Java
Lecture 1 - Language Basics

PRAGMATIC

IT Learning & Outsourcing Center

Who is that guy ?

- 4 години опит в разработката на различни Java и Android приложения
- Опит в разработката и поддръжката на интегрирана среда за разработка на софтуер и системи за Forex търговия
- Опит с Continuous Integration, Unit и Automation testing, Agile методологии и най-добрите практики за разработка на софтуер
- Стартирал стартъпа FutureSoft Ltd.
- Водил множество семинари и обучения по програмиране в различни училища и на различни конференции
- Бивш възпитаник на Pragmatic
- Бакалавърска степен – Специалност “Международни икономически отношения” в УНСС
- Магистърска степен – Специалност “Управление на ИТ проекти” в НБУ
- Текуща позиция – Java Developer в Avus Capital Ltd.

The course

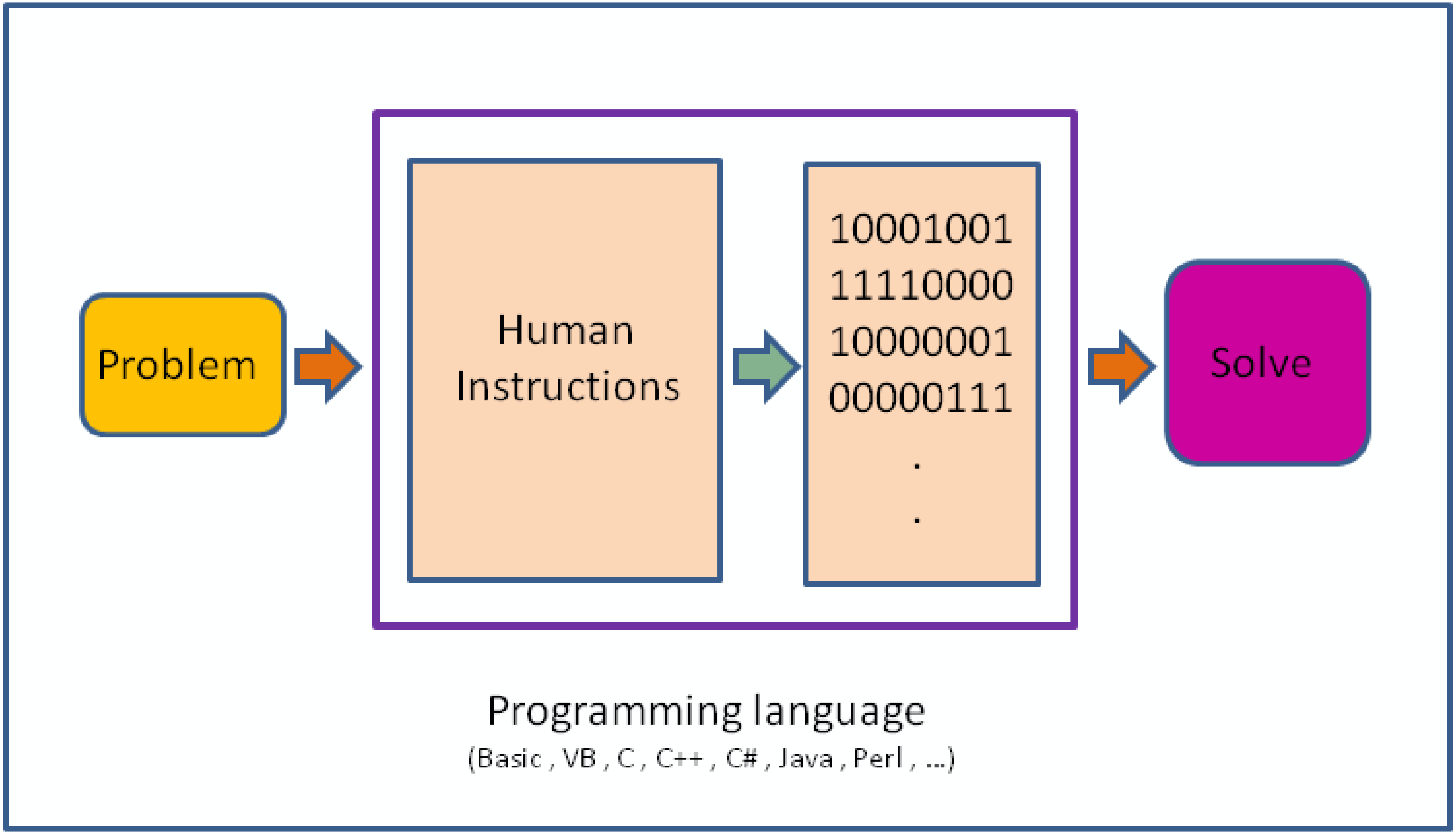
- Introduction, Primitives, if-else
- Loops
- Strings, Classes, Fields, Methods
- Methods and Constructors
- Inheritance
- Interfaces, Abstract classes
- Arrays and Collections
- Generics and Exceptions
- Practice

