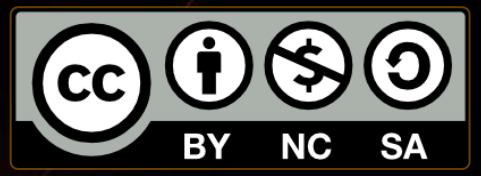
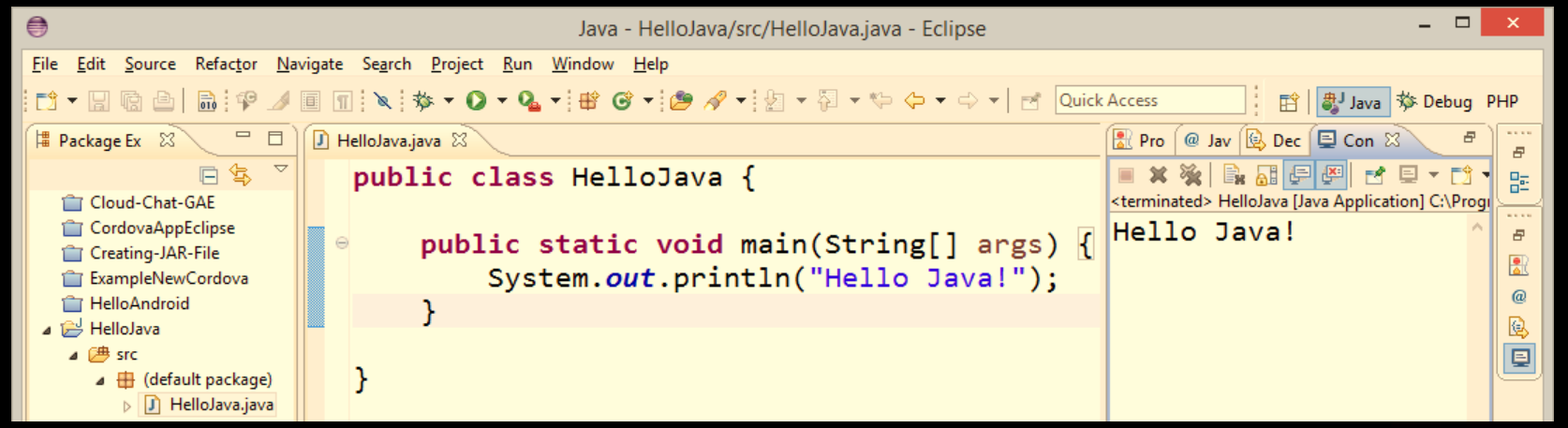
Java Syntax

Data Types, Variables, Operators, Expressions, Statements, Console I/O, Conditional Statements



Bogomil Dimitrov
Technical Trainer

Software University
http://softuni.bg

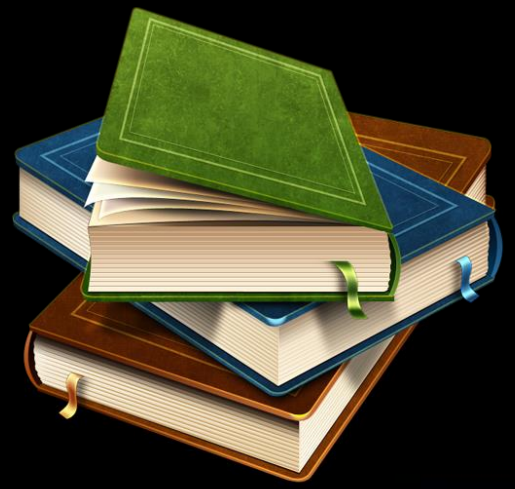


SoftUni Diamond Partners

SOFTWARE UNIVERSITY FOUNDATION

Table of Contents

1. Primitive Data Types
2. Variables
3. Operators
4. Expressions
5. Statements
6. Console-Based Input and Output
7. Conditional Statements
   - if-else, switch-case, …



Warning: Not for Absolute Beginners

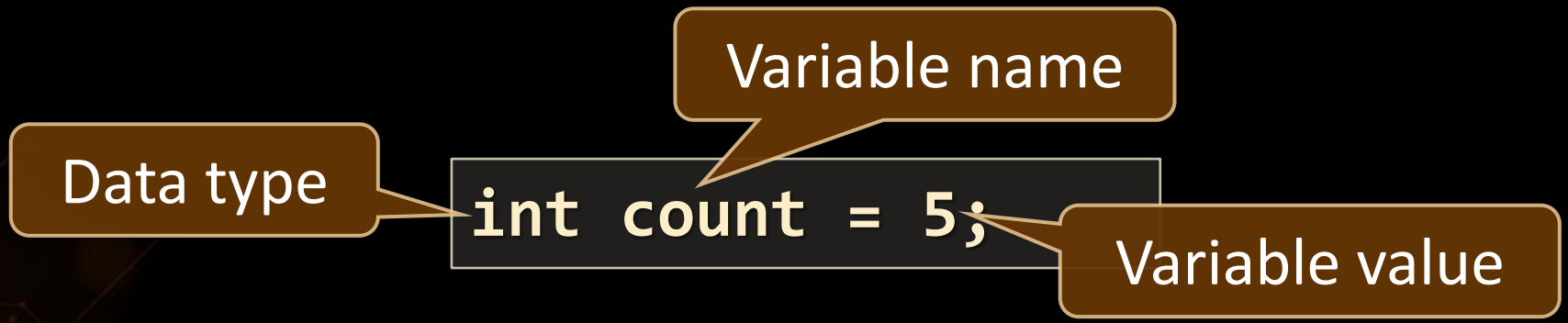
- The "**Java Basics**" course is NOT for absolute beginners
  - Take the "C# Basics" course at SoftUni first: https://softuni.bg/courses/csharp-basics
  - The course is for beginners, but with previous coding skills
- Requirements
  - Coding skills – entry level
  - Computer English – entry level
  - Logical thinking

SOFTWARE UNIVERSITY FOUNDATION

Primitive Data Types in Java

How Computing Works?

- Computers are machines that process data
  - Data is stored in the computer memory in variables
  - Variables have name, data type and value
- Example of variable definition and assignment in Java



- When processed, data is stored back into variables

What Is a Data Type?

