

COMP 110/L Lecture 3

Maryam Jalali

Slides are adapted from Dr. 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

A variable is a container which holds values that are used in a Java program.

Do you remember the basic math you learned in school?

$$y = x + 1$$

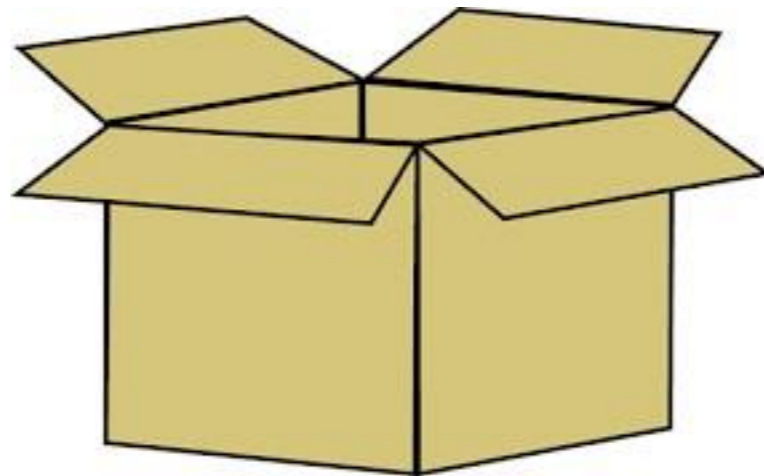
Here, as you can see, **the y variable changes when the x variable is different**. For example:

- ❑ if $x = 1$, then $x + 1 = 2$
- ❑ if $x = 2$, then $x + 1 = 3$
- ❑ if $x = 1.5$, then $x + 1 = 2.5$

In Java, variables play the same role as in the above math example: $y = x + 1$. So, variables are containers that hold values.

