

# Введение в программирование на языке Java

Занятие 1. Знакомство с языком

# О себе

- Закончил МГТУ им. Баумана
- Проходил стажировку в 1С ЦМС
- Работаю в 1С в отделе разработки инструментов разработки

# План занятия

- История появления Java
- Для чего нужна и где сейчас используется
- Базовый синтаксис
- Примитивные типы
- Операторы
- Методы

# История появления

- Sun Microsystems (1991-1996)
  - Write Once, Run Anywhere
  - C/C++ - style
- FOSS (2006)
- Oracle (2009-2010)











## **3 Billion Devices Run Java**

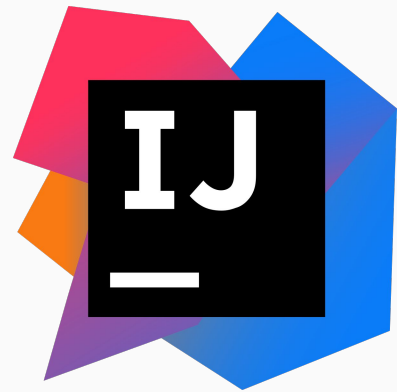
Computers, Printers, Routers, Cell Phones, BlackBerry, Kindle, Parking Meters, Public Transportation Passes, ATMs, Credit Cards, Home Security Systems, Cable Boxes, TVs...

**ORACLE**

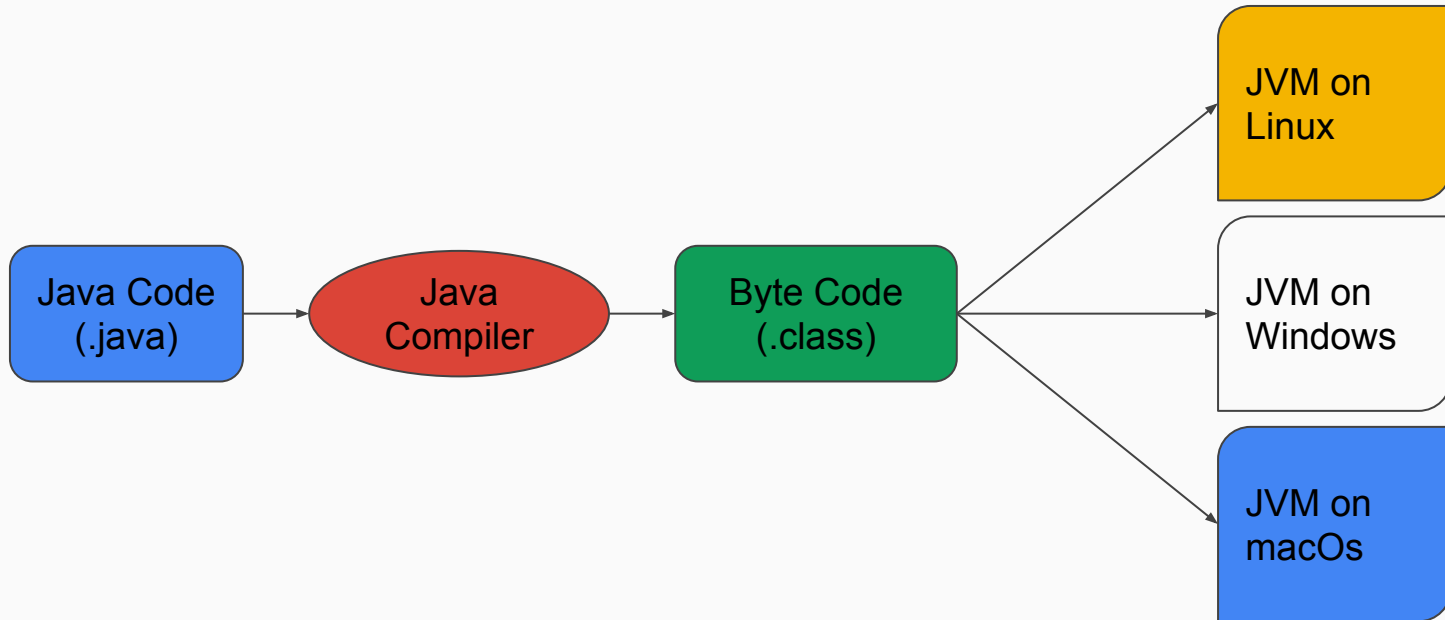
# Tiobe

Aug 2021	Aug 2020	Change	Programming Language		Ratings	Change
1	1			C	12.57%	-4.41%
2	3	▲		Python	11.86%	+2.17%
3	2	▼		Java	10.43%	-4.00%
4	4			C++	7.36%	+0.52%
5	5			C#	5.14%	+0.46%
6	6			Visual Basic	4.67%	+0.01%
7	7			JavaScript	2.95%	+0.07%
8	9	▲		PHP	2.19%	-0.05%

