COMP 110/L Lecture 9

Kyle Dewey

Outline

- Modulus (`%`) operator
- The `boolean` type
- `if` / `else`
    - Testing approaches with `if` / `else`

Modulus (%) Operator

Modulus (%) Operator

Gets the remainder after division is performed on `int`s.

Modulus (%) Operator

Gets the remainder after division is performed on `int`s.

```
int x = 5 / 2;
```

Modulus (%) Operator

Gets the remainder after division is performed on `int`s.

```
int x = 5 / 2;
```

```
x: 2
```

Modulus (%) Operator

Gets the remainder after division is performed on `int`s.

```
int x = 5 / 2;
```

`x:` 2 2 remainder 1

Modulus (%) Operator

Gets the remainder after division is performed on `int`s.

```
int x = 5 / 2;
```

`x:` 2 2 remainder 1

```
int x = 5 % 2;
```

Modulus (%) Operator

Gets the remainder after division is performed on `int`s.

```
int x = 5 / 2;
```

`x:` 2 2 remainder 1

```
int x = 5 % 2;
```

`x:` 1 2 remainder 1

Modulus (%) Operator

Gets the remainder after division is performed on `int`s.

```
int x = 1 / 2;
```

Modulus (%) Operator

Gets the remainder after division is performed on `int`s.

```
int x = 1 / 2;
```

```
x: 0
```

Modulus (%) Operator

Gets the remainder after division is performed on `int`s.

```
int x = 1 / 2;
```

`x:` `0` `0` remainder `1`

Modulus (%) Operator

Gets the remainder after division is performed on `int`s.

```
int x = 1 / 2;
```

`x:` `0` `0` remainder `1`

```
int x = 1 % 2;
```

Modulus (%) Operator

Gets the remainder after division is performed on `int`s.

```
int x = 1 / 2;
```

x: 0 0 remainder 1

```
int x = 1 % 2;
```

x: 1

Modulus (%) Operator

Gets the remainder after division is performed on `int`s.

```
int x = 1 / 2;
```

`x:` `0` `0` **remainder** `1`

```
int x = 1 % 2;
```

`x:` `1` `0` **remainder** `1`

Example:

ModExample.java

boolean

`boolean`

- Represents the *truth value* of a question
- Only two possible values: `true` and `false`

`boolean`

- Represents the *truth value* of a question
- Only two possible values: `true` and `false`

```
boolean x = true;
```

`boolean`

- Represents the *truth value* of a question
- Only two possible values: `true` and `false`

```
boolean x = true;
```

```
boolean y = false;
```

-No quotes around true and false
-"true" is a string holding the text "true", whereas `true` is a boolean value indicating truth

Comparisons

`boolean` is useful for *arithmetic comparisons*

Comparisons

`boolean` is useful for *arithmetic comparisons*

```
boolean a = 5 > 1; // sets a to true
```

Comparisons

`boolean` is useful for *arithmetic comparisons*

```
boolean a = 5 > 1; // sets a to true
```

```
boolean b = 5 < 1; // sets b to false
```

Comparisons

`boolean` is useful for *arithmetic comparisons*

```
boolean a = 5 > 1; // sets a to true
```

```
boolean b = 5 < 1; // sets b to false
```

```
boolean c = 5 <= 5; // sets c to true
```

Comparisons

`boolean` is useful for *arithmetic comparisons*

```
boolean a = 5 > 1; // sets a to true
```

```
boolean b = 5 < 1; // sets b to false
```

```
boolean c = 5 <= 5; // sets c to true
```

```
boolean d = 6 >= 5; // sets d to true
```

Comparisons

`boolean` is useful for *arithmetic comparisons*

```
boolean e = 5 == 5; // sets e to true
```

Comparisons

`boolean` is useful for *arithmetic comparisons*

```
boolean e = 5 == 5; // sets e to true
```

```
boolean f = 5 == 6; // sets f to false
```

Comparisons

`boolean` is useful for *arithmetic comparisons*

```
boolean e = 5 == 5; // sets e to true
```

```
boolean f = 5 == 6; // sets f to false
```

```
boolean g = 5 != 5; // sets g to false
```

Comparisons

`boolean` is useful for *arithmetic comparisons*

```
boolean e = 5 == 5; // sets e to true
```

```
boolean f = 5 == 6; // sets f to false
```

```
boolean g = 5 != 5; // sets g to false
```

```
boolean h = 5 != 6; // sets h to true
```

String Concatentaion

Works as you might expect

String Concatentaion

Works as you might expect

```
true + "foo"
```

String Concatentaion

Works as you might expect

```
true + “foo”
“truefoo”
```

String Concatentaion

Works as you might expect

```
true + “foo”
```

```
“truefoo”
```

```
“bar” + false
```

String Concatentaion

Works as you might expect

```
true + “foo”
“truefoo”
```

```
“bar” + false
“barfalse”
```

Example:

Comparisons.java

if / else

`if` / `else`

Used to *conditionally* execute code
based on a `boolean` truth value

if / else

Used to *conditionally* execute code based on a `boolean` truth value

```
if (true) {
  System.out.println(“yes”);
} else {
  System.out.println(“no”);
}
```

`if` / `else`

Used to *conditionally* execute code based on a `boolean` truth value

```
if (true) {
  System.out.println("yes");
} else {
  System.out.println("no");
}
```

Prints `yes`

if / else

Used to *conditionally* execute code based on a `boolean` truth value

```
if (5 < 2) {
  System.out.println("yes");
} else {
  System.out.println("no");
}
```

if / else

Used to *conditionally* execute code based on a `boolean` truth value

```
if (5 < 2) {
  System.out.println("yes");
} else {
  System.out.println("no");
}
```

Prints `no`

Example:

IsGreaterThan5.java

-The real utility if/else is when we don't know the truth value of the condition ahead of time

Example:

MultipleReturn.java

-The real utility if/else is when we don't know the truth value of the condition ahead of time

Testing Advice with `if` / `else`

- Ideally, for each `if` / `else`, have *two* tests
  - One for if the condition is `true`
  - Another for if the condition is `false`

Testing Advice with `if` / `else`

- Ideally, for each `if` / `else`, have *two* tests
  - One for if the condition is `true`
  - Another for if the condition is `false`

Question: which tests may be good for testing absolute value?

Testing Advice with `if` / `else`

- Ideally, for each `if` / `else`, have *two* tests
  - One for if the condition is `true`
  - Another for if the condition is `false`

Question: which tests may be good for testing absolute value?

A positive value and a negative value

Example:

MultipleReturnTest.java