Lecture 1 - Agenda

- The Java language
- Setting up working environment
- Introduction to Eclipse
- First java program
- Primitives and variables
- Basic operations
- Statements
- Working with the console
- If-else statement and blocks

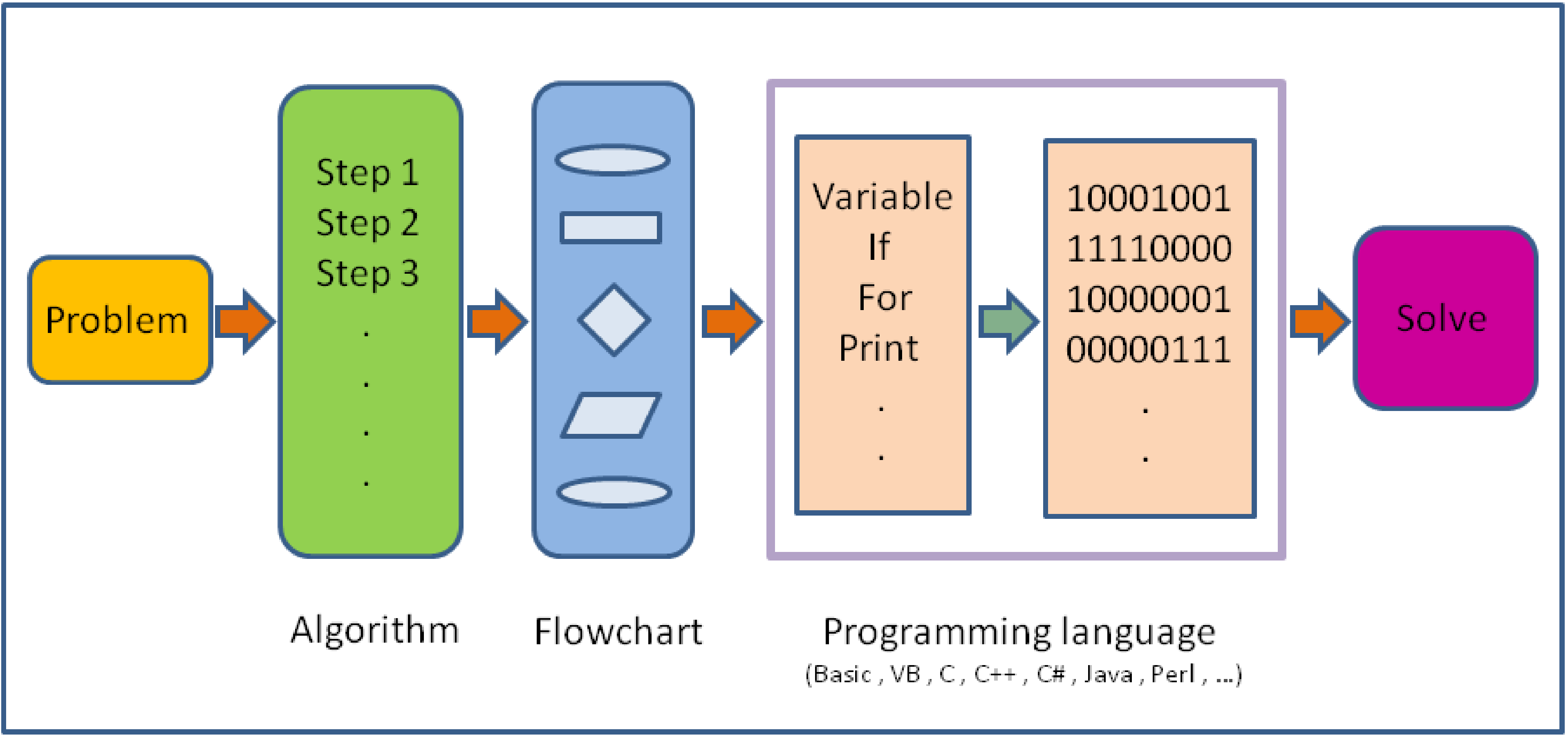
Before we begin

Programing Fundamentals



Programming language
(Basic , VB , C , C++ , C# , Java , Perl , ...)

Programing Fundamentals



Programing Fundamentals

- Why is it called a programing LANGUAGE ?