- A data type:
  - Is a domain of values of similar characteristics
  - Defines the type of information stored in the computer memory (in a variable)

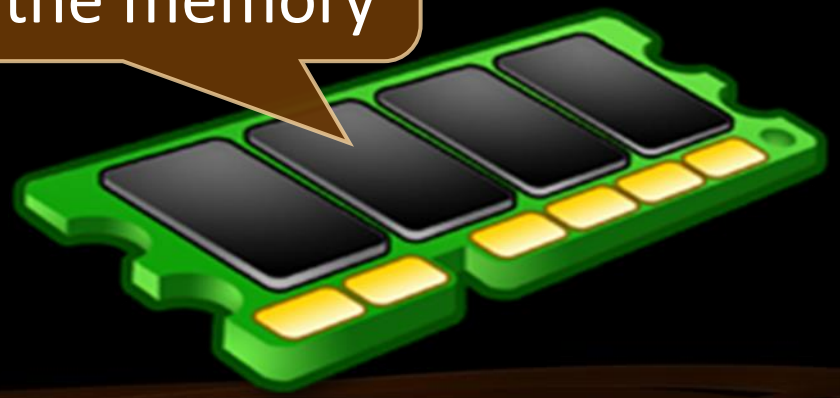
- Examples:
  - Positive integers: `1`, `2`, `3`, …
  - Alphabetical characters: `a`, `b`, `c`, …
  - Days of week: `Monday`, `Tuesday`, …

Data Type Characteristics

- A data type has:
  - Name (Java keyword, e.g. **`int`**)
  - Size (how much memory is used)
  - Default value
- Example:
  - Integer numbers in C#
  - Name: **`int`**
  - Size: **`32 bits`** (4 bytes)
  - Default value: **`0`**

`int`: sequence of 32 bits in the memory

`int`: 4 sequential bytes in the memory



Integer Types

- `byte` (-128 to 127): signed 8-bit
- `short` (-32,768 to 32,767): signed 16-bit
- `int` (-2,147,483,648 to 2,147,483,647): signed 32-bit
- `long` (-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807): signed 64-bit

```
byte b = 1;
int i = 5;
long num = 3L;
short sum = (short) (b + i + num);
```

SOFTWARE UNIVERSITY FOUNDATION

Integer Types

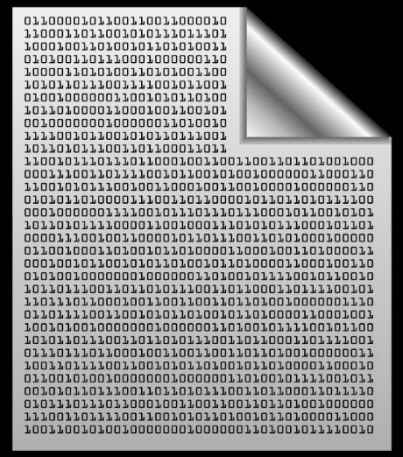
Live Demo

int

byte

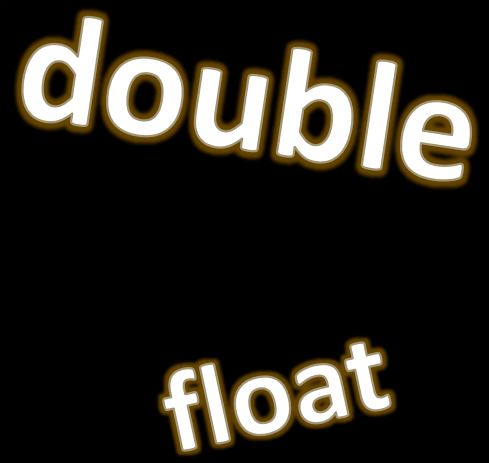
long

short



Floating-Point Types

- Floating-point types are:
  - `float` ($\pm 1.5 \times 10^{-45}$ to $\pm 3.4 \times 10^{38}$)
    - 32-bits, precision of 7 digits
  - `double` ($\pm 5.0 \times 10^{-324}$ to $\pm 1.7 \times 10^{308}$)
    - 64-bits, precision of 15-16 digits
- The default value of floating-point types:
  - Is `0.0F` for the `float` type
  - Is `0.0D` for the `double` type



Floating-Point Types – Examples


```
float f = 0.33f;
double d = 1.67;
double sum = f + d;
float fSum = f + d; // This will not compile
double infinity = 3.14 / 0;

System.out.println(f); // 0.33
System.out.println(d); // 1.67
System.out.println(sum); // 2.0000000013113022
System.out.println(infinity); // Infinity
```

BigDecimal

- The floating-point arithmetic sometime works incorrectly
  - Don't use **`float`** and **`double`** for financial calculations!
- In Java use the **`BigDecimal`** class for financial calculations:

```
import java.math.BigDecimal;
…
BigDecimal bigF = new BigDecimal("0.33");
BigDecimal bigD = new BigDecimal("1.67");
BigDecimal bigSum = bigF.add(bigD);
System.out.println(bigSum); // 2.00
```

SOFTWARE UNIVERSITY FOUNDATION

Floating-Point and BigDecimal Types

Live Demo

Other Primitive Data Types

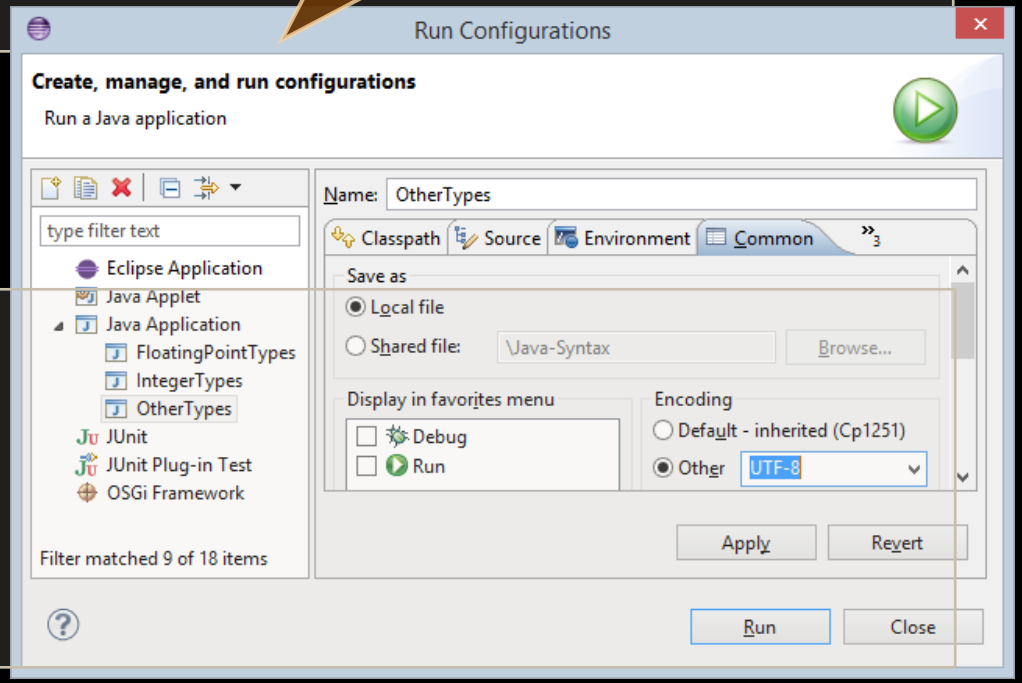
- Boolean

```
boolean b = true;
System.out.println(b); // true
System.out.println(!b); // false
```

