CSE 11
Accelerated Intro to Programming
Lecture 2

Greg Miranda, Summer 1 2021

This lecture is being recorded

Announcements

- PA0.5 due Thursday @ 11:59pm
- PA1 due Thursday @ 11:59pm
- Quiz 1 released today @ 11am
  - Due Friday @ 11:59pm
  - Can find it in Gradescope
    - Can take it as often as you want before the due date

50% to get credit

Text

whole #s

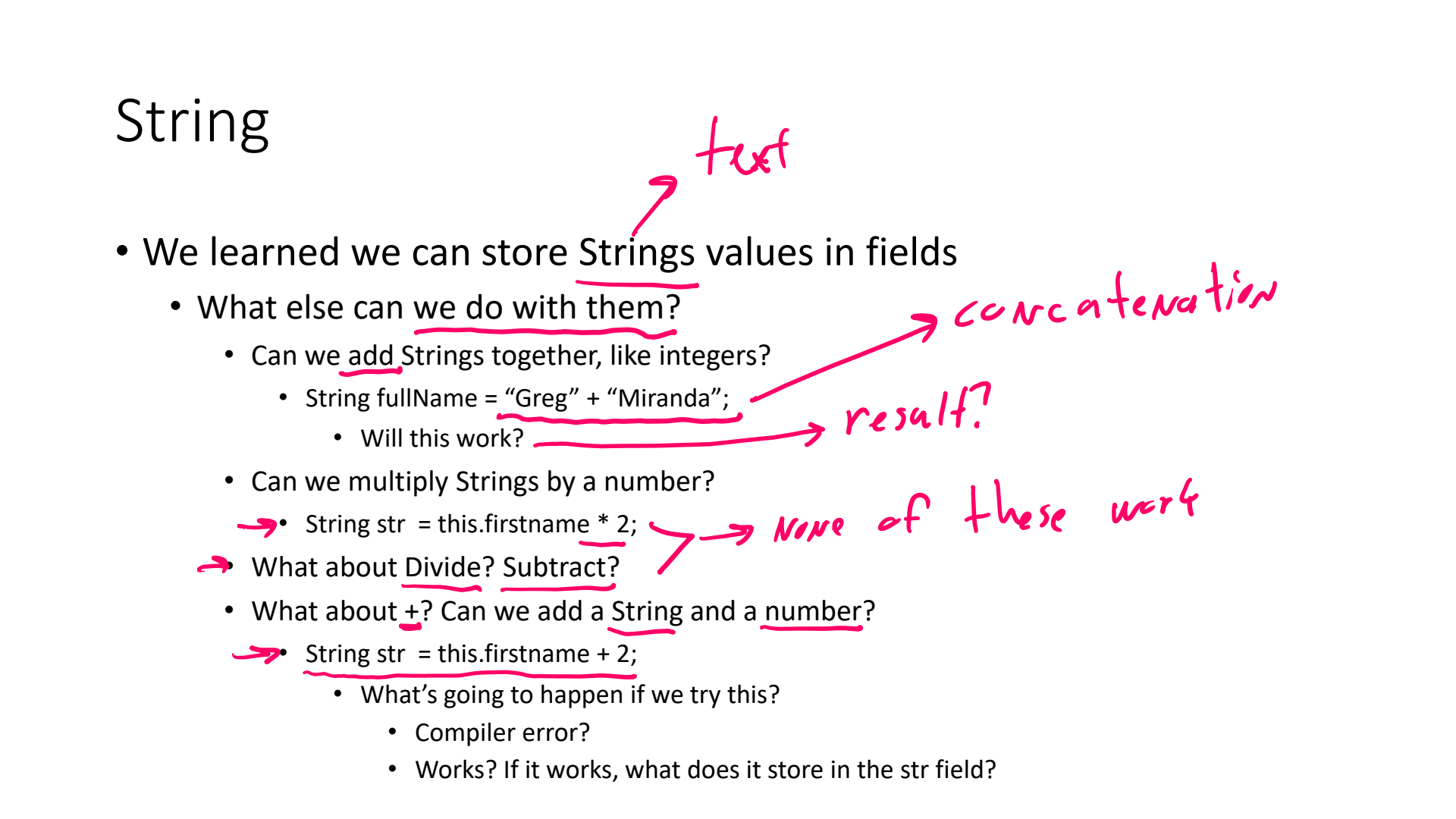
- Integers (int) – common kind of data programmers work with
- New kind of data – also really common – text
  - Examples: usernames, passwords, email, names, addresses
  - Data type for text - String
- Previous examples - had int as the type
  - int numberOfStaff = 14;
- Now – using String as the type
  - String name = “Greg Miranda”; //String value, string literal
- String className = 11;
  - What happens? Does it work? → no → Strongly typed
- String className = “11”;
  - What happens? Does it work? Is it text or a number?

