Elements of Programs

Effective Programming in Scala

Elements of Programs

Let us start with a very basic reminder of programming concepts to make sure we are all on the same page and agree on the vocabulary.

What Are Programs Made Of?

A program **expresses** a **computation**.

For instance:

What is the result of adding one to one?

What Are Programs Made Of?

A program **expresses** a **computation**.

For instance:

What is the result of adding one to one?

```
1 + 1
```

Evaluating the program "1 + 1" returns the value "2".

Literals and Expressions

```
1 + 1
```

- We say that `1` is a **literal**
- `1 + 1` is an **expression** combining two literals with the **operation** `+`

2nd Example

How many letters are in the text "Hello, world!"?

2nd Example

How many letters are in the text "Hello, world!"?

```
"Hello, world!".length
```

- Here, `"Hello, world!"` is a literal
- We apply the operation `length` to it

Puzzle

What is the result of the following program?

```
"length".length
```

O 6

O Error

Puzzle

What is the result of the following program?

```
"length".length
```

- 0
- 6
- Error

Text literals are distinguished from **names** by the enclosing double quotes.

More Examples of Operations

Program	Result
`1 > 0`	`true`
`1 == 0`	`false`
`1.max(0)`	`1`
`-5.abs`	`5`
`"Hello, " + "world!"`	`"Hello, world!"`
`"#" * 3`	`"###"`
`"Alice".toUpperCase`	`"ALICE"`
`true && true`	`true`
`true.&&(true)`	`true`

More Examples of Operations

Program	Result
`1 > 0`	`true`
`1 == 0`	`false`
`1.max(0)`	`1`
`-5.abs`	`5`
`"Hello, " + "world!"`	`"Hello, world!"`
`"#" * 3`	`"###"`
`"Alice".toUpperCase`	`"ALICE"`
`true && true`	`true`
`true.&&(true)`	`true`

Operations can be applied to values by using the **dot-notation** or by using the **infix syntax**.

Arithmetic Operators

- Arithmetic operators have the same **precedence** as in mathematics. The following expressions are equivalent:

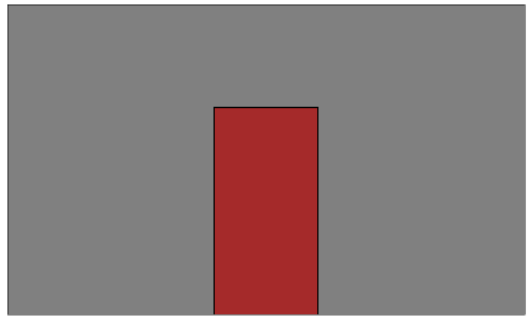
```
1 + 2 * 3
1.+(2.*(3))
```

- The result of arithmetic operations has the type of its widest operand:

```
1 + 2   // Int
1 + 2.0 // Double
```

3rd Example

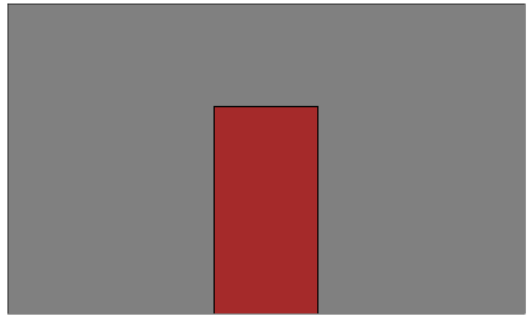
Our house is currently gray. We want to paint it in yellow.



The house is 5 meters wide by 3 meters height, and the (brown) door is 1 meter wide by 2 meters height.
How much surface do we need to paint?

3rd Example

Our house is currently gray. We want to paint it in yellow.



The house is 5 meters wide by 3 meters height, and the (brown) door is 1 meter wide by 2 meters height.
How much surface do we need to paint?

Program	Result
5 * 3	15
1 * 2	2
5 * 3 - 1 * 2	13

Definitions

Large expressions are hard to read and write.

We can **give names** to fragments of expressions and then refer to them by using these names:

```
val facade = 5 * 3
val door   = 1 * 2
facade - door
```

Definitions

Large expressions are hard to read and write.

We can **give names** to fragments of expressions and then refer to them by using these names:

```
val facade = 5 * 3
val door   = 1 * 2
facade - door
```

Names are introduced with the `val` keyword.

Reuse Definitions

Let us say that the house also has two windows (1 meter by 1 meter each).

```
val facade = 5 * 3
val door   = 1 * 2
val window = 1 * 1
facade - door - window - window
```

Naming an expression makes it easy to **reuse** it multiple times.

Summary

Programs *express* computations.

Programs are made of *values* combined together with *operations*.

Intermediate results can be *named* to be easily reused.