- Character

```
ch = 'Ю';
System.out.println(ch);
ch = '\u03A9'; \\ Ω
System.out.println(ch);
```

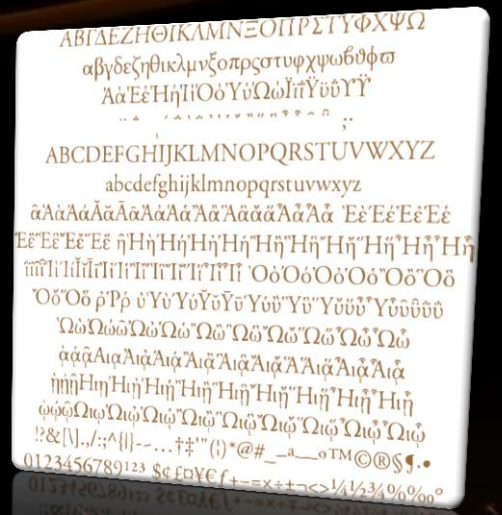
Enable Unicode in the Eclipse console



SOFTWARE UNIVERSITY FOUNDATION

Boolean and Character Types

Live Demo



true

false

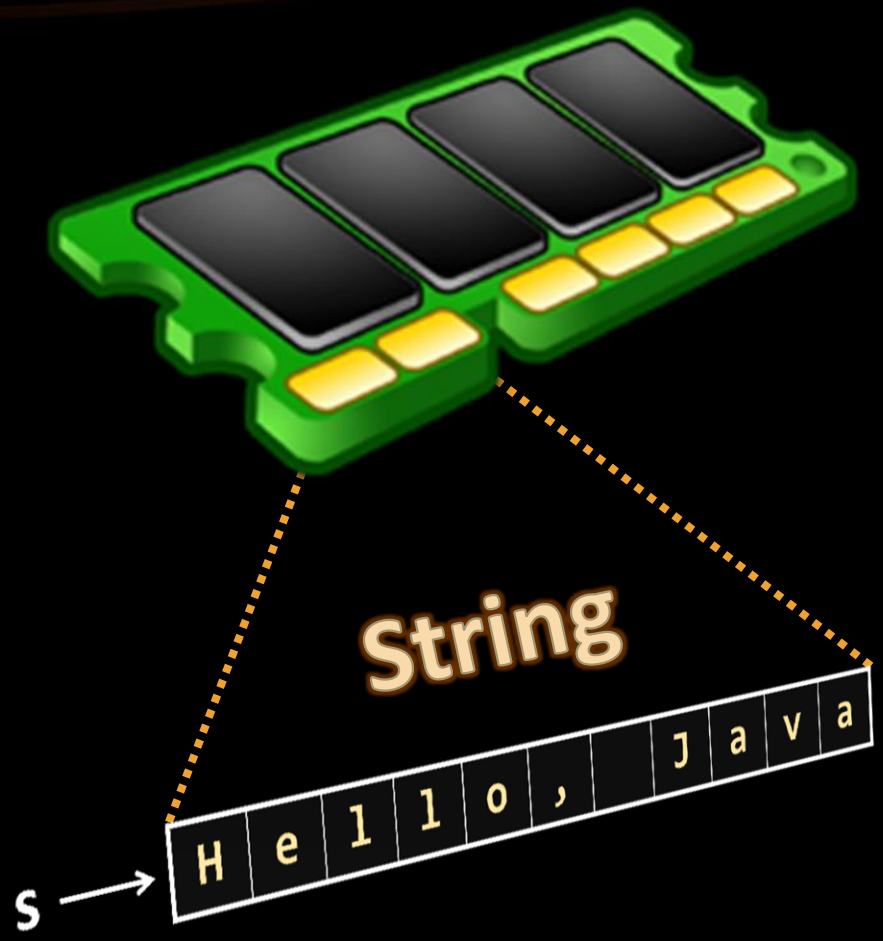
char

The String Data Type

- The String data type:
  - Declared by the **`String`** class
  - Represents a sequence of characters
  - Has a default value **`null`** (no value)
- Strings are enclosed in quotes:

```
String s = "Hello, Java";
```

- Strings can be concatenated
  - Using the **+** operator



Saying Hello – Example

- Concatenating the names of a person to obtain the full name:

```
String firstName = "Ivan";
String lastName = "Ivanov";
System.out.println("Hello, " + firstName);

String fullName = firstName + " " + lastName;
System.out.println("Your full name is: " + fullName);
```

- We can concatenate strings and numbers as well:

```
int age = 21;
System.out.println("Hello, I am " + age + " years old");
```

String Type

Live Demo

string

The Object Type

- The Object type:
  - Is declared by the `java.lang.Object` class
  - Is the base type of all other types
  - Can hold values of any type

Object

```
object dataContainer = 5;
System.out.print("The value of dataContainer is: ");
System.out.println(dataContainer);

dataContainer = "Five";
System.out.print("The value of dataContainer is: ");
System.out.println(dataContainer);
```

SOFTWARE UNIVERSITY FOUNDATION

Objects

Live Demo

Object

true 0xFE "\uE52B"

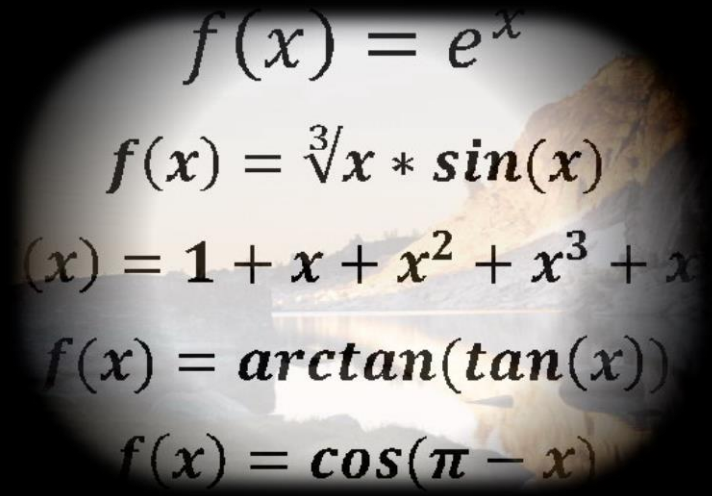
null 6.02e+23

"\t\r\n\r\n\r\n"

SOFTWARE UNIVERSITY FOUNDATION

Variables, Identifiers, Literals

counter



Declaring Variables

- When declaring a variable we:
  - Specify its type
  - Specify its name (called identifier)
  - May give it an initial value

- The syntax is the following:

```
<data_type> <identifier> [= <initialization>];
```

- Example:

