COMP 110/L Lecture 3

Kyle Dewey

Outline

- Types (`int` and `String`)
- String concatenation
- Variables
- User input

Types

Expressions

- From the last lab, you wrote code like:
  - `"Hello, world!"`
  - `2 * (1 + 4)`
- Each of these is an expression (produces a value)

Types

- All values are of a particular type
  - `"Hello, world!": String`
  - `2 * (1 + 4): int` (integers)
- Transitively, all expressions are of a particular type

String Concatenation

String Concatenation

Strings can be combined together with the + operator.

String Concatenation

Strings can be combined together with the + operator.

```
"foo" + "bar"
```

String Concatenation

Strings can be combined together with the + operator.

```
"foo" + "bar"
```

```
"foobar"
```

String Concatenation

Strings can be combined together with the + operator.

```
“foo” + “bar”
```

```
“foobar”
```

```
“foo” + “bar” + “baz”
```

String Concatenation

Strings can be combined together with the + operator.

```
“foo” + “bar”
```

```
“foobar”
```

```
“foo” + “bar” + “baz”
```

```
“foobarbaz”
```

Demo:

`StringConcat.java`

Concatenation with `int`

String concatenation also works with
Strings and integers (`int`).

Concatenation with `int`

String concatenation also works with Strings and integers (`int`).

```
"foo" + 7
```

Concatenation with `int`

String concatenation also works with Strings and integers (`int`).

```
"foo" + 7
```

```
"foo7"
```

Concatenation with `int`

String concatenation also works with Strings and integers (`int`).

```
"foo" + 7
```

```
"foo7"
```

```
"bar" + 28
```

Concatenation with `int`

String concatenation also works with Strings and integers (`int`).

```
"foo" + 7
```

```
"foo7"
```

```
"bar" + 28
```

```
"bar28"
```

Demo:

`IntStringConcat.java`

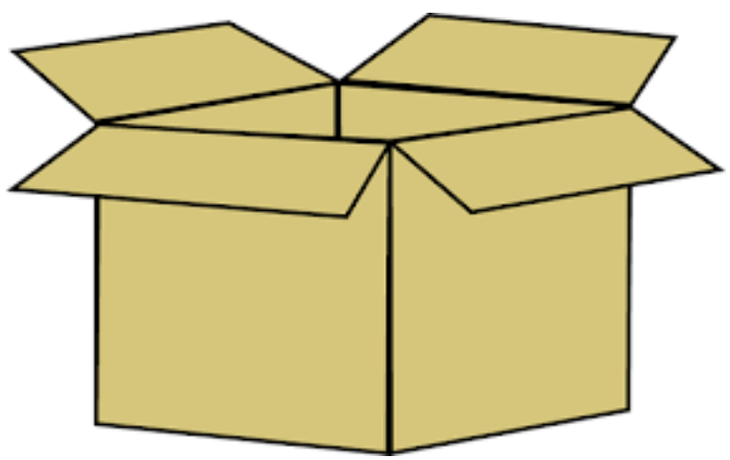
Variables

Variables

- Related to variables in math
- A named "box" you can put a value in

Variables

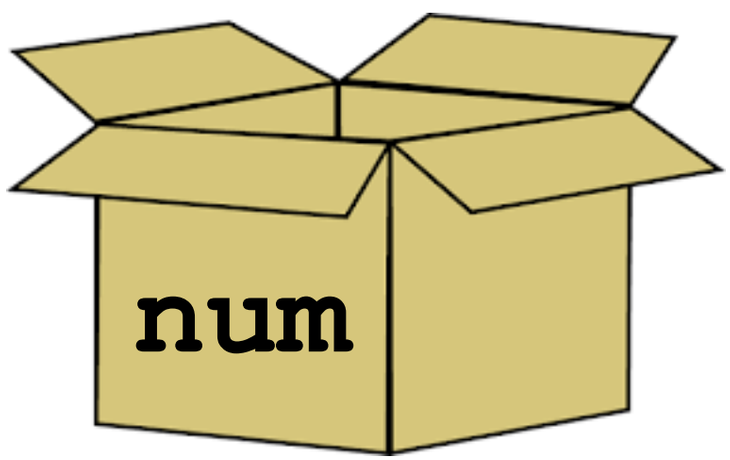
- Related to variables in math
- A named "box" you can put a value in



-I have a box (a variable)...

