


EECS1022
MOBILE COMPUTING



YONKERS UNIVERSITY

JAVA BASICS

LEXICAL ELEMENTS

(SLIDES ADAPTED FROM PROF.H. ROUMANI)

PROF. Y. LESPÉRANCE
Dept. of Electrical Engineering & Computer Science

1

```
import java.lang.System;  
public class Area  
{  
    public static void main(String[] args)  
    {  
        int width;  
        width = 8;  
        int height = 3;  
        int area = width * height;  
        System.out.println(area);  
    }  
}
```

2

```
import java.lang.System;  
public class Area  
{  
    public static void main(String[] args)  
    {  
        int width;  
        width = 8;  
        int height = 3;  
        int area = width * height;  
        System.out.println(area);  
    }  
}
```

Imports

3

```
import java.lang.System;  
public class Area  
{  
    public static void main(String[] args)  
    {  
        int width;  
        width = 8;  
        int height = 3;  
        int area = width * height;  
        System.out.println(area);  
    }  
}
```

Imported Class
= Delegation

4

```
import java.lang.System;
public class Area
{
    public static void main(String[] args)
    {
        int width;
        width = 8;
        int height = 3;
        int area = width * height;
        System.out.println(area);
    }
}
```

Class Header

Class Body, a Block

5

```
import java.lang.*;
public class Area
{
    public static void main(String[] args)
    {
        int width;
        width = 8;
        int height = 3;
        int area = width * height;
        System.out.println(area);
    }
}
```

Method Header

Method Body, a Block

6

Style

- Class naming convention**
A noun. Use Pascal/Title case, e.g. Math, ArrayList.
- Method naming convention**
A verb. Use camel case, e.g. equals, toString, isLeapYear
- Variable naming convention**
A noun. Use camel case, e.g. length, interestRate, gender
Applies also to attributes and parameters.
- Block layout**
Braces must align vertically and the all statements must be left justified and indented by one tab position.

7

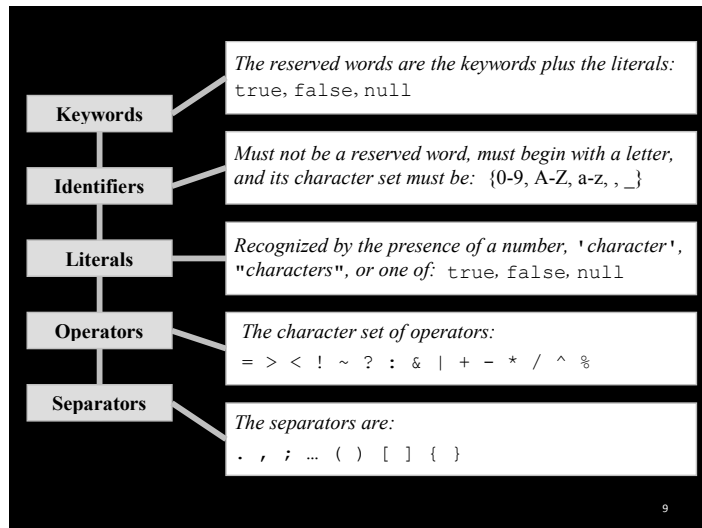
Lexical Elements

Without worrying about syntax or semantics, let us identify the elements of a program:

- Keywords
- Identifiers
- Literals
- Operators
- Separators

8

Lexicon



Keywords

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

Example

Identify the **language elements** in the following program...

Keywords, Identifiers, Literals, Operators, Separators

```
import java.lang.System;

public class Area
{
    public static void main(String[] args)
    {
        int width;
        width = 8;
        int height = 3;
        int area = width * height;
        System.out.println(area);
    }
}
```

Keywords, Identifiers, Literals, Operators, Separators

```
import java.lang.System;
public class Area
{
    public static void main (String[] args)
    {
        int width;
        width = 8;
        int height = 3;
        int area = width * height;
        System.out.println(area);
    }
}
```

Keywords, Identifiers, Literals, Operators, Separators

13

```
import java.lang.System;
public class Area
{
    public static void main (String[] args)
    {
        int width;
        width = 8;
        int height = 3;
        int area = width * height;
        System.out.println(area);
    }
}
```

Keywords, Identifiers, Literals, Operators, Separators

14

Compile Time vs Run Time Errors

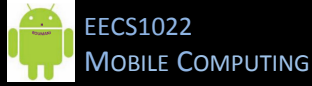

- Before program can run, it must be **compiled** to (translated) Java bytecode
- Studio does this as you enter/edit your code; it flags **compile-time errors**:
 - **Syntax errors**, e.g. missing ; { (
 - **Type errors**, e.g. "abc" * 3

15

Compile Time vs Run Time Errors

- When you execute your program, you may get **runtime errors**:
 - **ArithmeticException**, e.g. 10 / 0
 - **ArrayIndexOutOfBoundsException**, etc.
- **Logic errors**: program appears to run normally but does not behave as required

16

JAVA BASICS
DECLARATION
 (SLIDES ADAPTED FROM PROF.H. ROUMANI)

PROF. Y. LESPÉRANCE
 Dept. of Electrical Engineering & Computer Science

1

The Declaration Statement

type name ;

↙

The name of a primitive or non-primitive type, e.g. int, double...