Programing Fundamentals

Понякога ще идвам във съня ти
като нечакан и неискан гостенин.
Не ме оставяй ти отвън на пътя –
вратите не залоствай.

Ще влезна тихо. Кротко ще приседна,
ще вперя поглед в мрака да те видя.
Когато се наситя да те гледам –
ще те целуна и ще си отида.

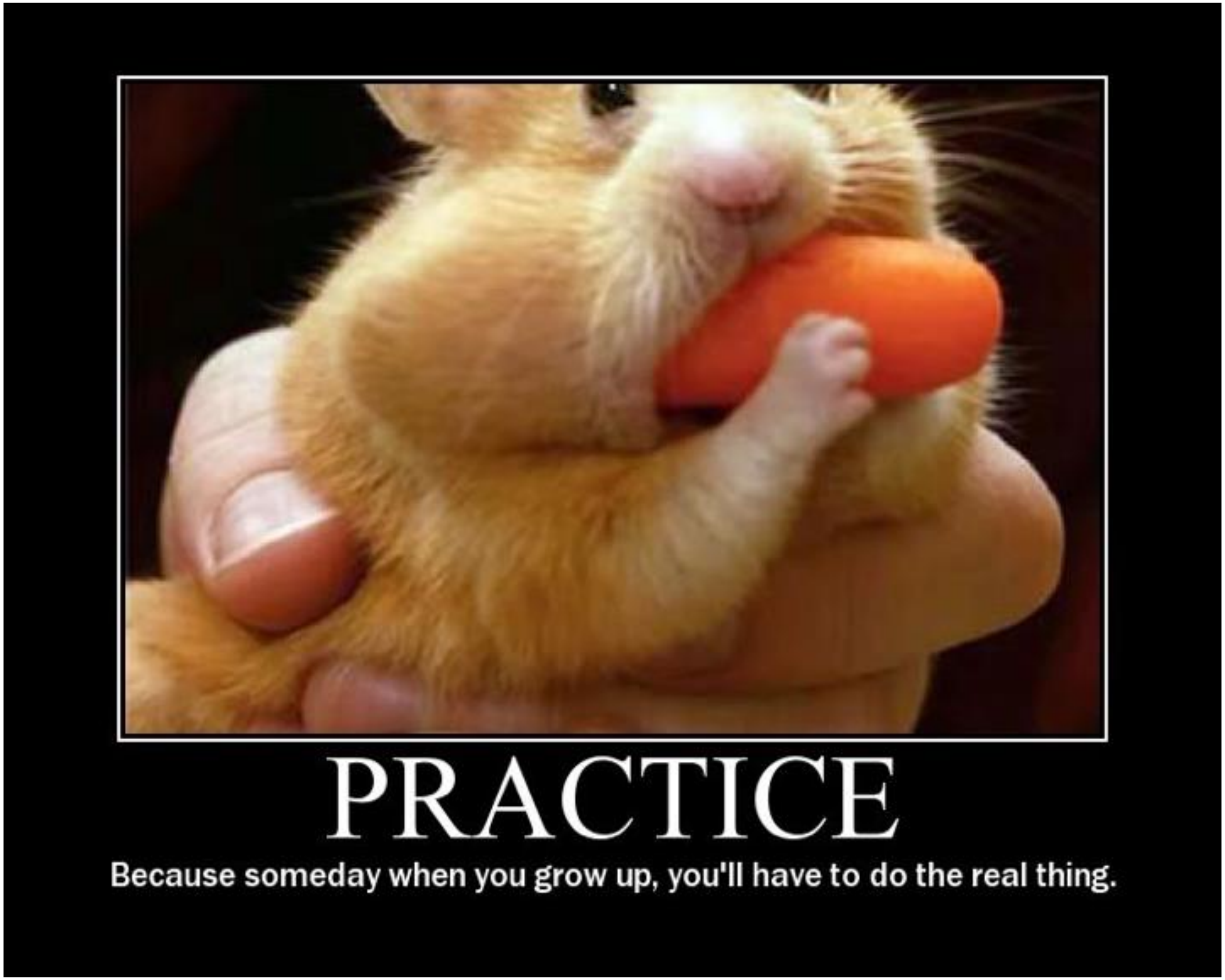
- - Кой е автора ?