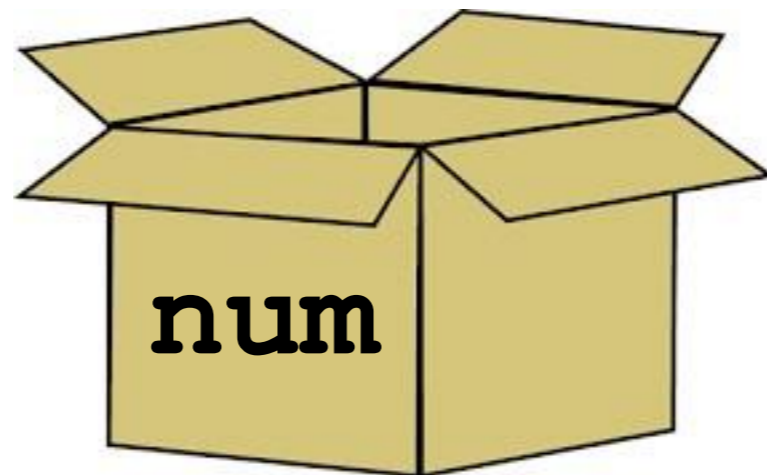
Variables

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



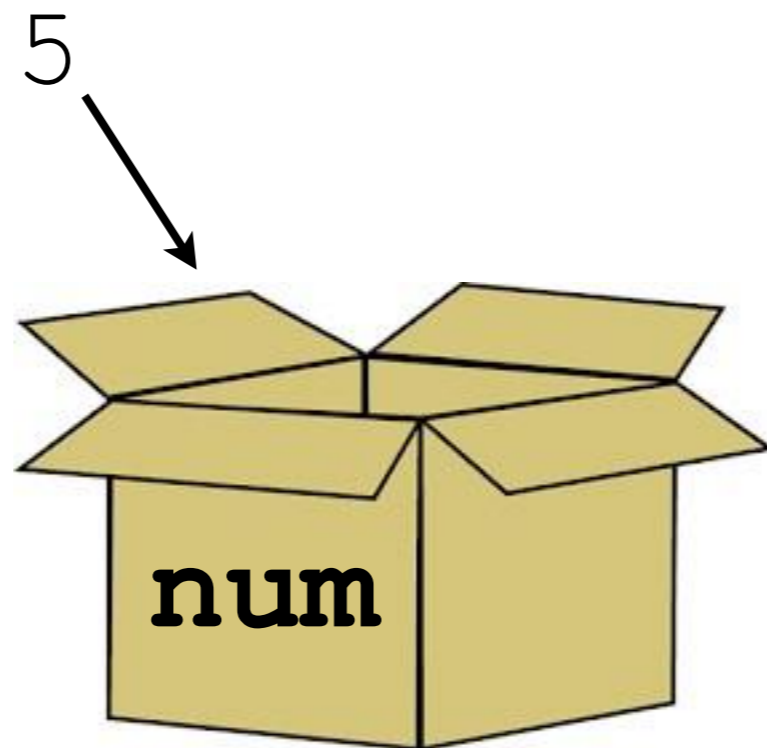
Variables

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



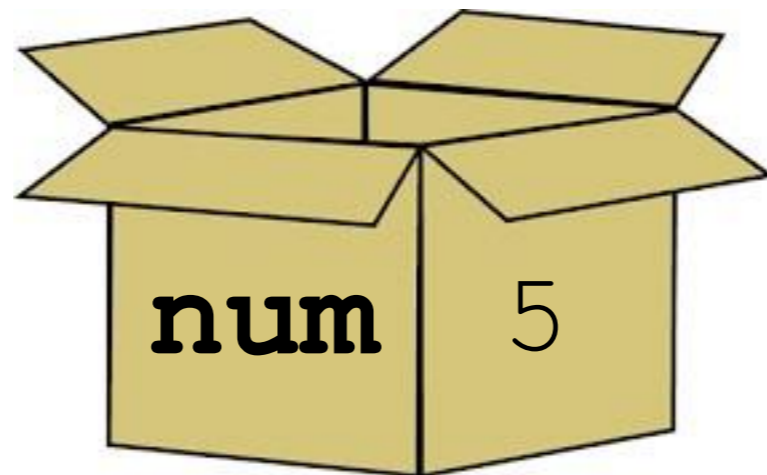
Variables

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



Variables

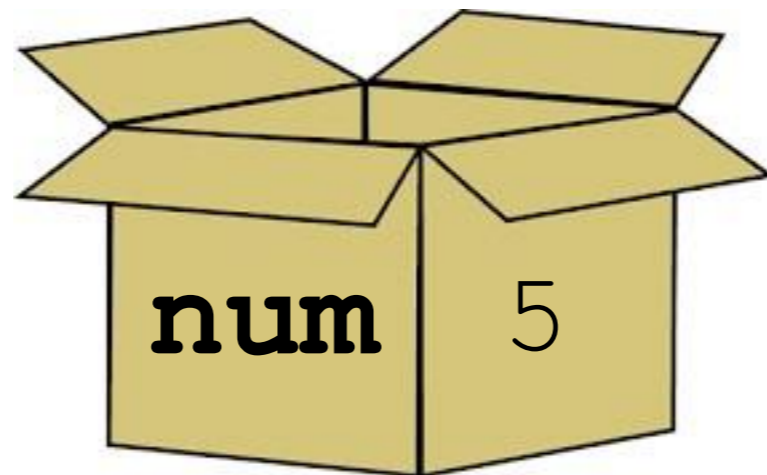
- Related to variables in math
- A named “box” you can put a value in



Variables

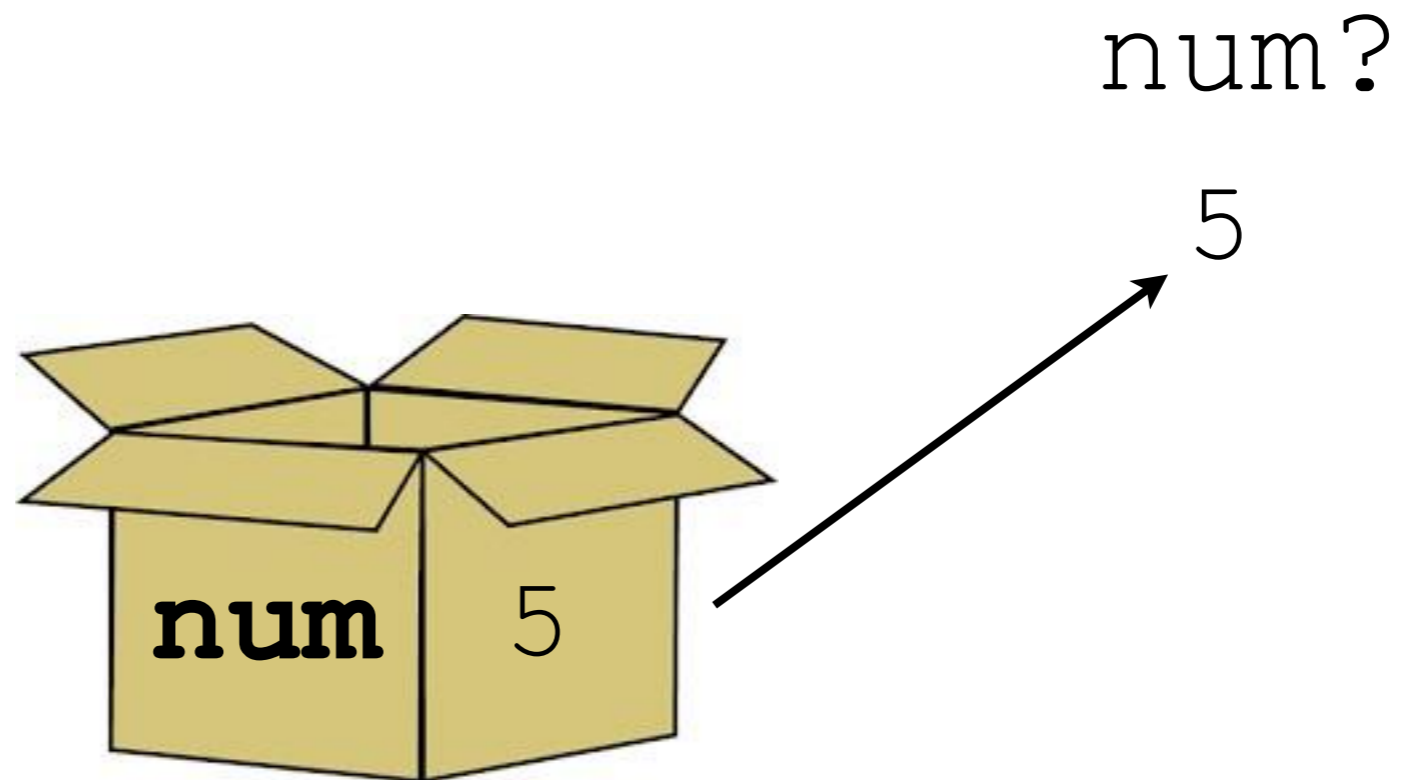
- Related to variables in math
- A named “box” you can put a value in