Огромное количество библиотек,  
фреймворков и инструментов



# Write Once, Run Everywhere



# Где используется Java





# Инструменты разработки

- JDK (Java Development Kit) - Набор инструментов для разработки и запуска Java приложений
  - OpenJDK, BellSoft
- IntelliJ IDEA / Eclipse Java Development Tools
- Maven - система сборки
- Git - система контроля версий

# Java Development Kit (JDK)



The diagram illustrates the components of the Java Development Kit (JDK) using three nested rectangles. The outermost rectangle is red and represents the JDK. Inside it is a blue rectangle representing the JRE (Java Runtime Environment). Inside the blue rectangle is a yellow rectangle representing the JVM (Java Virtual Machine). The text labels are placed within their respective rectangles: 'JDK: javac, jheap, jmap...' in the red rectangle, 'JRE: Java Class Library' in the blue rectangle, and 'JVM' in the yellow rectangle.

**JDK: javac, jheap, jmap...**

**JRE: Java Class Library**

**JVM**

# Hello World!

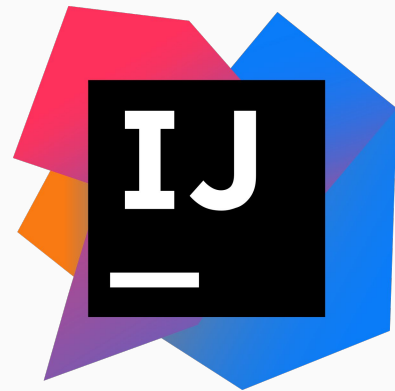


Main.java

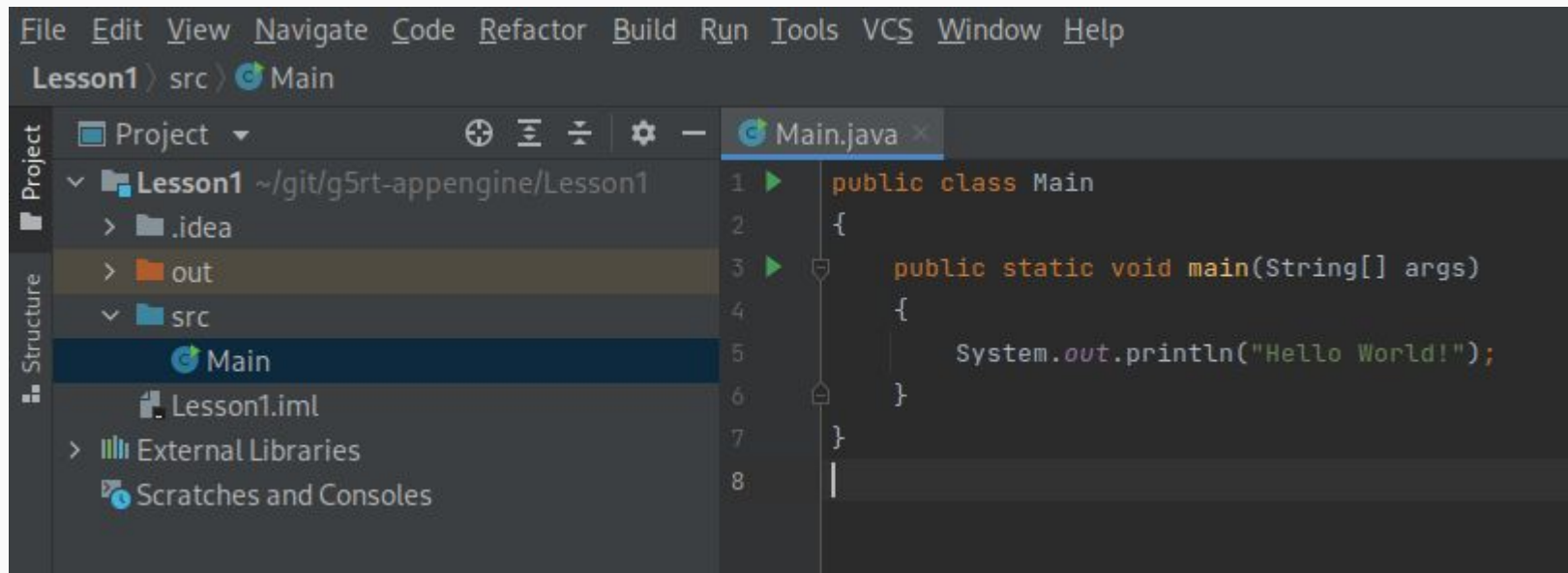
```
1  public class Main {  
2      public static void main(String[] args) {  
3          System.out.println("Hello, World!");  
4      }  
5  }  
6
```

# IntelliJ IDEA

- Ускоряет процесс разработки
  - Компилятор и валидатор
  - Работа с текстом
  - Автоматическая сборка
  - Отладка