Variables

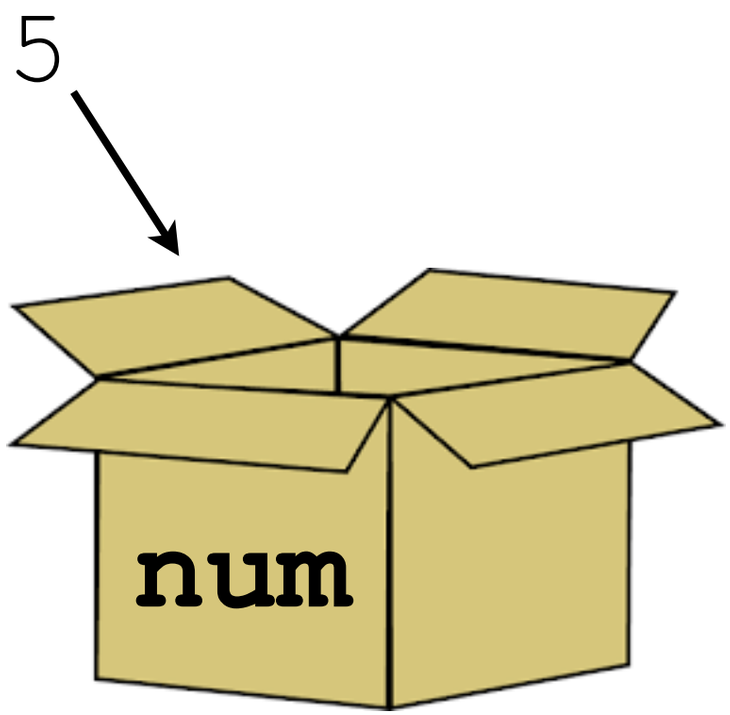
- Related to variables in math
- A named "box" you can put a value in



-...and I'm going to name my box "num"
-I can name a box just about anything I want, though usually the name should reflect the sort of thing I want to put into the box

Variables

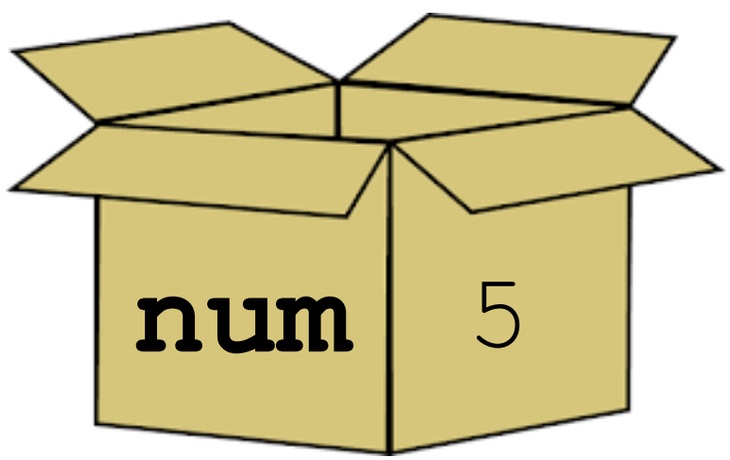
- Related to variables in math
- A named "box" you can put a value in



-I can then put a value into this box. In this case, I put the value 5

Variables

- Related to variables in math
- A named "box" you can put a value in

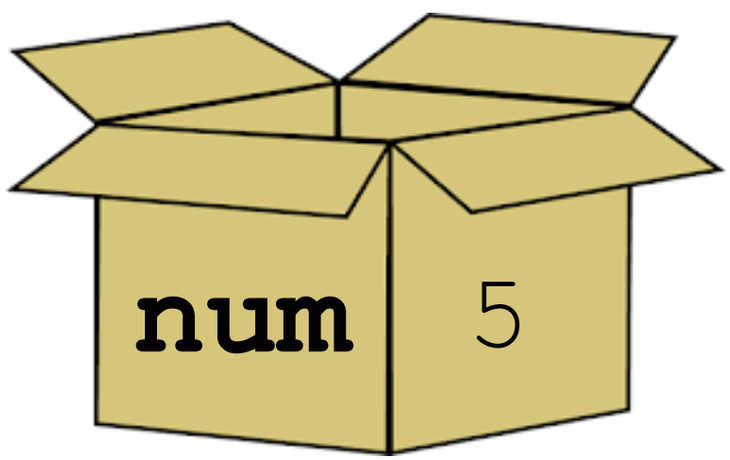


-The box retains the value I put into it. In this case, I put in 5, so it holds 5.