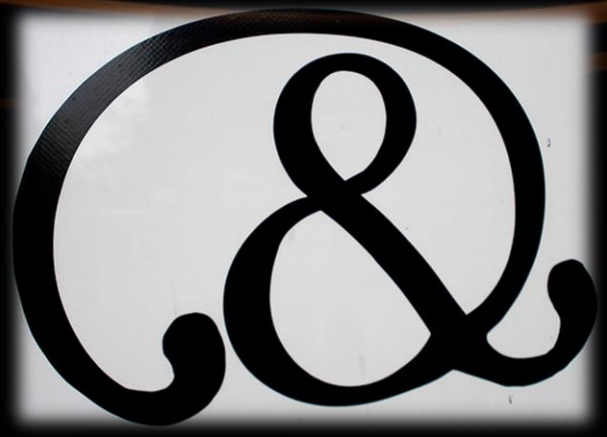
```
int height = 200;
```

Variable Scope

- Variables in Java live inside their { } scope:

```
int sum = 0;
for (int i = 0; i < 10; i++) {
    System.out.println(i);
    sum += i;
    int temp = 2*i;
}

System.out.println(sum); // sum will be printed here
System.out.println(i); // Error: i is out of scope here
System.out.println(temp); // Error: temp is out of scope here
```

Identifiers



- Identifiers may consist of:
  - Letters (Unicode)
  - Digits [`0-9`]
  - Underscore "`_`"
- Examples: `count`, `firstName`, `Page`, `броя4`, `计数器`
- Identifiers
  - Can begin only with a letter or an underscore
  - Cannot be a Java keyword (like `int` or `class`)

Literals in Java

- Literals are the representations of values in the source code

```
int dec = 5; // decimal value 5
int hex = 0xFE; // hexadecimal value FE -> 254
int bin = 0b11001; // binary value 11001 -> 25
int bigNum = 1_250_000; // decimal value 1250000
long num = 123456789012345678 9L;
long hexNum = 0x7FFF_FFFF_FFFF_FFFFL;
boolean bool = true;
float floatNum = 1.25e+7f; // 12500000
double doubleNum = 6.02e+23; // 602000000000000000000000
char newLine = '\n'; // Character <new line>
char unicodeChar = '\u00F1'; // Character: ñ
long fourBytes = 0b11010010_01101001_10010100_10010010; // -764832622
String str = "Hello,\nI\'m Java.";
```

Nullable Types: Integer, Long, Boolean, …

- Each primitive type in Java has a corresponding wrapper:
  - `int` → `java.lang.Integer`
  - `double` → `java.lang.Double`
  - `boolean` → `java.lang.Boolean`
- Primitive wrappers can have a value or be `null` (no value)

```
Integer i = 5; // Integer value: 5
i = i + 1; // Integer value: 6
i = null; // No value (null)
i = i + 1; // NullPointerException
```

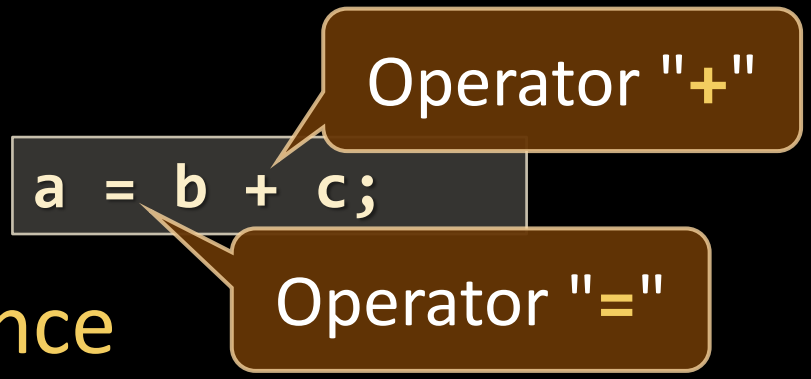
SOFTWARE UNIVERSITY FOUNDATION

```
int bit = (n & (1 << p)) >> p;
n = n & (~(1<<p)) | (bit<<p);
```

Operators and Expressions in Java

What is an Operator?

- Operator is an operation performed over data at runtime
  - Takes one or more arguments (operands)
  - Produces a new value
  - Example of operators:



- Operators have precedence
  - Precedence defines which will be evaluated first
- Expressions are sequences of operators and operands that are evaluated to a single value, e.g. `(a + b) / 2`

Operators in Java

Category	Operators
Arithmetic	`+ - * / % ++ --`
Logical	`&& || ^ !`
Binary	`& | ^ ~ << >> >>>`
Comparison	`== != < > <= >=`
Assignment	`= += -= *= /= %= &= |= ^= <<= >>= >>>=`
String concatenation	`+`
Other	`instanceof . [] () ?: new`

Operators Precedence

Precedence	Operators
Highest	`() [] .`
	`++ --` (postfix) `new typeof`
	`++ --` (prefix) `+ -` (unary) `! ~`
	`* / %`
	`+ -`
	`<< >> >>>`
	`< > <= >=`
	`== !=`
	`&`
Lower	`^`

Operators Precedence (2)

Precedence	Operators
Higher	|
	&&
	||
	?:
	= *= /= %= += -= <<= >>= >>>= &= ^= |=
Lowest	,

- Parenthesis operator always has the highest precedence
- Note: prefer using parentheses to avoid ambiguity

Expressions

- Expressions are sequences of operators, literals and variables that are evaluated to some value (formulas)
- Examples:

```
int r = (150-20) / 2 + 5; // r=70

// Expression for calculating a circle area
double surface = Math.PI * r * r;

// Expression for calculating a circle perimeter
double perimeter = 2 * Math.PI * r;
```

Operators and Expressions – Examples


```
int x = 5, y = 2;
int div = x / y; // 2 (integral division)
float divFloat = (float)x / y; // 2.5 (floating-point division)
long num = 567_972_874; // 567972874
long mid3Digits = (num / 1000) % 1000; // 972
int z = x++; // z = x = 5; x = x + 1 = 6

boolean t = true;
boolean f = false;
boolean or = t || f;
boolean and = t && f;
boolean not = !t;
```

Operators and Expressions – Examples (2)


```
System.out.println(12 / 3); // 4
System.out.println(11 / 3); // 3

System.out.println(11.0 / 3); // 3.6666666666666665
System.out.println(11 / 3.0); // 3.6666666666666665
System.out.println(11 % 3);   // 2
System.out.println(11 % -3);  // 2
System.out.println(-11 % 3);  // -2

System.out.println(1.5 / 0.0);  // Infinity
System.out.println(-1.5 / 0.0); // -Infinity
System.out.println(0.0 / 0.0);  // NaN

int zero = 0;
System.out.println(5 / zero); // ArithmeticException
```

