

CSE 11

Accelerated Intro to Programming

Lecture 2

Greg Miranda, Summer 2 2021

This lecture is being recorded

Announcements

- Discussion starts today @ 2pm
 - Will cover PA0.5 and PA1
- Lecture Quiz 1 released today @ 11am
 - Based on the first two lectures
- PA0.5 released yesterday – due Thursday
- PA1 released yesterday – due Thursday

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 className = 11 + "200";`
 - Error
 - `String className = 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;  
}
```

How many field definitions are in this class?

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

How many field definitions are in this class?

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

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

Program Steps

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

Expressions

- `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;  
}
```

- 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

```
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);  
}
```