Variables

- Related to variables in math
- A named "box" you can put a value in

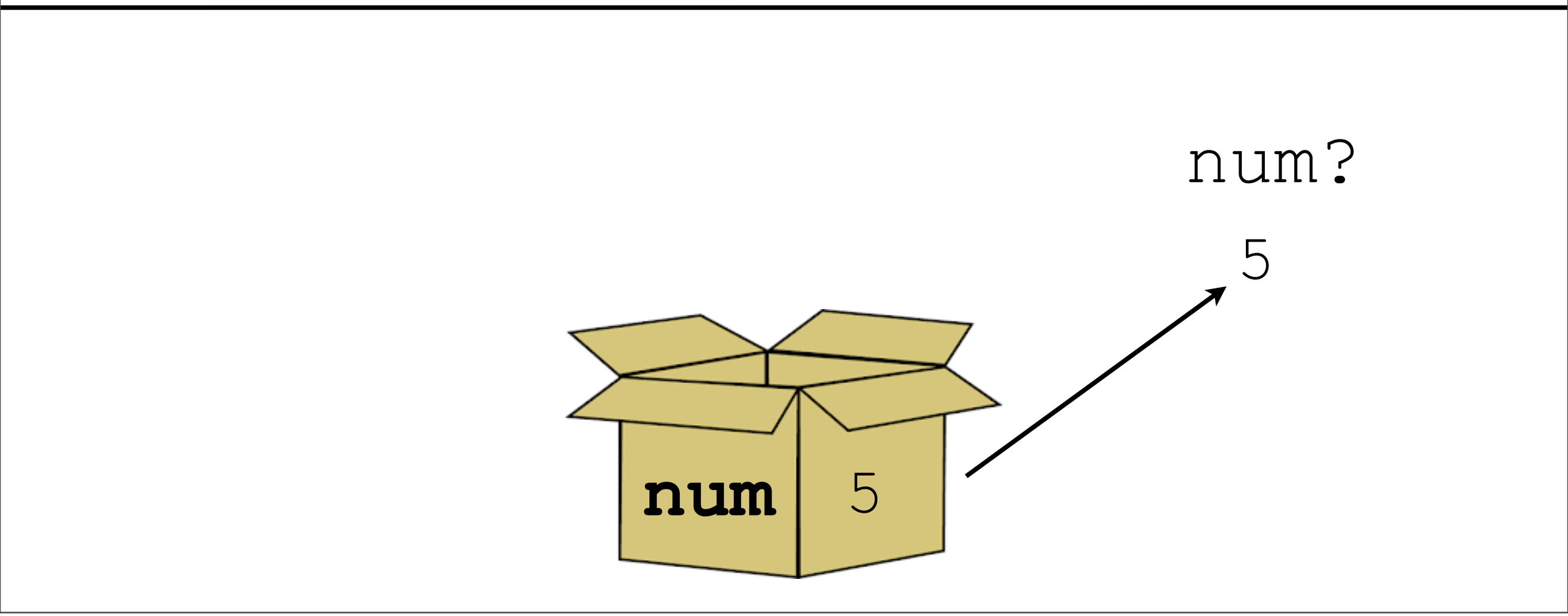
num?



-Later on, I can easily retrieve the value that is in the box

Variables

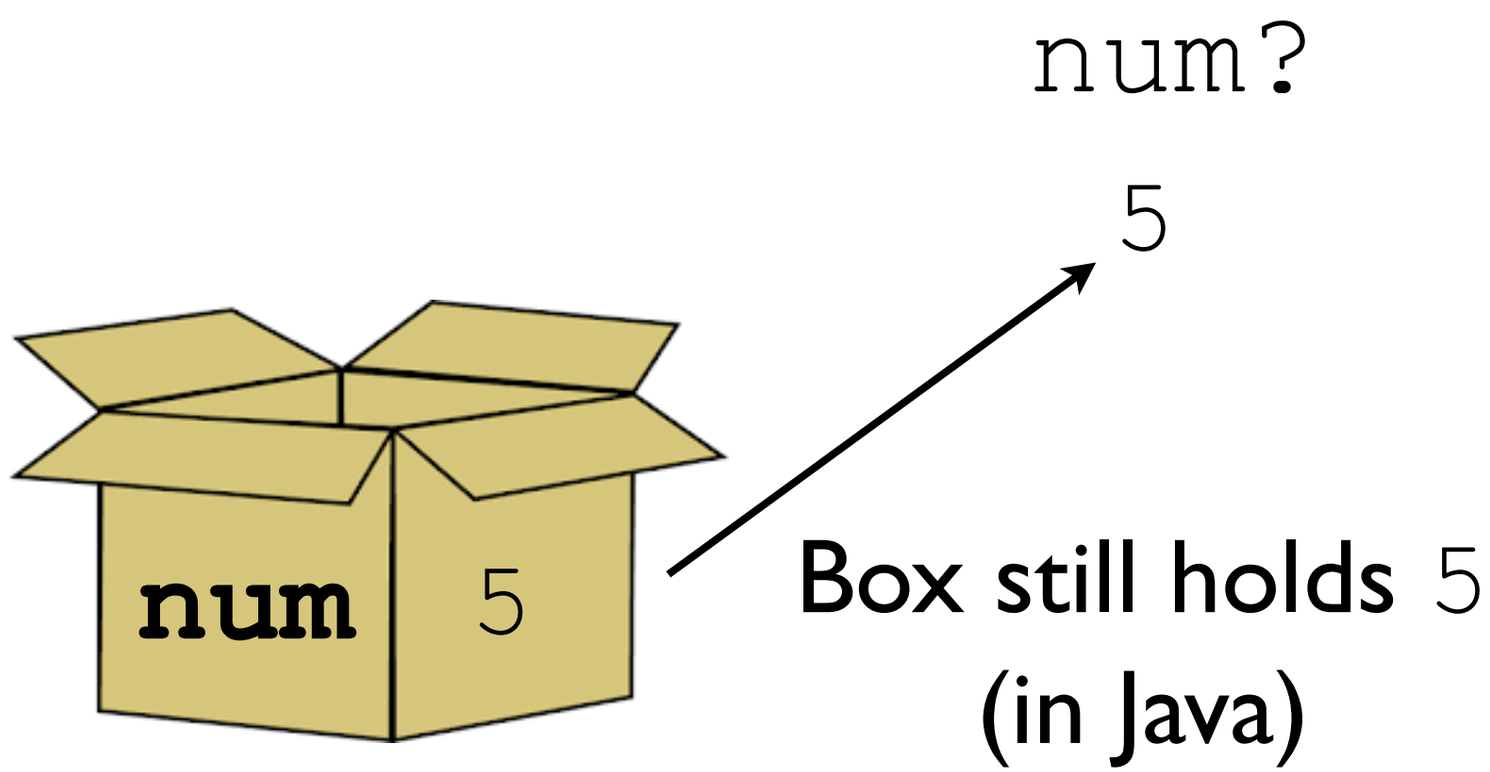
- Related to variables in math
- A named "box" you can put a value in



