Supplementary Notes on Java Grammar and Syntax

Rachel Dowavic

August 3, 2017

This document covers .java file structure, variable assignment, boolean statements, if statements, switch statements, while loops, do-while loops, for loops, enhanced for loops, methods, and declaring arguments of a method.

`.java` file structure

Every *executable* java file called `X.java` **must** conform to the following structure:

```
import java.util.SomethingINeed; //optional
...

public class X {
   ...
   public static void main(String[] args) {

   }
   ...
}
```

Variable assignment

Let `z` be a primitive data type. (`int, float, char, ...`).

```
z myVariable = zLiteral;
z myOtherVariable = MethodThatReturnsAz();
```

Examples of int, float, char, ... literals are provided in this wiki document.
https://en.wikibooks.org/wiki/Java_Programming/Literals

Let `Z` be an Object type. (`Scanner, ArrayList<Integer>, ...`).

```
Z myVariable = new Z(..);
Z myOtherVariable = myVariable;
Z myThirdVariable = MethodThatReturnsAZ();
```

Boolean statement

Java allows you to use boolean algebra operators to form expressions.
Visit http://introcs.cs.princeton.edu/java/71boolean/ for the specific java symbols.

```
boolean itIsHot = temperature > 40;
boolean youAreOld = age >= 100;
```

If statement

Let `b` be a boolean statement.

```
if (b) {
   ....
} else {
   ....
}
```

Note that `if (b)` is equivalent to `if (b == true)`, and `if (!b)` is equivalent to `if (b == false)`. An alternative is the ternary operator. Let `b` be a boolean statement, and `c, d` be a data type matching the return type of this method we can pretend we're in.

```
...
return b ? c : d;
```

means `return (is b true?  if so, c :  otherwise, d);`

Switch statement

Let `a` be some primitive, enumerated, or String type in `int, String, char, ...`

```
...
a userInput = ...;
...
switch (userInput) {
    case aLiteral1: doSomething();
                    break;
    case aLiteral2: doSomethingElse();
                    break;
    ....
    case aLiteralN: doAnotherThing();
                    break;
    default:        break;
}
```

While loop

```
while (b) {
  .....
}
```

Do-while loop

```
do {
   ....
} while (b);
```

For loop

```
for (variable assignment; boolean statement; action taken after each iteration) {
    ....
}

//example

for (int i = 0; i < 6; i++) {
    .....
}
```

Note that a `for loop` even with all fields of `for(..;..;..)` left empty is still syntactically correct and will compile.

Enhanced For loop

While you can write

```
for (int i = 0; i < array.length; i++) {
//do something with array[i]
}
```

to iterate over all elements of an array, you can also write

```
for (int element: array) {
    //do something with element
}
```

Read this for loop as:
"for each `element` of type `int` in the iterable variable `array`, do something with the given `element`."

Method

Called a function in many other programming languages.

```
privacySetting (static || nothing) returnType methodName(a myA, b myB) {
    ....
}
```

The privacy modifier is covered well in this stack overflow question:
https://stackoverflow.com/questions/215497/in-java-difference-between-default-public-protected-and-private

Also, the following website provides excellent coverage of the Java method definition:
https://docs.oracle.com/javase/tutorial/java/javaOO/methods.html

Declaring the arguments of your method

Let `a1, ... an` be any types, and `x1, .. xn` be identifiers, in other words, variable names. Variable names must be unique. A method called methodName with n parameters is declared in the following way:

```
public static returnType methodName(a1 x1, a2 x2, ..., an xn) {
   ....
}
```