Programing Fundamentals

- What is programming
  - Problem-solving
  - Creative process

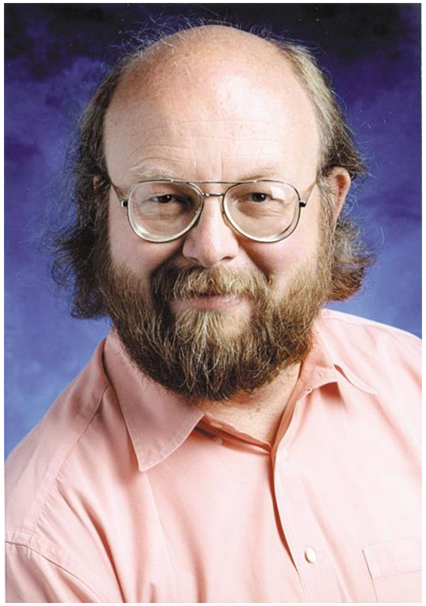
- How to become a programmer
  - Read
  - Practice
  - Practice
  - Practice
  - And practice some more
  - Repeat

Programing Fundamentals


PRACTICE
Because someday when you grow up, you'll have to do the real thing.

The Java Language

- What is java as language
  - Developed in 1995 by James Gosling
  - Very widely used programming language
  - Suitable for desktop, web, embedded applications
  - Object-Oriented language
  - Uses C-like syntax
  - Java is platform independent - programs run on Java Virtual Machine (JVM)

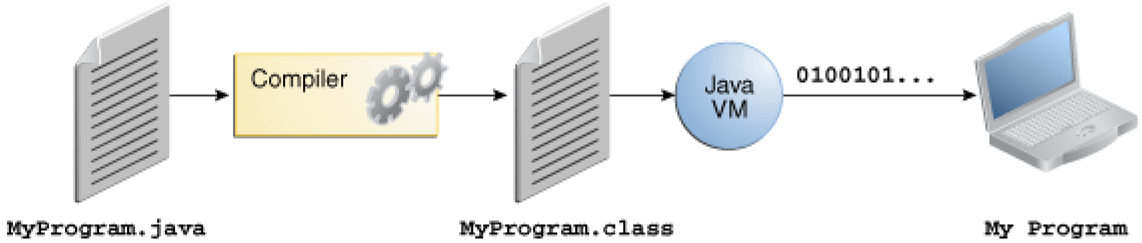


The Java Language

- Java runtime environment (JRE) – consists of
  - JVM
  - core classes
  - supporting files.
- Programmers use JDK (Java Development Kit)

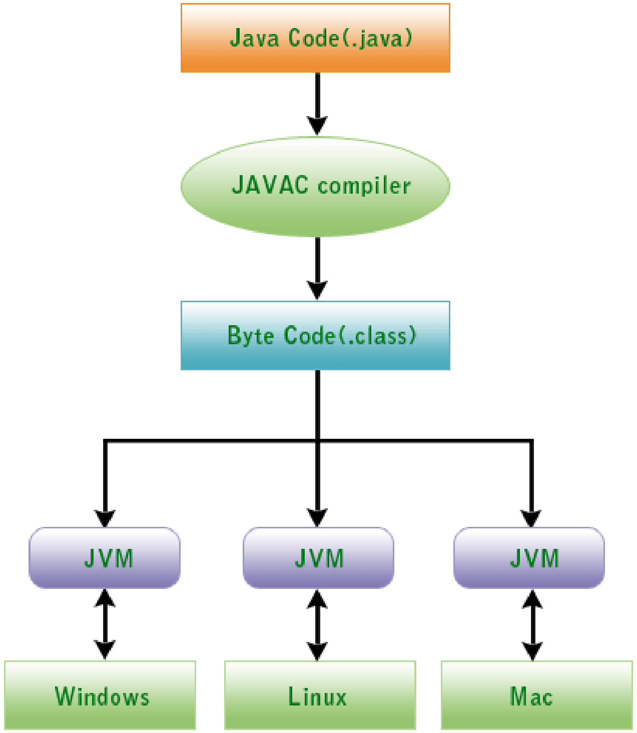
Java Compiler

- Java source code is human readable code in .java files
- Compilation
- .class file does not contain code that is native to your processor. It instead contains bytecodes
- Java virtual machine



Platform-independent

- Because the Java VM is available on many different operating systems, the same .class files are capable of running on Windows, Linux, Mac OS ...