↓

An identifier to be associated with a memory block

↘

A separator

- The *scope* of the variable = the enclosing block of the declaration.
- The variable is not known outside its scope.
- Declaration does not initialize. Not with 0 or null or anything else.

2

Primitive & Non-Primitive

Primitive

Non-Primitive

- number
- character
- boolean
- class
- interface
- array

3

NUMERIC TYPES

Integer	↳	int	4	±2G	exact
	↳	long	8	±2E	exact
Integer literals are int by default unless suffixed with L					
Real	↳	float	4	±10 ³⁸	SD=7
	↳	double	8	±10 ³⁰⁸	SD=15
Real literals are recognized by a decimal point or an exponent. They are double by default unless suffixed with F. For exponential notation, use E.					

4

INTEGER OR REAL?

Integer ••• Use for integer data, e.g. count. *100% exact*

Real ••• Use for real data, e.g. amount. *Inherently inaccurate*

5

The Type `boolean`

- Stores the result on a condition
- Has only two possible values
- `true` and `false` are reserved words
- Boolean variables are not integers
- The Boolean operators are: `&&` (for and), `||` (for or), and `^` (for xor)

Note: Boolean literals are the easiest to recognize!

6

The Character Type `char`

- A letter, digit, or symbol
- Digits versus Numbers
- Store the code, not the typeface
- The case of English: ASCII vs UniCode
- `char` is thus an (unsigned) integer type
- No `char` operators! They auto-promote to `int`.

Character literals are recognized by single quotes surrounding the character, e.g. 'A'

7

More on Characters

Code	Character
0	
⋮	
32	space
⋮	
48-57	'0'-'9'
⋮	
65-90	'A'-'Z'
⋮	
97-122	'a'-'z'
⋮	
65535	

Escape	Meaning
<code>\uxxxx</code>	The character whose code is (hex) <code>xxxx</code>
<code>\'</code>	Single quote
<code>\"</code>	Double quote
<code>\\</code>	Backslash
<code>\n</code>	New line
<code>\r</code>	Carriage return
<code>\f</code>	Form Feed
<code>\t</code>	Tab
<code>\b</code>	Backspace

8

Declaration

PRIMITIVE TYPES		Type	Size (bytes)	Approximate Range		S.D.
		min		max		
NUMERIC	SIGNED	byte	1	-128	+127	N/A
		short	2	-32,768	+32,767	N/A
		int	4	-2×10^9	$+2 \times 10^9$	N/A
		long	8	-9×10^{18}	$+9 \times 10^{18}$	N/A
	UNSIGNED	char	2	0	65,535	N/A
REAL	SINGLE	float	4	$+3.4 \times 10^{38}$	$+3.4 \times 10^{38}$	7
	DOUBLE	double	8	-1.7×10^{308}	$+1.7 \times 10^{308}$	15
BOOLEAN		boolean	1	true/false		N/A

Class Type `String` (in java.lang)

- Stores a sequence of characters
- Optimized for speed → immutable
- Optimized declaration → shortcut
- Optimized concatenation → + operator
- Rich API (e.g. `indexOf`, `charAt`, `substring`)

Note: String literals are surrounded with double quotes and can use the same escape sequences as chars.

10

Class Type `Date` (in java.util)

- Stores an instance of time
- Captures both date and time
- Accurate to a millisecond
- Simple API (`toString` and `getTime`)

Note: Like all class types (except for String), Date has no literals and no operators.

11

Class Type `RectangleModel` (in our package)

- Stores an instance of a rectangle
- Captures the height and width as int
- API (`getArea` and `getCircumference`)

Like all class types (except for String), it has no literals and no operators.

12

Class Type `FractionModel`

(in our package)

- Stores an instance of a fraction
- Numerator and denominator are long
- API (add, sub, multiply, divide, ...)

Like all class types (except for String), it has no literals and no operators.

13

Class Type `TextView`

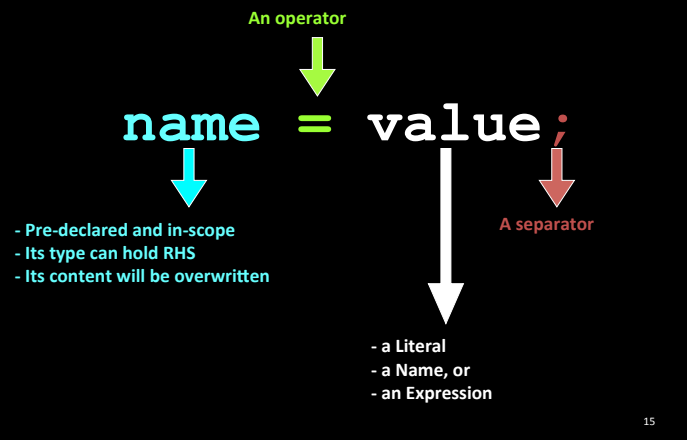
(in android.widget)

- Stores a UI label
- Many attributes: text, layout, style, ...
- API (getText, setText, setTypeFace, ...)

Like all class types (except for String), it has no literals and no operators.

14

THE ASSIGNMENT STATEMENT



15