Kotlin Program

By: Héba Bouzidi & Fayssal El-Ansari

For: Software Engineering (M1 Computing Science)

Project Description

In this project we will try to build a meta model for the famous programming language Kotlin. The tasks were not precisely cut, we both touched and modified the model and the Xtext grammar.

General Notes

- In each step of the meta-model development, we will try to reproduce a code snippet with increasing complexity. In a way that reflects the logic behind the Kotlin language. While avoiding to build unnecessary entities and reusing entities when possible, and using the right type of relationship between them. Our goal is to let the user generate a syntactically **correct** code from a built Kotlin program model.
- The order of execution of steps is the same as the order of adding children to an entity.
- This file's examples and our objectives were made by following the official Kotlin guide: https://kotlinlang.org/docs/basic-syntax.html

Disclaimer

- Eclipse Modeling is required, as the project has been set up according to it.
- A way to try how rightful is the code you will make in Testing.kt, here is an online compiler https://play.kotlinlang.org/

Meta Model Features

All the planned features, those that are checked have been made.

- ☒ Package definition and imports

```
// This is an end-of-line comment

/* This is a block comment
   on multiple lines. */

/* The comment starts here
/* contains a nested comment */
and ends here. */
```

- ☐ Package definition and imports

```
package my.demo

import kotlin.text.*

// ...
```

- ☒ Variable declaration

```
val a: Int = 1  // immediate assignment
```

- ☐ More ways to use variables

```
val b = 2   // `Int` type is inferred
val c: Int  // Type required when no initializer is provided
c = 3       // deferred assignment

var x = 5 // `Int` type is inferred
x += 1

val PI = 3.14
var x = 0

fun incrementX() {
    x += 1
}
```

- ☒ Print to the standard output

```
print("Hello ")
print("world!")

println("Hello world!")
```

- ☐ Print content that are not String

```
print(42)
```

- ☒ Functions

```
fun mult(a: Int, b: Int): Int {
    return a * b
}

fun printSum(a: Int, b: Int): Unit {
    println("sum of $a and $b is ${a + b}")
}

fun printSum(a: Int, b: Int) {
    println("sum of $a and $b is ${a + b}")
}
```

- ☐ Other way to write a function

```
fun sum(a: Int, b: Int) = a + b
```

- ☒ Program Entry Point (main function)

```
fun main() {
    println("Hello world!")
}
```

- ☐ Program Entry Point (main functin)

```
fun main(args: Array<String>) {
    println(args.contentToString())
}
```

- ☒ Creating classes and instances

```
class Rectangle(var height: Double, var length: Double) {
}

val rectangle = Rectangle("5.0", "2.0") //this is known as incorrect
println("The perimeter is ${rectangle.perimeter}")
```

- ☐ Creating classes and instances

```
class Shape

val rectangle = Rectangle(5.0, 2.0)

open class Shape

class Rectangle(var height: Double, var length: Double): Shape() {
    var perimeter = (height + length) * 2
}
```

Usage

To use this project:

1. clone the repository in a directory of your choice
2. open eclipse modeling
3. set the current workspace to the *`Kotlin - Model Files`* folder
4. edit!

Debugging

One of the commun problems while importing a project is due to not regenerating: *`src-gen`*, *`*.edit`* and *`*.editor`* folder. So if there are some problems in the project after import, try deleting (from the hard disk as well) before regenerating all the files from the *`*.genmodel`* file.

Another could be due to a `wrong import`, if there are multiple projects in the imported project, select the individual project one by one and deselect the parent project, to avoid reconfiguring the `build path`.

Lastly: The default eclipse runtime workspace is used.

Enjoy trying out this project!