Programming Fundamentals 2

Pierre Talbot

16 February 2021

University of Luxembourg

Chapter I. Basics of Java Syntax

Syntax vs Semantics

- The **syntax** of a programming language defines the set of symbols allowed in the program, and its structure.
- The **semantics** of a programming language gives meaning to the sentences.

Examples

- Syntactically incorrect: "The eert is high" (unknown symbol "eert").
- Syntactically incorrect: "The tree is is high" (bad structure: repetition of "is").
- Syntactically correct but semantically incorrect: "The tree is reading a glass of water".

Syntax vs Semantics

It is similar with computer programs.

Examples

- Syntactically incorrect: `tni i = 1;` (unknown symbol "tni").
- Syntactically incorrect: `int i 1` (missing equility symbol).
- Syntactically correct but semantically incorrect: `int i = "a";` (expected type `int`, got `String`).
- Syntactically correct but semantically incorrect: `int i = 1; int i = 2;` (`i` redeclared).

The differences will be made precise in the class Programming Languages (BAINFOR-53).

Java Syntax

A language is a mix of various syntactic components such as:

- Statements
- Expressions
- Types
- Literals

When learning a language, we often look at examples, but this is not a formal nor complete specification of a language.
Therefore, we need a formalism to describe syntax: context-free grammar.

The syntax and informal semantics of Java is described in the *Java SE specification*:

`https://docs.oracle.com/javase/specs/jls/se15/html/index.html`

Statements (§14)

A **statement** is a construct that produces side-effect (e.g., it modifies the value of a variable, prints on the screen, ...).

```
Statement:
  Block
  LocalVariableDeclarationStatement  int x = 1;
                                     Integer o = new Integer(3);
  Assignment                         x = 3;
  IfThenStatement
  IfThenElseStatement
  WhileStatement
  ForStatement
  ClassDeclaration
Block:
  { [Statement] }                    int x = 1; x = x + 1;
IfThenStatement:
  if ( Expression ) Statement        if(x < 4) x = 2;
IfThenElseStatement:
  if ( Expression ) Statement else Statement
WhileStatement:
  while ( Expression ) Statement
ForStatement:
  for ( [ForInit] ; [Expression] ; [ForUpdate] ) Statement  for(int i = 0; i < n; i++) {
                                                                System.out.println(i);
                                                            }
ClassDeclaration:
  {ClassModifier} class TypeIdentifier [TypeParameters] [Superclass] [Superinterfaces] ClassBody
                                  public class Rectangle {
                                      private int width;
                                      private int height;
                                      ...
                                  }
```

Dangling `else` problem

Unlike Python, indentation is not mandatory in Java (that being said, you should indent as in Python).
This can lead to some problems with `if-else` statements (in C and C++ as well):

```
if (x > 0)
  if (y < 0)
    y = 2;
else
  x = 1;
```

The last `else` statement actually belong to the innermost `if`, here `if(y < 0)`.

To avoid ambiguity, always use curly braces:

```
if (x > 0) {
  if (y < 0) {
    y = 2;
  }
}
else {
  x = 1;
}
```

Expression (§15)

An **expression** is a code that evaluates to a value.

- Variable name: `i`, `x`, `average`.
- Array access: `arr[i]`, `matrix[i][j]`.
- Arithmetic expression: `7 + 8`, `x / 8 + 2 * 4`.
- Function call: `fibonacci(8)`.
- ...

Basically, if you can write `x = E;`, then `E` is an expression.

We will complete this list as we progress.