Setting Up Working Environment
JDK + Eclipse IDE

- Java Development Kit (JDK) - http://www.oracle.com/technetwork/java/javase/downloads/index.html

- Configure JAVA_HOME and Path system vars
- Windows Button + R, type ‘cmd’
- Type: java -version

- Eclipse IDE - http://www.eclipse.org/downloads/

Java – first steps

- My first class
  - All java classes start with capital letter
  - Class names do not include spaces
  - Each class is a file. File and class name are the same
  - .class and .java
  - Java is case sensitive

My First Program

- *main method – entry point for each java program*
- System.out.println();
- HelloWorld program
- What is console?

```
public class  HelloWorld {
        public static void main(String[] args) {
                System.out.println("Hello from the other side");
        }
}
```

- HelloWorld.java in the code examples

Variables (part 1)

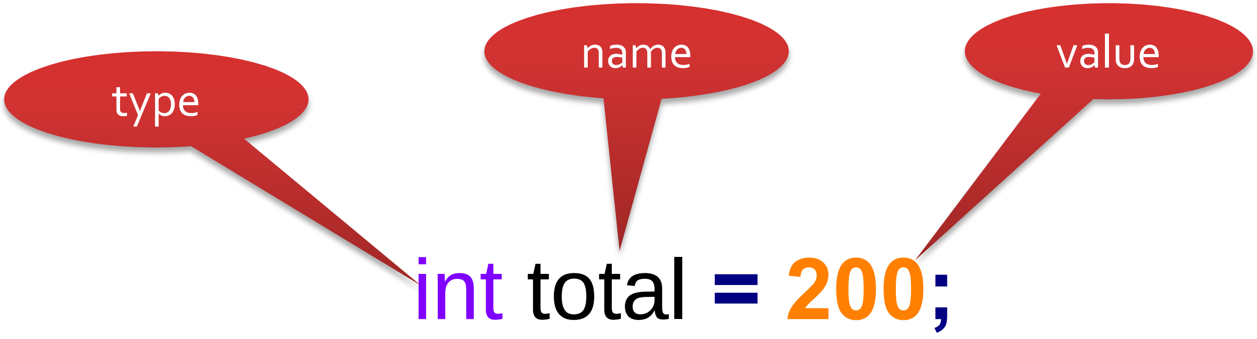
- The Java programming language uses both "fields" and "variables" as part of its terminology.
  - Instance variables (non-static fields) are unique to each instance of a class.
  - Class variables (static fields) are fields declared with the *static* modifier - there is exactly one copy of a class variable, regardless of how many times the class has been instantiated.
  - Local variables store temporary state inside a method.
  - Parameters are variables that provide extra information to a method.

Variables (part 2)

- Both local variables and parameters are always classified as "variables" (not "fields").
- When naming your fields or variables, there are rules and conventions that you should (or must) follow.

Variables (part 3)

- Variables in java
  - It's purpose is to hold information
  - Have an unique name
  - Have a type
  - Have a value (can be changed)
- Declaring variable



type

name

value

```
int total = 200;
```

Има ли будни?

Primitive Types in Java

- Primitives are basic java type
- Primitives can be used with basic operations
- Primitives' values can be assigned to variables

- Primitive types in java
  - byte, short, int, long
  - float, double
  - boolean
  - char

Numeric types

- Numeric types are **byte, short, long, int, double, float**
- **byte – 8b (-128 : 127)**
- *byte b = 100;*
- **short – 16b (-32768 : 32767)**
- *short s = 10000;*
- **int – from integer, 32b**
- *int i = 10000;*
- (-2,147,483,648 : 2,147,483,647)