Types

- int – integer type – integer literal
- String – text type – string literal (written in double quotes)
- Java will enforce that we always
  - store string values in String typed fields
  - numeric values in numeric typed fields
- Programmer's job to get this right
  - Java will give an error message if we don't

String

- We learned we can store Strings values in fields
  - What else can we do with them?
    - Can we add Strings together, like integers?
      - String fullName = “Greg” + “Miranda”;
        - Will this work?
    - Can we multiply Strings by a number?
      - String str = this.firstname * 2;
    - What about Divide? Subtract?
    - What about +? Can we add a String and a number?
      - String str = this.firstname + 2;
        - What’s going to happen if we try this?
          - Compiler error?
          - Works? If it works, what does it store in the str field?



- We can + other things besides numbers to Strings and get similar behavior
  - More on this in upcoming weeks
- Adding Strings and numbers
  - Can be convenient
    - Can turn a number into text
  - Can also be confusing
    - String className = "11" + 200;
    - int klassName = 11 + "200";
      - Error
    - String klassName = 11 + "200";
  - Java does do this automatic conversion of Strings and numbers
    - Be careful in your own code

Vocabulary

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

class definition

field definition
(more kinds of defs coming)

<type> <name> = <expression>;

class <ClassName>:

3

"inside" the class definition

"body" of the definition

A class def has a sequence of field def in the body

How many field definitions are in this class?

```
1 class C {
2   ① int a = 10;
3   ② String b = 5 + "A";
4 }
```

2 field def

How many field definitions are in this class?

```
1 class D {
2  ① int a = 10;
3  ② String b = this.a + " dollars";
4 }
```

2 field def

Do you think there's a limit on how many field definitions can be in a class?

Java has no limit

Program Steps

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

expression

⇓

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

5 is a value

⇓

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

⇓

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

3

```
./run Example

new Example:1 (
    this.x = 5
    this.y = 20
)
```

v1 + v2
-
*
/
%

Expressions

- int x = 3 + 2;
  - 3 + 2
    - Arithmetic expression
    - Binary operator expression
- int y = this.x * 4;
  - this.x
    - Field access expression → replaced w/ value of X → 5
  - this.x * 4
    - Arithmetic expression where the left-hand operand is a field access expression

5 * 4

Methods (functions, procedures)

- 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

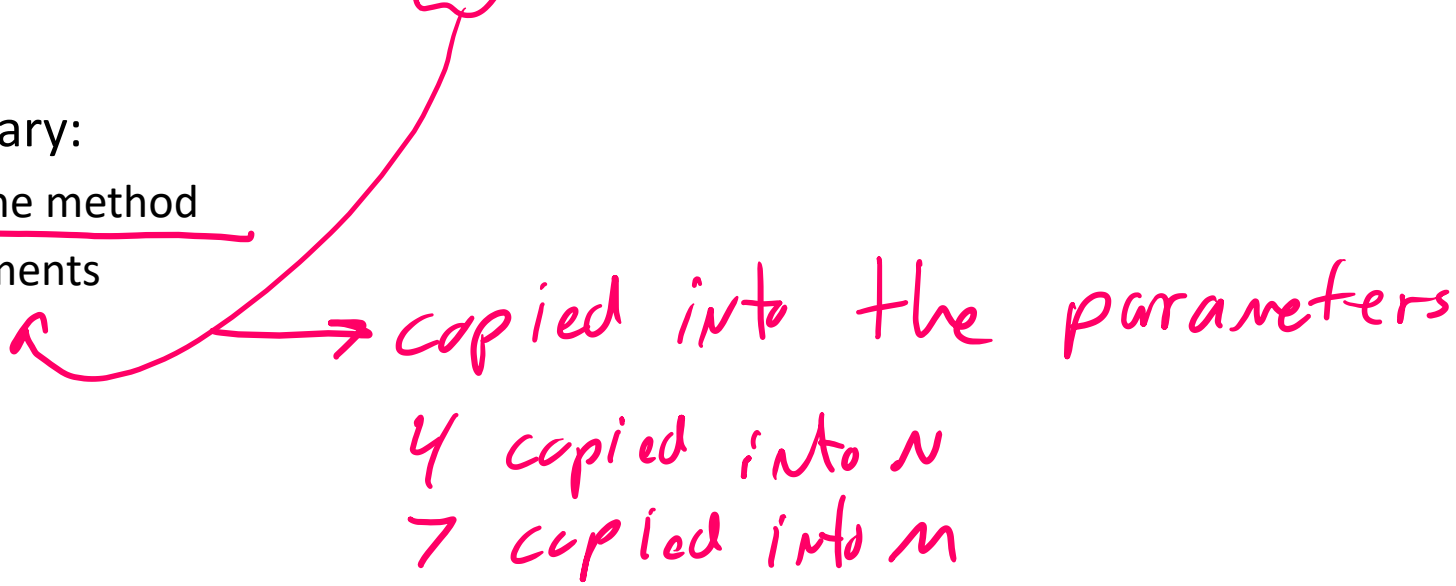
```
int sumSquares(int n, int m) {
  return n * n + m * m;
}
```

parameters

signature
method header
method declaration

- 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:
  - Call 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

camelCase

↑lower ↑upper

```
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);
}
```

Method Definition

```
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);
}
```