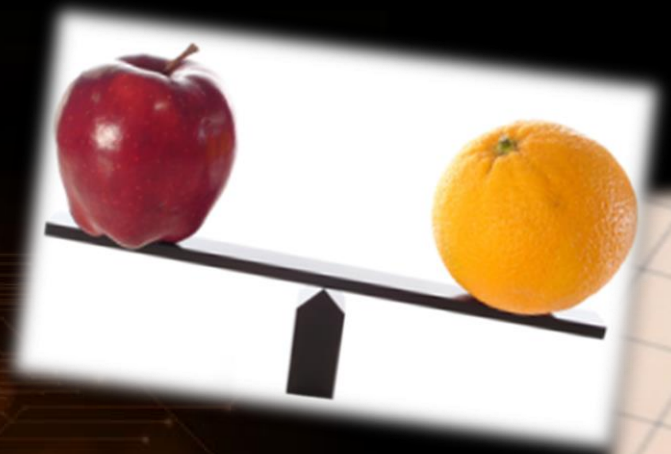
Operators and Expressions – Examples (3)


```
short a = 3;                    // 00000000 00000011
short b = 5;                    // 00000000 00000101
System.out.println( a | b);     // 00000000 00000111 --> 7
System.out.println( a & b);     // 00000000 00000001 --> 1
System.out.println( a ^ b);     // 00000000 00000110 --> 6
System.out.println(~a & b);     // 00000000 00000100 --> 4
System.out.println( a << 1);    // 00000000 00000110 --> 6
System.out.println( a >> 1);    // 00000000 00000001 --> 1

System.out.println(a < b ? "smaller" : "larger");
```

SOFTWARE UNIVERSITY FOUNDATION

Operators

Live Demo



Type Conversion

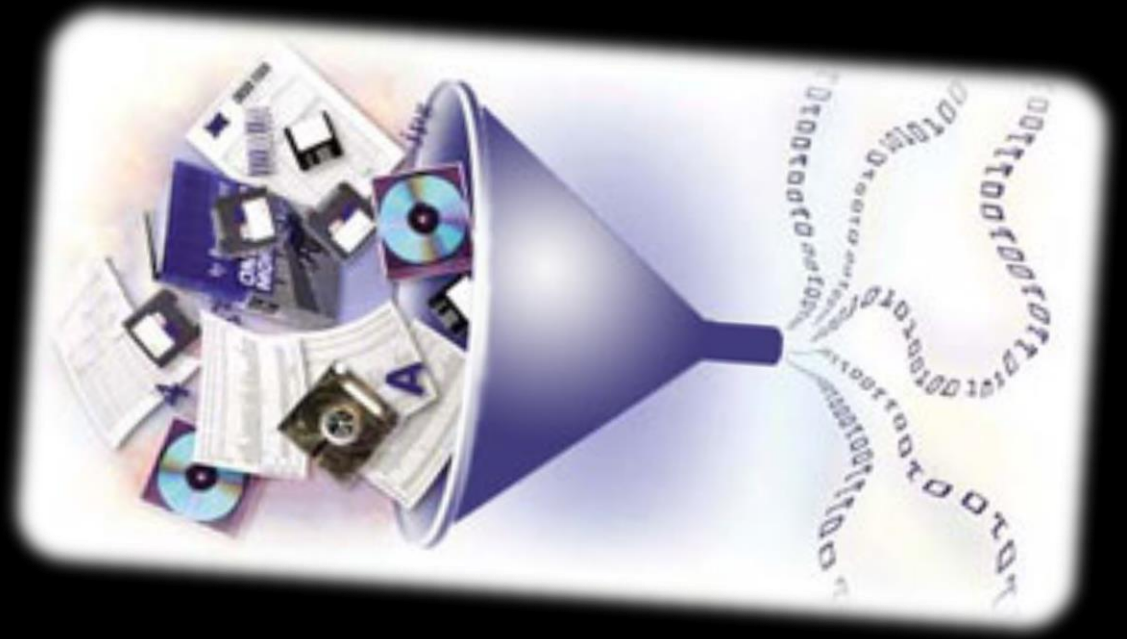
- Type conversion and typecasting change one type to another:

```
long lng = 5;
int intVal = (int) lng; // Explicit type conversion

float heightInMeters = 1.74f; // Explicit conversion
double maxHeight = heightInMeters; // Implicit
double minHeight = (double) heightInMeters; // Explicit
float actualHeight = (float) maxHeight; // Explicit
//float maxHeightFloat = maxHeight; // Compilation error!

// Explicit type conversion with data loss
byte dataLoss = (byte)12345; // 57
```

SOFTWARE UNIVERSITY FOUNDATION

Type Conversions

Live Demo



SOFTWARE UNIVERSITY FOUNDATION

Console Input and Output

Scanner and Formatted Printing

Reading from the Console

- The `java.util.Scanner` class reads strings and numbers

```
import java.util.Scanner;
…

Scanner input = new Scanner(System.in);
int firstNum = input.nextInt();
int secondNum = input.nextInt();
```

Sample Inputs
3 -5
3 -5

- The numbers can be separated by any sequence of whitespace characters (e.g. spaces, tabs, new lines, …)
- Exception is thrown when non-number characters are entered

Reading from the Console (2)

- A more complex example:
  - Read two words from the first line
  - Integer and two doubles from the second line
  - A string from the third line

```
Scanner input = new Scanner(System.in);
String firstWord = input.next("\\w+");
String secondWord = input.next("\\w+");
int numInt = input.nextInt();
double numDouble1 = input.nextDouble();
double numDouble2 = input.nextDouble();
input.nextLine(); // Skip to the line end
String str = input.nextLine();
```

SOFTWARE UNIVERSITY FOUNDATION

Reading from the Console

Live Demo

Printing to the Console

- Using `System.out.print()` and `System.out.println()`:

```
String name = "SoftUni";
String location = "Sofia";
double age = 0.5;
System.out.print(name);
System.out.println(" is " + age +
    " years old organization located in " + location + ".");

// Output:
// SoftUni is 0.5 years old organization located in Sofia.
```

Formatted Printing

- Java supports formatted printing by `System.out.printf()`

```
String name = "SoftUni";
String location = "Sofia";
double age = 0.5;
System.out.printf(
    "%s is %.2f years old organization located in %s.",
    name, age, location);
```

- `%s` – prints a string argument
- `%f` – prints a floating-point argument
- `%.2f` – prints a floating-point argument with 2 digits precision

Formatted Printing (2)

- `%td` / `%tm` / `%tY` – prints day / month / year from a date
- `%1$f` – prints the first argument as floating-point number
- `%2$d` – prints the second argument as integer number

```
System.out.printf(
  "Today is %1$td.%1$tm.%1$tY\n",
  LocalDate.now());
// Today is 10.05.2014

System.out.printf("%1$d + %1$d = %2$d\n", 2, 4);
// 2 + 2 = 4
```

- Learn more at http://docs.oracle.com/javase/8/docs/api/java/util/Formatter.html

