updates the variable by modulating it.	`int x = 25;`
		`x %= 5;`
		`x -> 0;`

Download

Operator	Use
>	True if left value is greater than right value
<	True if left value is less than right value
==	True if left value is equal to the right value
>=	True if left value is greater than or equal to the right value
<=	True if left value is less than or equal to the right value
!=	True if left value is not equal to the right value

Download

In Class: Input Practice

Download

Note; a use-case is a situation/circumstance where you would use something. For example, a use-case of an if-statement is when we need to check if two values are the same, then running some code.