Syntax and Semantics

Syntax – describes what constructions are possible in the language, what is a correct program and what is not

Semantics – describes what these constructions mean, e.g., what computer does when it performs given commands

Java distinguishes between two types of errors:

Compile time errors: They are produced by compiler. These errors are either

- **syntax** – for example missing ;
- **semantic** – for example assignment between incompatible types

Run time errors: They are produced by Java Virtual Machine during execution of a program. Java contains no dangerous constructions, always an **exception** is generated.

Overview of the Syntax of Java

- The programs consist of **classes**.
- Each class consists of definitions of data and instructions for a certain kind of **objects**.
  - **fields (attributes):** data of an object
  - **methods:** instructions that manipulate with these data (also called **functions** or **procedures** in other languages)
- A method consists of a **header** that defines its name, arguments, return type, ... and **body** that contains **statements**.
- Statements manipulate with **data** stored in **variables** – either in **fields** or in **local variables**.
- Execution of statements includes evaluation of **expressions**. The values of expressions are then assigned into variables.
- Each variable, value and expression is of some **type**.
- On the lowest level a program is a sequence of **lexical elements (tokens)**.

Lexical Elements

White space characters and comments are ignored:

- **white space characters**: space (SP), horizontal tab (HT), form feed (FF), new line (LF), carriage return (CR)
- **comments**: `/* this is a comment */`

Basic types of lexical elements (tokens) are:

- **identifiers**: `x` `dist1` `System9` `number_of_elements`
- **keywords**: **`while`** **`float`** **`int`** **`public`** **`class`**
- **literals**: `124` **`true`** `'d'` `"hello"`
- **separators**: `(` `)` `{` `}` `[` `]` `;` `:` `,` `.`
- **operators**: `+` `-` `*` `/` `&&` `=` `*=` `<` `>>=`

Literals

- **integer literals:**
  `0` `237L` `033` `0xDadaCafe` `1996` `0x00FF00FF`
- **floating point literals:**
  `1e1` `2.` `.3` `0.0` `3.14f` `1.213e-9` `1E137D`
- **boolean literals:**
  `true` **`false`**
- **character literals:**
  `'a'` `'%'` `'\t'` `'\\'` `'\''` `'\177'` `'\u03a9'`
- **string literals:**
  `""` `"\""` `"This is a string."` `"\r\n"`
- **the null literal:**
  `null`

Possible **escape sequences** in character and string literals:

- `\b` `\t` `\n` `\f` `\r` `\"` `\'` `\\` `\177` `\u2B97`

Keywords

abstract	for	strictfp
boolean	goto	super
break	if	switch
byte	implements	synchronized
case	import	this
catch	instanceof	throw
char	int	throws
class	interface	transient
const	long	try
continue	native	void
default	new	volatile
do	package	while
double	private	
else	protected	
extends	public	
final	return	
finally	short	
float	static	

Comments

Java supports three kinds of comments:

- **One-line comment** – the compiler ignores everything from the // to the end of line.

```
// This is a one-line comment.
```

- **Multi-line comment** – the compiler ignores everything from the /* to an occurrence of */ .
  Note: /*/ is not a valid comment.

```
/* This is a comment that
   continues across lines. */
```

- **Documentation comment** – the compiler ignores everything from the /** to an occurrence of */ . javadoc tool generates documentation based on content of the comment.

```
/** This is a documentation comment.
 *  The comment may contain html tags as well as special
 *  tags that begin with the '@' sign. */
```

Types, Values and Variables

Variables are used by program to hold data. Each variable used in program must be explicitly specified by its **data type** and **name**. Java has two kinds of data types: **reference** and **primitive**.

- **Primitive**

  A variable of primitive type contains a **single value** of the appropriate size and format for its type: a number, a character or a boolean value.

```
boolean b = true;
int i = 456;
float f = 2.71828;
```

- **Reference**

  The value of a reference type variable, in contrast to that of a primitive type, is a **reference** to (an address of) an object or an array.

```
Hashtable h = new Hashtable();
int[] a = new int[20];
```

Integral Types and Values

Type	Range	Size [bits]
byte	-128..127	8
short	-32768..32767	16
int	-2147483648..2147483647	32
long	-9223372036854775808..9223372036854775807	64
char	0..65535	16

Possible operations on integer values are:

- the comparison operators (<, <=, >, >=, ==, !=)
- the unary plus and minus (+, -)
- the binary arithmetic operators (+, -, *, /, %)
- the prefix and postfix increment and decrement operators (++, --)
- the signed and unsigned shift operators (<<, >>, >>>)
- the bitwise complement operator (~)
- the integer bitwise operators (&, |, ^)

Floating- Point Types and Values

The floating-point values are numbers of the form $sm2^e$ where

Type	s	m	e	Size [bits]
float	-1,1	$0..2^{24}-1$	-149..104	32
double	-1,1	$0..2^{53}-1$	-1075..970	64

Type	Min. value	Max. value
float	1.40239846e-45f	3.40282347e+38f
double	4.94065645841246544e-324	1.79769313486231570e+308

Possible operations on floating-point values are:

- the comparison operators (<, <=, >, >=, ==, !=)
- the unary plus and minus (+, -)
- the binary arithmetic operators (+, -, *, /, %)
- the prefix and postfix increment and decrement operators (++, --)

The Boolean Type and Values

The type **boolean** has two possible values: **true** and **false**

Possible operations on floating-point values are:

- the relational operators (==, !=)
- the logical complement operator (!)
- the binary logical operators (&, |, ^)
- the conditional-and and conditional-or operators (&&, ||)
- the ternary conditional operator (?:)

Boolean expressions determine the control flow in several kinds of statements:

- the **if** statement
- the **while** statement
- the **do** statement
- the **for** statement