num?



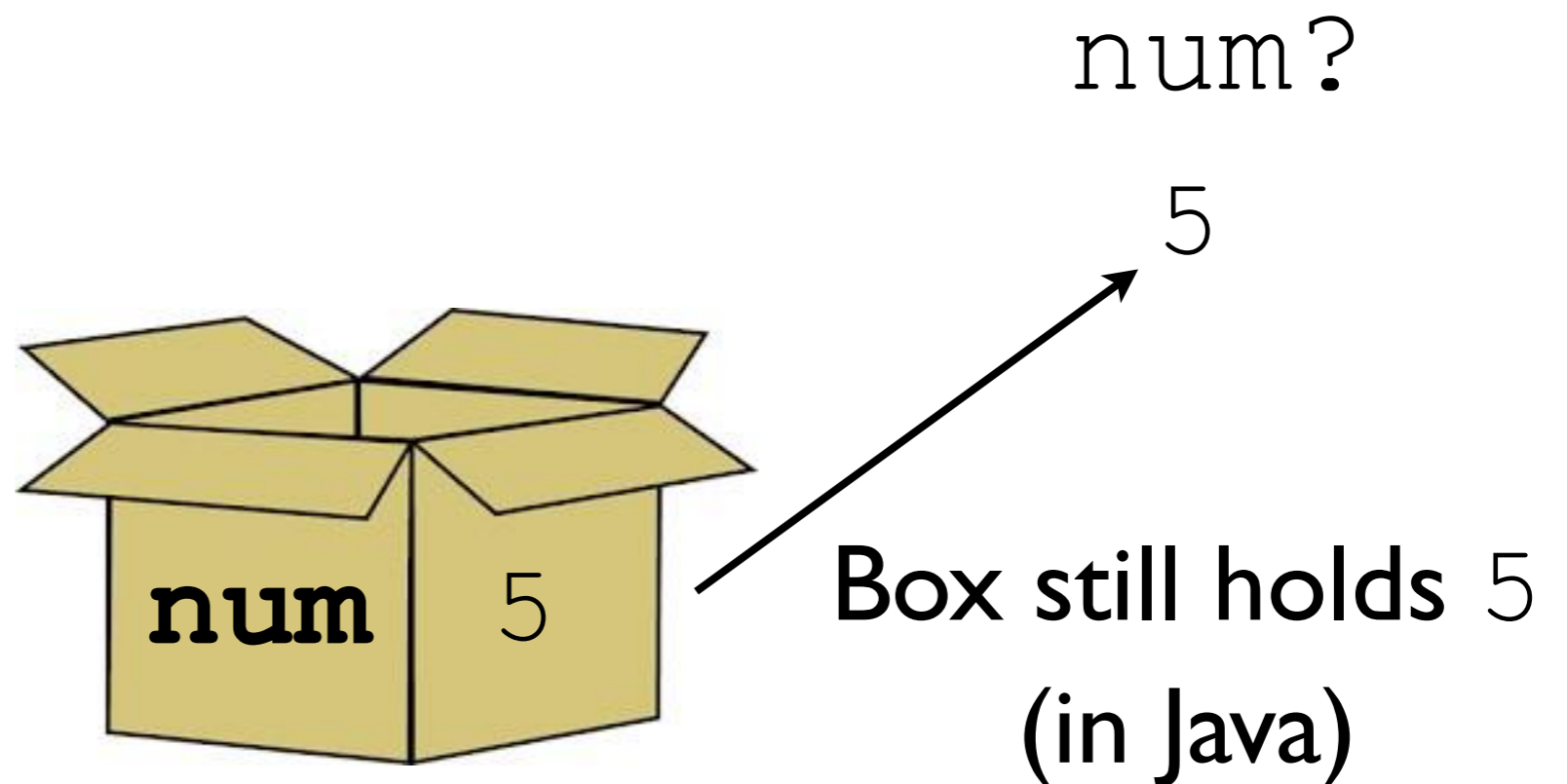
Variables

- 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



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

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);  
        ...  
    }  
}
```

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