Types (§4)

Java is a **statically typed language**: a variable `x` has an *explicit* and *single* type during the execution of the program.

```
Type:
  PrimitiveType
  ReferenceType
PrimitiveType:
  (one of)
  boolean float double byte short int long char
ReferenceType:
  (see §4.3)        String, java.util.Scanner, ArrayList<Integer>
```

Arrays (§10)

1D array:

```
int n = 10;
int[] grades = new int[n]; // Create an array of size ‘n‘, all elements are initialized to 0.
// ... Populate the array with grades (not shown)
int sum = 0;
// grades.length is an attribute of array giving the size of the array (here equals to ‘n‘).
for(int i = 0; i < grades.length; ++i) {
  sum += grades[i];
}
System.out.println("The average of the student is "
  + (sum / grades.length));
```

2D array:

```
int n = 10;
int m = 29;
int[][] matrix = new int[n][m]; // Create a 2D array of size ‘n * m‘, all elements are initialized to 0.
matrix[2][0] = 10; // Initialize the elements at coordinate (2,0) to 10.
```

Literals (§3.10)

Literals are the possible ground values in the language:

```
Literal:
  IntegerLiteral              2, 0, -1
  FloatingPointLiteral        1.1, 1.1f, 2., 2.9e-3
  BooleanLiteral              true, false
  CharacterLiteral            'a' '\u0370'
  StringLiteral               "hello"
  TextBlock                   """ a very long multi-line string"""
  NullLiteral                 null
```

The set of literals is different according to the language, *e.g.*, in Python you have a literal for complex number (`3.14j`).

Unicode

Unicode is a standard to represent characters in a unified way.

`www.youtube.com/watch?v=-n2nlPHEMG8`

- Characters from more than 200 languages, but also emojis, are represented by a unique *code point*.
- For instance, `a` has the code point `U+0061`, and $\sum$ has `U+2211`.
- Code point can be encoded as:
  1. UTF-8: smaller string size, but linear array access (*e.g.*, no s[10]), because a character can occupies 1, 2, 3 or 4 bytes.
  2. UTF-16: 2 bytes per character, but constant array access.
- The Java `String` class uses UTF-16. You can write code points as `"\u1F602"`, which are automatically transformed into UTF-16.

Reading of the week: https://www.joelonsoftware.com/2003/10/08/the-absolute-minimum-every-software-developer-absolutely-positively-must-know-about-unicode-and-character-sets-no-excuses/

Floating-point numbers

The floating-point number `1.1` does not exist

But, sir, `System.out.println(1.1f)` is printing `1.1`! Well, kid, **it's a lie**! With enough precision `System.out.printf("%.10f", 1.1f);` will print `1.1000000238`.

Floating-point number are not exact!

Give you a treat, read this paper before you graduate:

What Every Computer Scientist Should Know About Floating-Point Arithmetic, David Goldberg, 1991

`http://pages.cs.wisc.edu/~david/courses/cs552/S12/handouts/goldberg-floating-point.pdf`

The (almost) smallest Java program

In order to execute some Java code, you absolutely need a `main` function. It indicates where the program actually starts.

```
public class Chess {
  public static void main(String[] args) {
    System.out.println("Welcome to my Chess program");
  }
}
```

The file **must** have the same name as the class, here `Chess.java`.

Input/Output in 2 minutes

```
import java.util.Scanner;

public class HelloWorld {
  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    System.out.print("What's your name? ");
    String name = scanner.nextLine();
    System.out.print("What's your age? ");
    int age = scanner.nextInt();
    System.out.println("Welcome " + name + " (" + age
      + "years' old)");
    scanner.close();
  }
}
```

We concatenate String with the operator +.

It works with literals and variables with primitive types as well.

Summary

We took a glimpse to some basic Java constructs.
You need nothing more to start coding your first Java programs :-)

Homework

- **Laboratory 1**, already available on Moodle.
- **Reading of the week**: `https://www.joelonsoftware.com/2003/10/08/the-absolute-minimum-every-software-developer-absolutely-positively-must-know-about-unicode-and-character-sets-no-ex`
- **Coding event** (optional)
  1. Register Google Hash Code (`hashcodejudge.withgoogle.com`).
  2. Give me the name of your team here: `https://docs.google.com/spreadsheets/d/1zSi6PG32kPGu5Kxhye19vsD7fqb2kqYVIntkh6Xd9fc/edit?usp=sharing`