-Later on, I can easily retrieve the value that is in the box

Variables

- Related to variables in math
- A named "box" you can put a value in



-In Java, the box will retain the value within
-Some languages may or may not retain the value in the box if you ask for the value (C++ gets strange here depending on the context you asked for the value, and usually the box will be empty in Rust)

Getting a Box

In Java, we must *declare a variable* to get a new box.

Part of this declaration includes the *type* of the thing we want to put into the box.

Getting a Box

In Java, we must *declare a variable* to get a new box.

Part of this declaration includes the *type* of the thing we want to put into the box.

```
int num;
```

Getting a Box

In Java, we must *declare a variable* to get a new box.

Part of this declaration includes the *type* of the thing we want to put into the box.

```
int num;
```

Variable named `num`, holds values of type `int`

Getting a Box

In Java, we must *declare a variable* to get a new box.

Part of this declaration includes the *type* of the thing we want to put into the box.

```
int num;
```

Variable named `num`**, holds values of type** `int`

```
String str;
```

Variable named `str`**, holds values of type** `String`

Example:

`VariableDeclarations.java`

Putting Values in the Box

- To put values into variables, we *assign into* them
- Assignment is performed with =

Putting Values in the Box

- To put values into variables, we *assign into* them
- Assignment is performed with =

```
int num;
num = 7;
```

Putting Values in the Box

- To put values into variables, we *assign into* them
- Assignment is performed with =

```
int num;
num = 7;
```

```
int num = 7;
```

Retrieving Values from the Box

- To get a value out of a variable, we need to *access* it
- Variable access is done by referencing a variable in an expression context

Retrieving Values from the Box

- To get a value out of a variable, we need to *access* it
- Variable access is done by referencing a variable in an expression context

```
int num = 7;
int otherNum = num;
int thirdNum = num + otherNum;
```

Example:

`VariableUsage.java`

Question

- Variables can have their values *reassigned*
- Question: what might this code snippet print?

```
int num = 9;
num = 12;
System.out.println(num);
```

Question

- Variables can have their values *reassigned*
- Question: what might this code snippet print?

```
int num = 9;
num = 12;
System.out.println(num);
```

Answer: `12`

-See also VariableReassignment.java

User Input

Program Input

- Programs without input can't do much
  - Can only produce predetermined values
- We'll look at one kind of input: user input from the console/terminal

Reading in Input

New bit of magic: `Scanner`

Reading in Input

New bit of magic: `Scanner`

```
import java.util.Scanner;

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

-The code above creates a Scanner, assigning it into variable in
-Once the Scanner is created, you can do things with it.

Reading in Integers (`int`)

```
Scanner in = new Scanner(System.in);
int first = in.nextInt();
int second = in.nextInt();
int third = in.nextInt();
```

```
// above code reads in
// three integers from the user
```

Demo:

`AddTwo.java`

Reading in Text (`String`)

```
Scanner in = new Scanner(System.in);
String firstLine = in.nextLine();
String secondLine = in.nextLine();

// above code reads in two lines
// of text
```

Demo:

`Parrot.java`

Demo:

`DoubleParrot.java`