Problems | Javadoc | Declaration | Console

<terminated> PrintingToConsole [Java Application] C:\Program Files\Java\jre1.8.0_20\bin\javaw.exe (May 10, 2014, 8:11:45 PM)

```
SoftUni is 0.5 years old organization located in Sofia.
SoftUni is 0.50 years old organization located in Sofia.
Today is 10.05.2014
2 + 2 = 4
```

Printing to the Console

Live Demo

SOFTWARE UNIVERSITY FOUNDATION

Regional Settings

Regional Settings and the Number Formatting

Locale

- Locales in Java define the country and language specific formatting rules
- Reading / printing to the console is locale-dependent
  - The Bulgarian locale uses "`,`" as decimal separator, e.g. `1,25`
  - The United States locale uses "`.`" as decimal separator, e.g. `1.25`
- Changing the System locale (for the entire VM)

```
Locale.setDefault(Locale.ROOT); // Language-neutral locale
Locale.setDefault(new Locale("BG", "BG")); // Bulgarian
Locale.setDefault(Locale.US); // United States
```

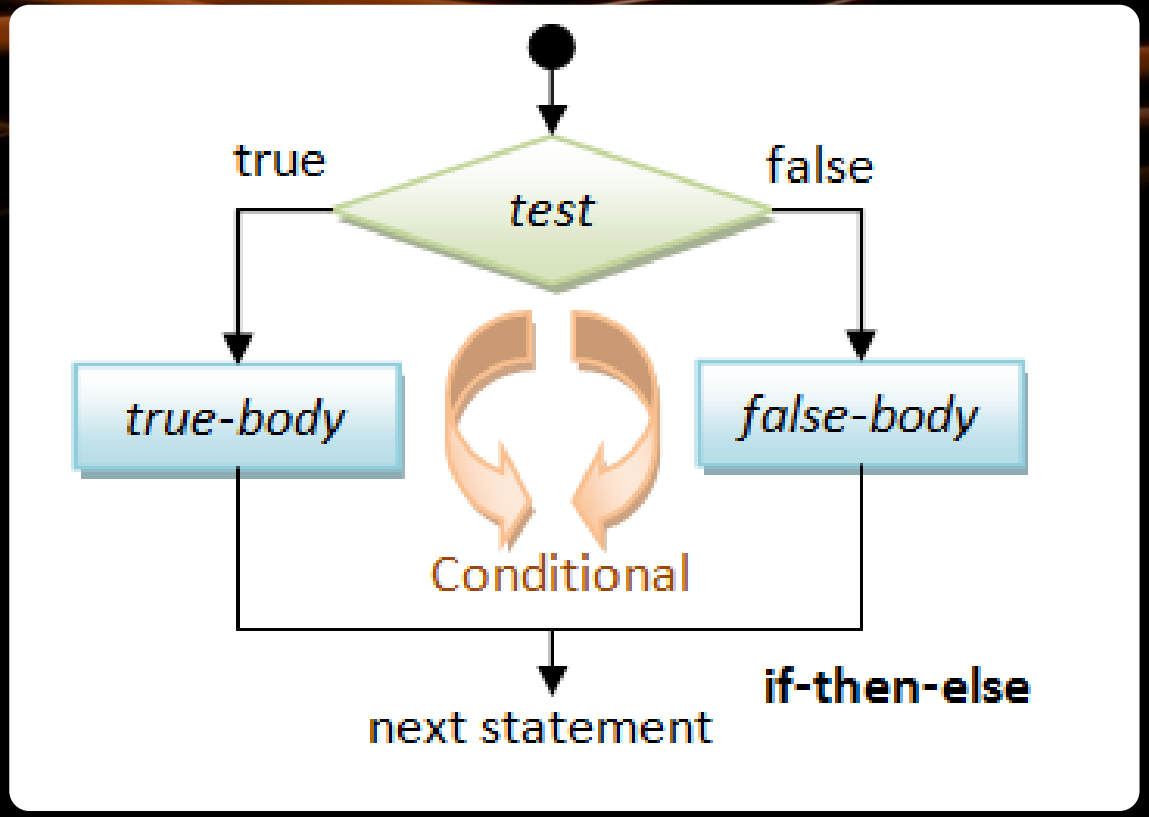
Locale-Specific Input and Output

SOFTWARE UNIVERSITY FOUNDATION

```
int age = 0.5;

Locale.setDefault(Locale.ROOT);
System.out.printf("%f\n", age); // 0.500000
System.out.println(age); // 0.5
Scanner inputRoot = new Scanner(System.in);
double d = inputRoot.nextDouble(); // Expects 1.25

Locale.setDefault(new Locale("BG", "BG"));
System.out.printf("%f\n", age); // 0,500000
System.out.println(age); // 0.5
Scanner inputBG = new Scanner(System.in);
d = inputBG.nextDouble(); // Expects 1,25
```

