CSE 11
Accelerated Intro to Programming
Lecture 3

Greg Miranda, Spring 2021

This lecture is being recorded

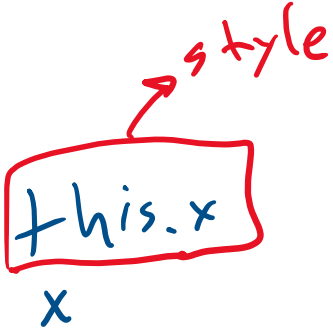
Announcements

- Quiz 3 due Monday @ 8am
- Survey 1 due tonight @ 11:59pm → required as part of engagement
- PA1 due Wednesday @ 11:59pm

PA0.5 → resubmission → 100%
↳ late adds

Program Steps

```
class Example {
  int x = 3 + 2;
  int y = this.x * 4;
}
```



⇒

```
class Example {
  int x = 5;
  int y = this.x * 4;
}
```

5 is a value

⇓

```
class Example5 {
  int x = 5
  int y = 5 * 4;
}
```

⇓

```
class Example5 {
  int x=5;
  int y = 20;
}
```

New Example1()
this.x = 5
this.y = 20

Expressions

binary operator

op1 + op2

~

*

/

- int x = 3 + 2;
  - 3 + 2
    - Arithmetic expression
    - Binary operator expression
- int y = this.x * 4;
  - this.x
    - Field access expression
  - this.x * 4
    - Arithmetic expression where the left-hand operand is a field access expression

Methods

- New class – MethodExample
- In programming, we often want to describe a computation once
  - Then reuse it on different numbers, or different values
  - Write once, use it over and over again
- Example:
  - Take two numbers and add up their squares
    - int sos1 = 3 * 3 + 5 * 5;
    - int sos2 = 4 * 4 + 7 * 7;

- Define a method to do the same thing

```
int sumSquares(int n, int m) {
  return n * n + m * m;
}
```

N^2 M^2

- Vocabulary:
  - Method definition
  - Parameters
  - Method body
    - return keyword

- Running it…
  - Method definition doesn't change what prints out or any of the fields
  - Run command – only prints out the values of the fields
- Can use sumSquares() to do the calculation
  - int ans1 = this.sumSquares(3, 5);
  - int ans2 = this.sumSquares(4, 7);

- Vocabulary:
  - Called the method
  - Arguments

- Methods: one of the building blocks for building programs
  - Not just useful for arithmetic
  - Useful for many more things
- Why do we care about methods?
  - Methods give us a centralized place to write a calculation
    - Change in one place, every place that uses the method will see that update
  - As program gets large:
    - Might have 100s of places where we want to use a formula or calculation
      - Update them all by changing one place
  - Methods are self documenting – with meaningful names