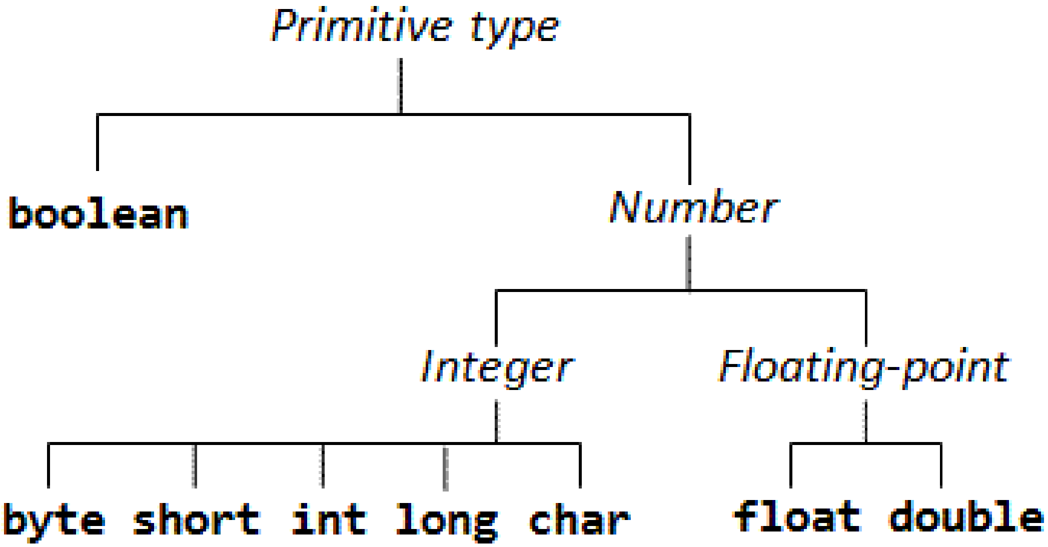
Numeric Types

- **long – 64b**
- `long l = 100l;`
- ***L** suffix is added automatically to indicate long type*
- (-9,223,372,036,854,775,808 : 9,223,372,036,854,775,807)
- **float - precision to 32b**
- `float f = 3.14f;`
- ***f** suffix is added automatically to indicate long type*
- **double – precision to 64b**
- `double d= 3.14d;`
- ***d** suffix is added automatically to indicate long type*

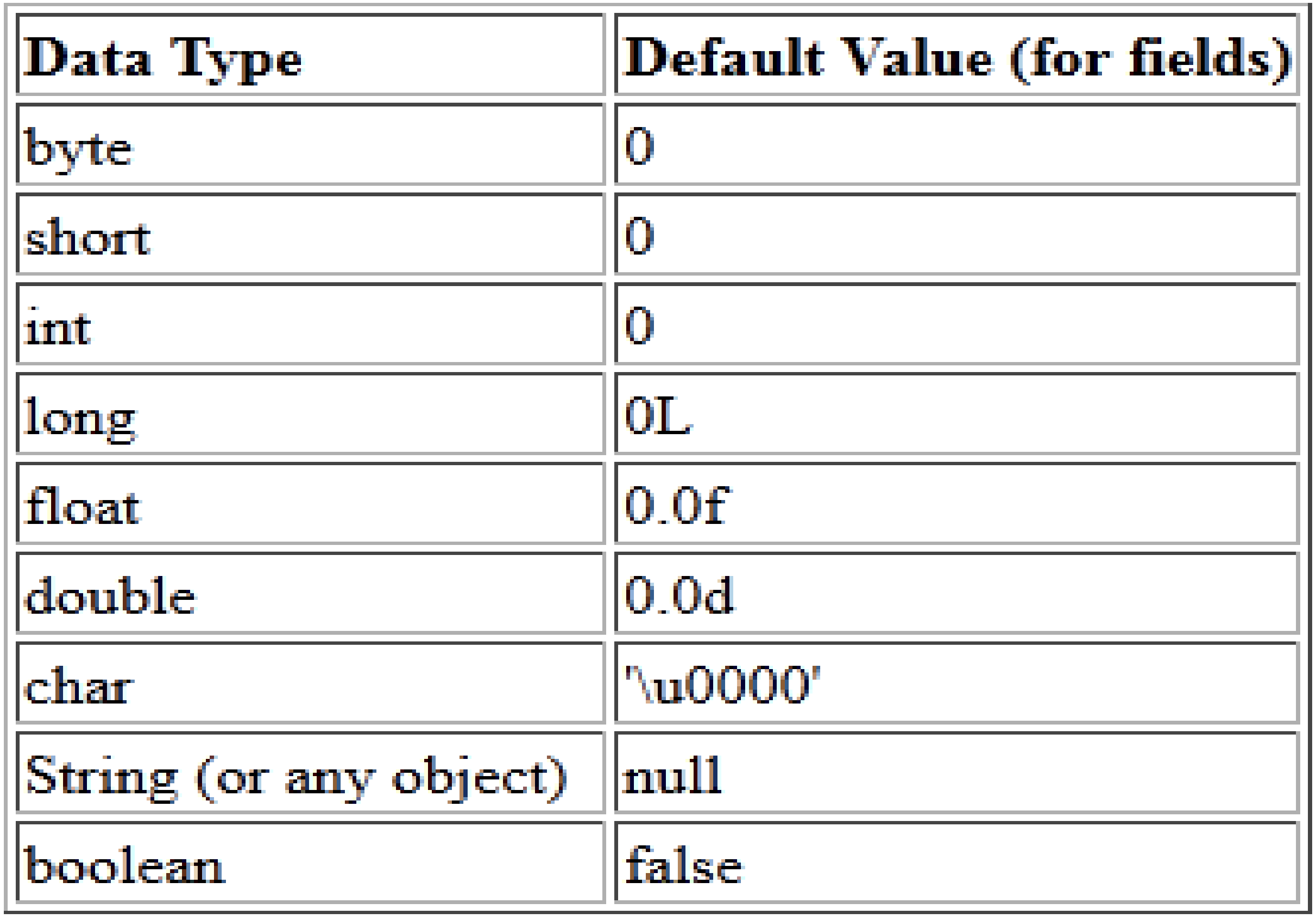
- ***Java primitive data types:***
- http://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html

char and boolean

- **char is used for 16b unicode character**
- Char values are embedded in ' `
- `char ch = 'c';`
- **boolean has two values - true or false**
- `boolean bool = false;`