Regional Settings

Live Demo



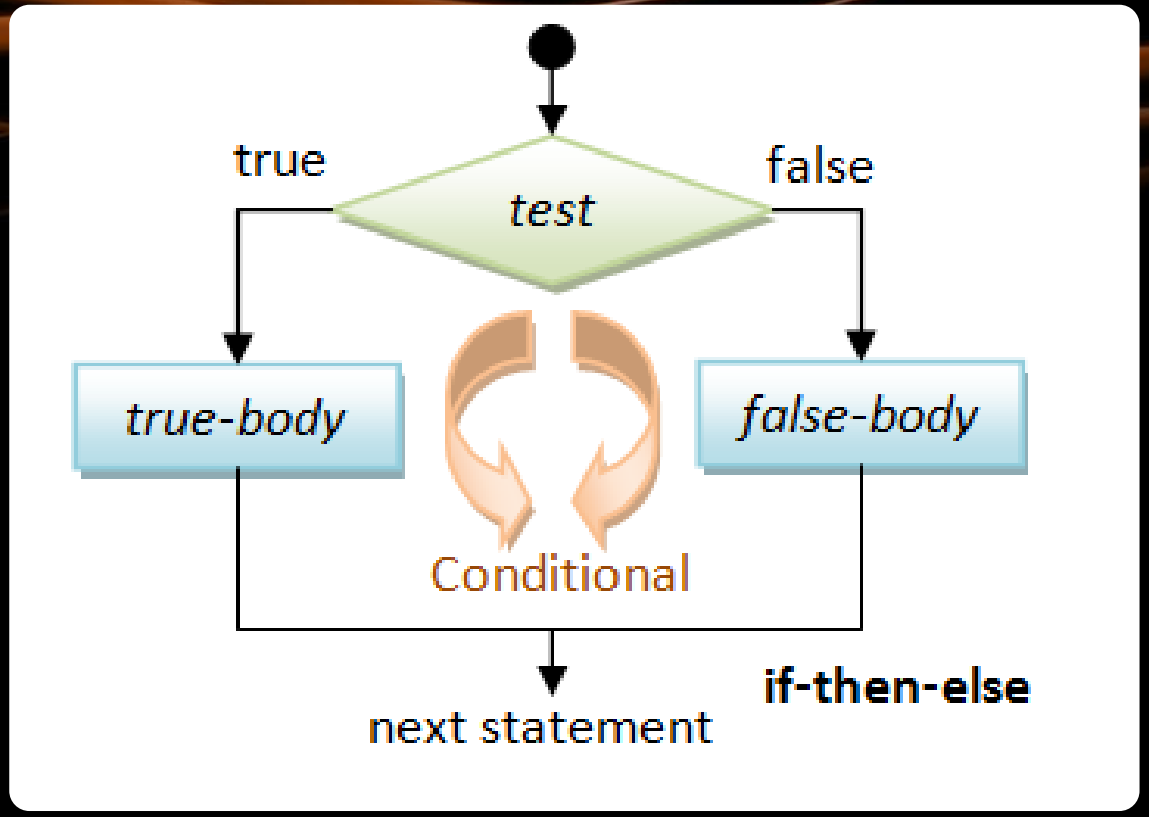
if and if-else

Implementing Conditional Logic

Conditional Statements: if-else

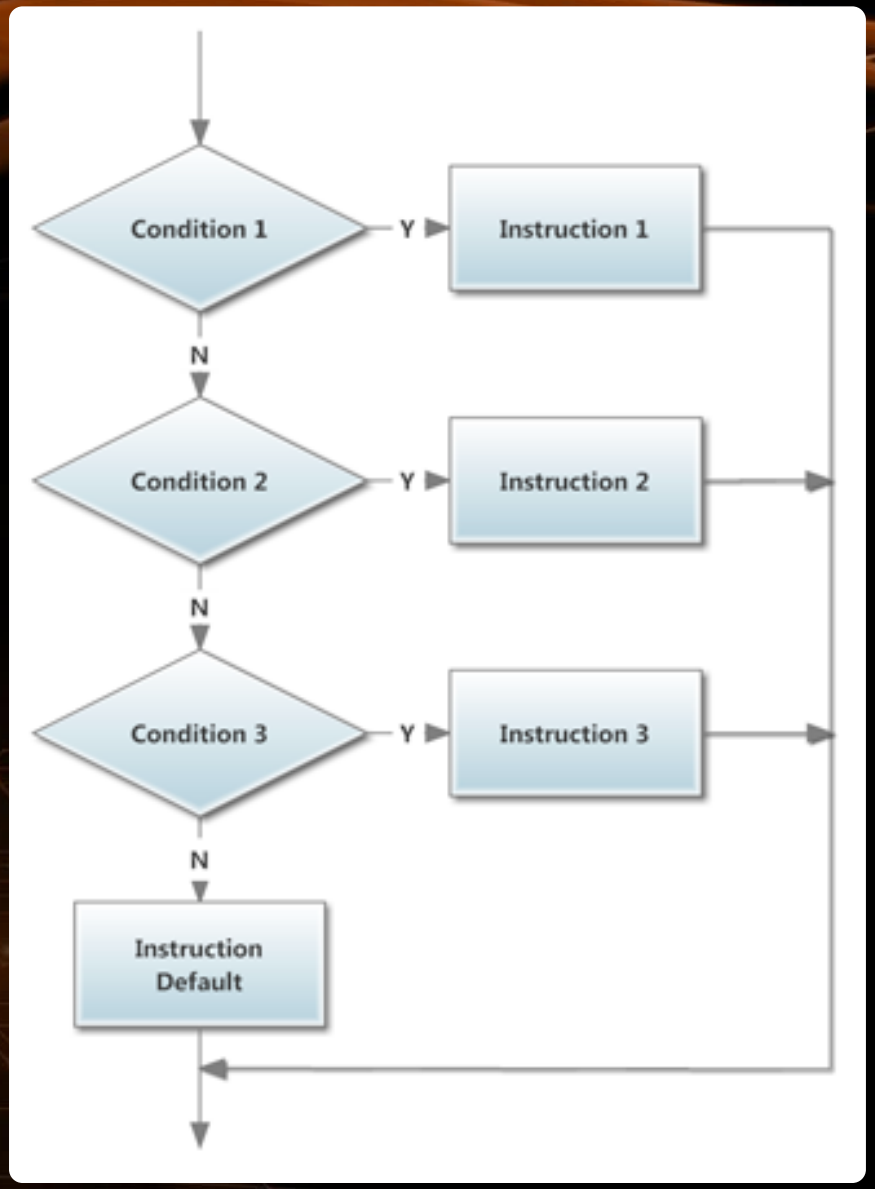
- Java implements the classical **if** / **if-else** statements:

```
Scanner scanner = new Scanner(System.in);
int number = Integer.parseInt(scanner.nextLine());

if (number % 2 == 0)
{
    System.out.println("This number is even.");
}
else
{
    System.out.println("This number is odd.");
}
```