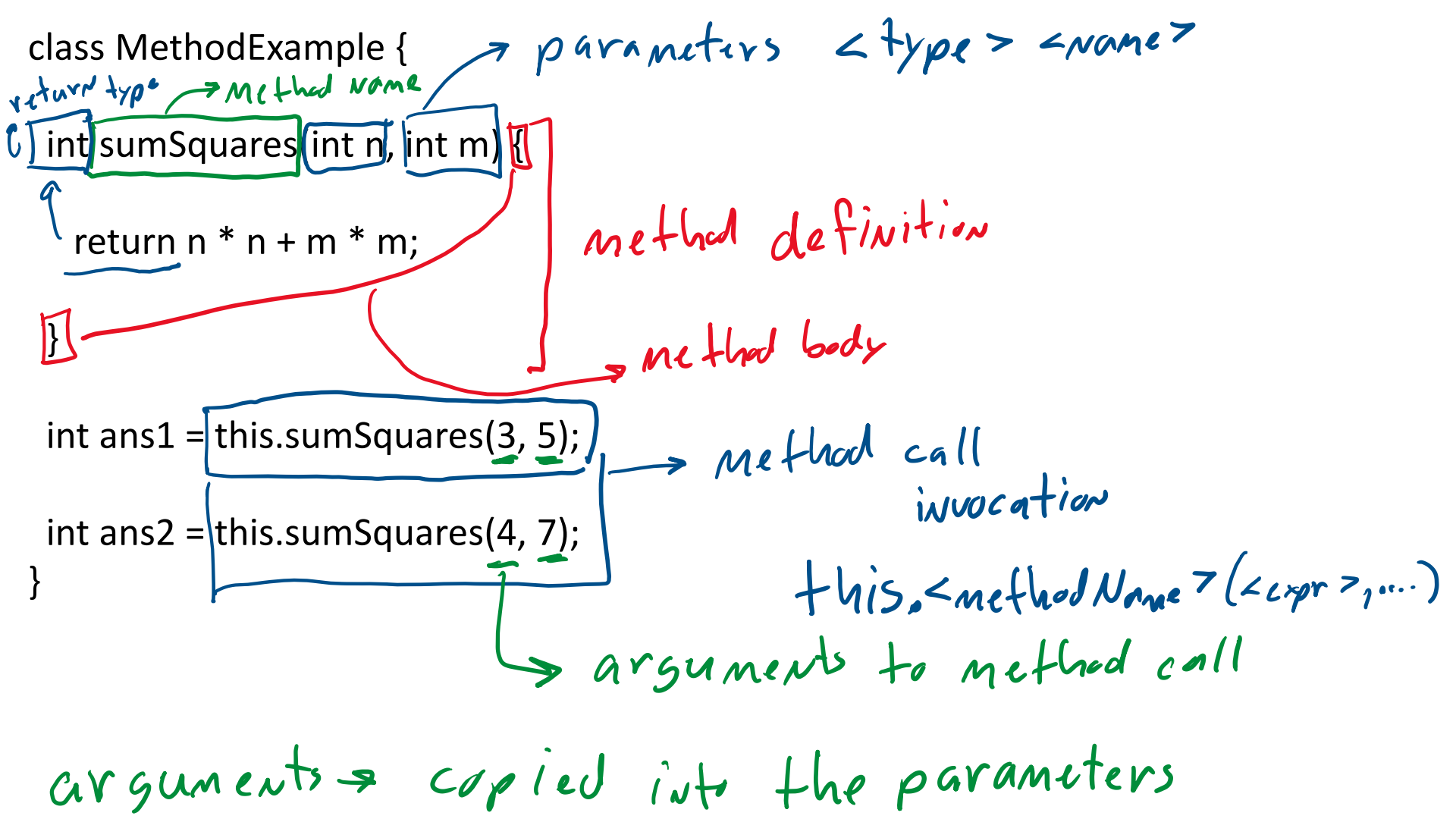
↳ style

this. sumSquares

↳ optional → style

camelCase

↑ ↑


class MethodExample {
return type
method name
parameters <type> <name>
int sumSquares int n, int m) {
return n * n + m * m;
}
method definition
method body
int ans1 = this.sumSquares(3, 5);
int ans2 = this.sumSquares(4, 7);
}
method call invocation
this.<methodName>(<expr>,...)
arguments to method call
arguments → copied into the parameters

int
string
↑
<type> <methodName> ([<parameter>, ...]) {
[<body>]

return <expr>; // required for int/string/etc.
}

void → return is optional → `return;`

Process

```
class MethodExample {
  int sumSquares(int n, int m) {
    return n * n + m * m;
  }
  int ans1 = this.sumSquares(3, 5);
  int ans2 = this.sumSquares(4, 7);
}
```

34

method call

sumSquares (N=3, m=5) {

return 3*3 + 5*5;

}

9 + 25

34

65

method call

sumSquares (N=4, m=7) {

return 4*4 + 7*7;

}

16 + 49

65

Can you change order of the arguments?

num exp

power(5, 2); $5^2 \rightarrow 25$

power(2, 5); $2^5 \rightarrow 32$

order of arguments matters!!

sumSquares("3", "5");

↳ compiler errar

```
String someMethod(String v1, int i2) {
    return v1 + i2;
}
```

```
String s = this.someMethod("a", 5);
```
"a5"

```
String s2 = this.someMethod(5, "a");
```
↳ compiler error