Primitive types - default values



Data Type	Default Value (for fields)
byte	0
short	0
int	0
long	0L
float	0.0f
double	0.0d
char	'\u0000'
String (or any object)	null
boolean	false

- VariableExample.java in the code examples

Simple Assignment Operator

- = Simple assignment operator
- Examples:
- int cadence = 0;
- int speed = 0;
- int gear = 1;

Arithmetic Operators

- +   Additive operator (also used for String concatenation)
- -   Subtraction operator
- *   Multiplication operator
- /   Division operator
- %   Remainder operator

- ArithmeticDemo.java in code examples

Unary Operators

- + Unary plus operator; indicates positive value (numbers are positive without this, however)
- - Unary minus operator; negates an expression
- ++ Increment operator; increments a value by 1
- -- Decrement operator; decrements a value by 1
- ! Logical complement operator; inverts the value of a boolean
- UnaryDemo.java and PrePostDemo.java in the code examples

Equality and Relational Operators

==	Equal to
!=	Not equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to

- ComparisonDemo.java in the code examples

Conditional Operators

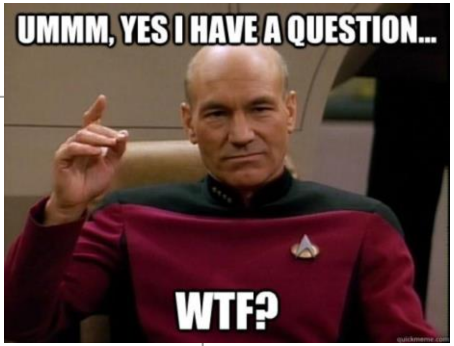
- && Conditional-AND
- || Conditional-OR
- ?: Ternary (shorthand for if-then-else statement)
- ConditionalDemo01.java and ConditionalDemo02.java in the code examples

Reading from console

- Using Scanner

```
Scanner sc = new Scanner(System.in);
```

Read user input with sc.nextXXX();



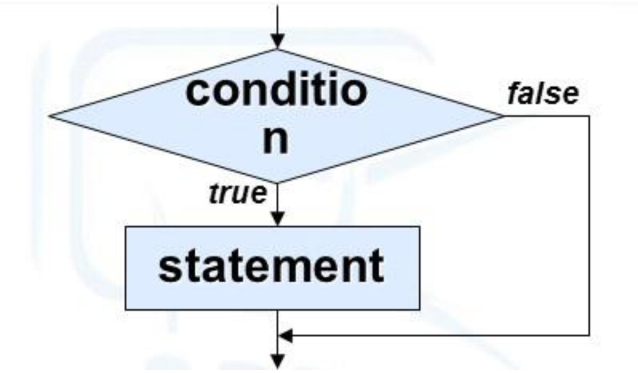
```
sc.nextInt();
sc.nextDouble();
sc.nextLong();



ScannerDemo.java in the code examples
```

Control flow

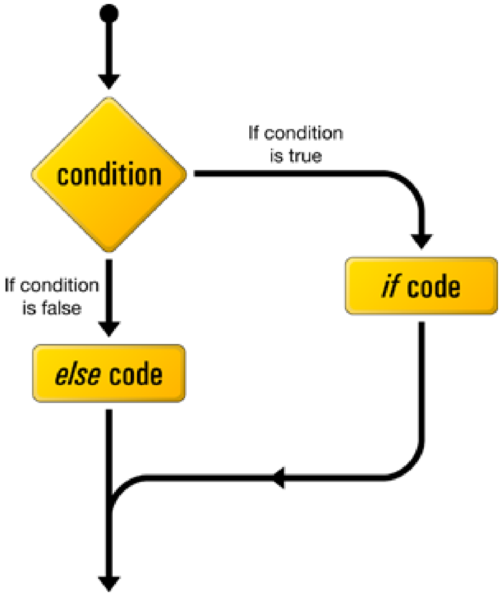
- Control flow is the way a program goes – execution of predefined statements
- Control flow may differ each time depending on conditions – either input data, or predefined conditions by the programmer(i.e – time and so on)
- During the program execution decisions are being met – the program flow branches