if and if-else

Live Demo



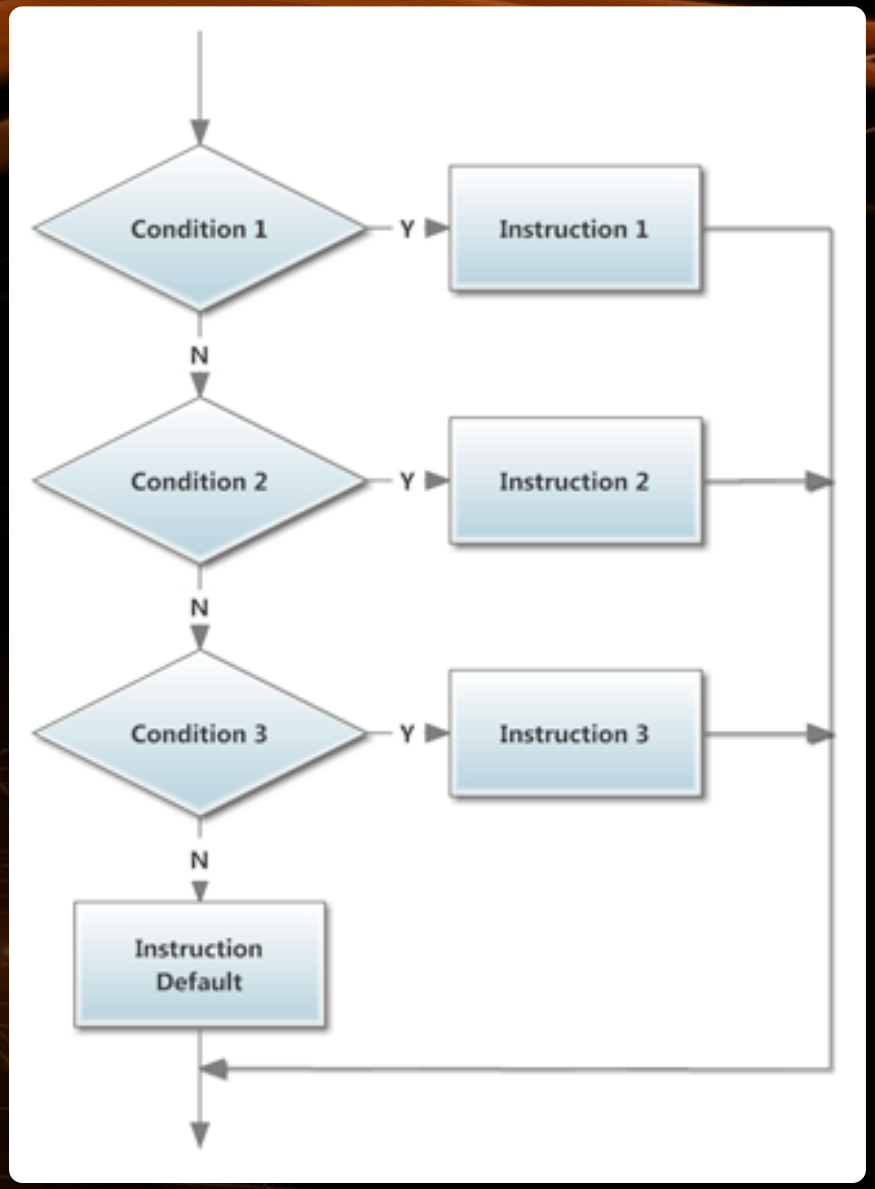
switch-case

Checking Multiple Conditions

Conditional Statements: switch-case

- Java implements the classical **switch-case** statements:

```
switch (day) {
   case 1: System.out.println("Monday"); break;
   case 2: System.out.println("Tuesday"); break;
   case 3: System.out.println("Wednesday"); break;
   case 4: System.out.println("Thursday"); break;
   case 5: System.out.println("Friday"); break;
   case 6: System.out.println("Saturday"); break;
   case 7: System.out.println("Sunday"); break;
   default: System.out.println("Invalid day!"); break;
}
```



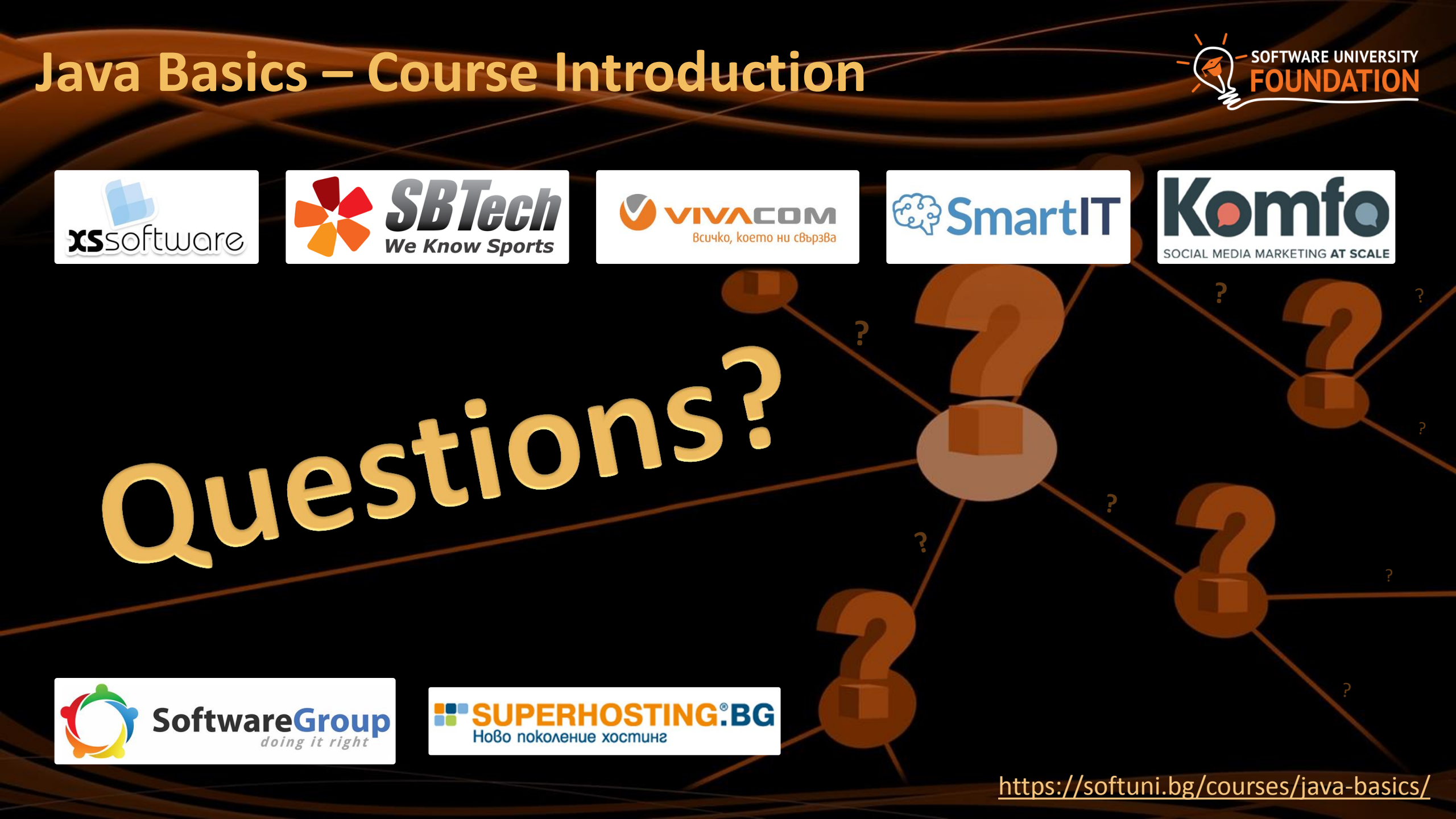
switch-case

Live Demo

Summary

- Java supports limited set of primitive data types: `byte`, `short`, `int`, `long`, `float`, `double`, `boolean`, `object`
- Java supports the classical operators, expressions and statements
  - Like in C#, C++, JavaScript, PHP, …
- `Scanner` and `System.out.printf()` provide formatted input / output
- Java supports the classical `if-else` and `switch-case`

Java Basics – Course Introduction



https://softuni.bg/courses/java-basics/

License

- This course (slides, examples, demos, videos, homework, etc.) is licensed under the "Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International" license

- Attribution: this work may contain portions from
  - "Fundamentals of Computer Programming with Java" book by Svetlin Nakov & Co. under CC-BY-SA license
  - "C# Basics" course by Software University under CC-BY-NC-SA license

Free Trainings @ Software University

- Software University Foundation – softuni.org
- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg
- Software University @ Facebook
  - facebook.com/SoftwareUniversity
- Software University @ YouTube
  - youtube.com/SoftwareUniversity
- Software University Forums – forum.softuni.bg