if-else statement

- if (condition) {
- *statement*
- }

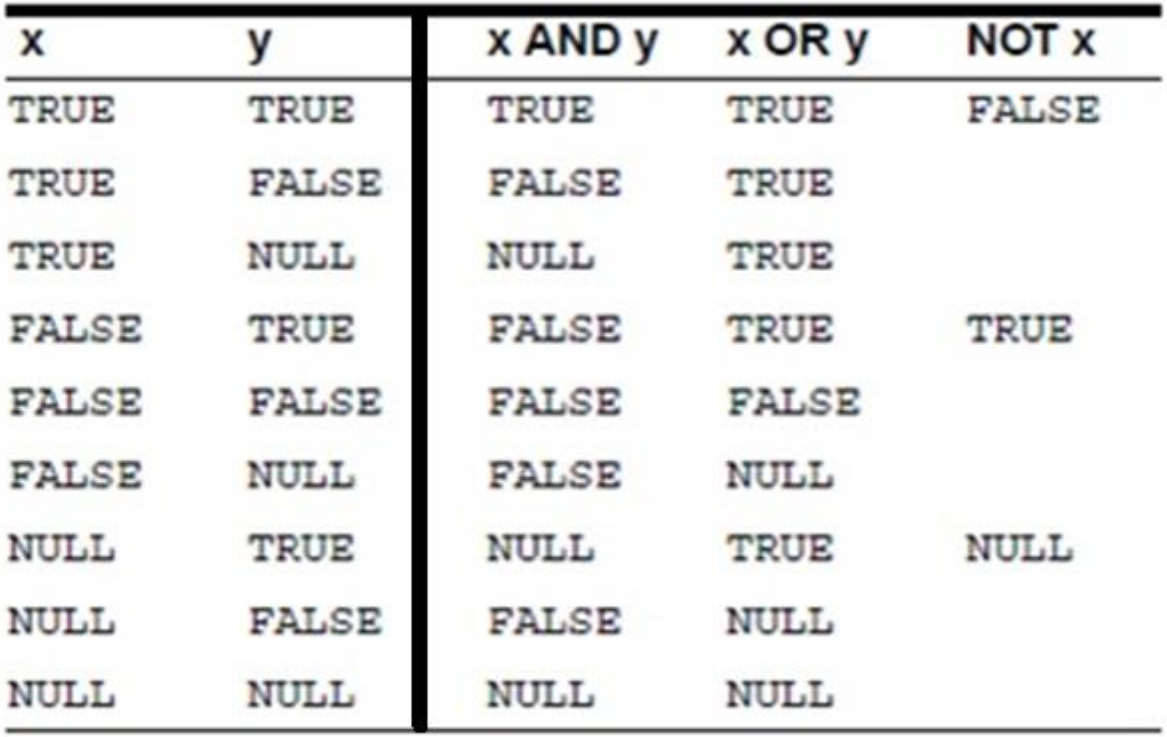


- if (condition) {
- *executionA*
- } else {
- *executionB*
- }



Conditional statement

- Logical NOT !
- Logical AND &&
- Logical OR ||



x	y	x AND y	x OR y	NOT x
TRUE	TRUE	TRUE	TRUE	FALSE
TRUE	FALSE	FALSE	TRUE	
TRUE	NULL	NULL	TRUE	
FALSE	TRUE	FALSE	TRUE	TRUE
FALSE	FALSE	FALSE	FALSE	
FALSE	NULL	FALSE	NULL	
NULL	TRUE	NULL	TRUE	NULL
NULL	FALSE	FALSE	NULL	
NULL	NULL	NULL	NULL	

if-else statement

- If can exist without else
- But
- Else can't exist without if
- Nested if-else statement

```java
double a = 7.5;
if (a < 0) {
      System.out.println("a is smaller than 0");
} else {
      if (a == 0) {
      System.out.println("a is 0");
      } else {
      System.out.println("a is bigger than 0");
      }
}
```

- IfElseExample.java in the code examples

Blocks

- A block is a group of zero or more statements between balanced braces and can be used anywhere a single statement is allowed

```
if (a > 10) {
      System.out.println("a is " + a);
      System.out.println("a is bigger than 10");
} else {
      System.out.println("a is not bigger than 10");
}
```

- Always format your code! Do NOT write code like this:

```
if (a > 10) {
System.out.println("a is " + a);
System.out.println("a is bigger than 10");}
else {System.out.println("a is not bigger than 10");
}
```

Mistake

```java
int a = 7;
if (a > 10); {
   System.out.println("a is " + a);
   System.out.println("a is bigger than 10");
}
```

- In these cases println statements will be executed no matter the condition!

```java
int a = 7;
if (a > 10);
{
   System.out.println("a is " + a);
   System.out.println("a is bigger than 10");
}
```

Summary

- Startup
- Variables
- Primitive types
- Operators

- Working with the console
- If-else statement and blocks