# Создание проекта в IDEA



# JAR

- JAR - Java ARchive
  - ZIP
  - Байткод
  - Ресурсы
  - Манифест



# Библиотеки

- Большинство приложений используют библиотеки
- Формат: JAR
- Формируют Classpath
- Используются как во время компиляции, так и во время исполнения

# Maven



- Разработан Apache
- Описывается pom.xml
  - Project Object Model



# Maven Lifecycle

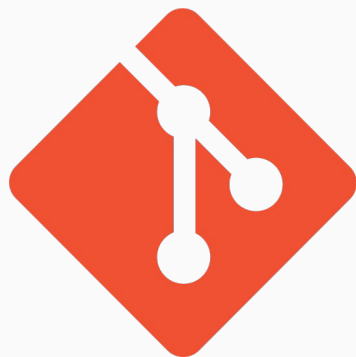
- validate
- compile
- test
- package
- integration-test
- verify
- install
- deploy

# POM

```
pom.xml > project
1  <?xml version="1.0" encoding="UTF-8"?>
2  <project xmlns="http://maven.apache.org/POM/4.0.0"
3      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4      xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
5      <modelVersion>4.0.0</modelVersion>
6
7      <groupId>org.lessons</groupId>
8      <artifactId>first</artifactId>
9      <version>1.0-SNAPSHOT</version>
10
11     <properties>
12         <maven.compiler.source>11</maven.compiler.source>
13         <maven.compiler.target>11</maven.compiler.target>
14     </properties>
15
16 </project>
```

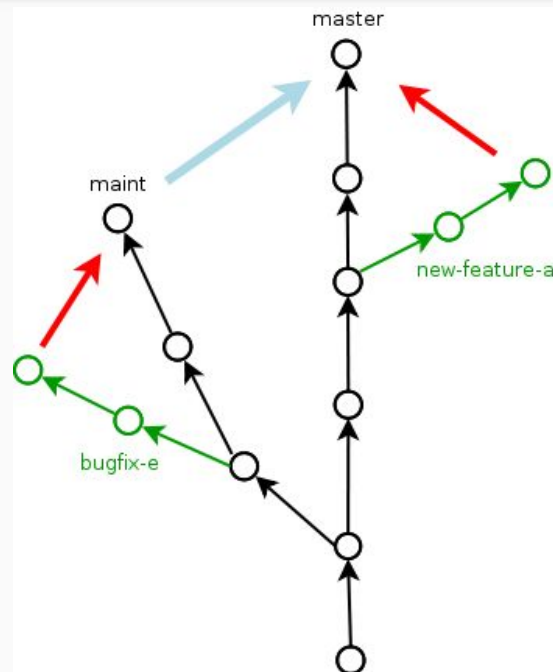
# Git

- Создавался для разработки линукса
- Распределённая система
- Терминология
  - Commit
  - Branch
  - Push/Pull
  - Merge
  - Clone



**git**

# Процесс разработки в git



# Команды

- config
- init
- clone
- add
- commit
- status
- remote
- checkout
- branch
- push
- pull
- merge
- diff
- reset
- tag
- log

# GitHub

- Упрощённый интерфейс
- Создадим репозиторий



# Примитивные типы

Тип	Размер (бит)	Диапазон
byte	8 бит	от -128 до 127
short	16 бит	от -32768 до 32767
char	16 бит	беззнаковое целое число, представляющее собой символ UTF-16 (буквы и цифры)
int	32 бит	от -2147483648 до 2147483647
long	64 бит	от -9223372036854775808L до 9223372036854775807L

# Примитивные типы с плавающей точкой

Тип	Размер (бит)	Диапазон
float	32	от 1.4e-45f до 3.4e+38f
double	64	от 4.9e-324 до 1.7e+308



# boolean

Тип	Размер (бит)	Значение
boolean	8 (в массивах), 32 (не в массивах используется int)	true (истина) или false (ложь)

# Арифметические операторы

Оператор	Описание
+	Складывает значения по обе стороны от оператора
-	Вычитает правый операнд из левого операнда
*	Умножает значения по обе стороны от оператора
/	Оператор деления делит левый операнд на правый операнд
%	Делит левый операнд на правый операнд и возвращает остаток
++	Инкремент - увеличивает значение операнда на 1
--	Декремент - уменьшает значение операнда на 1

# Операторы сравнения

Оператор	Описание
==	Проверяет, равны или нет значения двух операндов, если да, то условие становится истинным
!=	Проверяет, равны или нет значения двух операндов, если значения не равны, то условие становится истинным
>	Проверяет, является ли значение левого операнда больше, чем значение правого операнда, если да, то условие становится истинным
<	Проверяет, является ли значение левого операнда меньше, чем значение правого операнда, если да, то условие становится истинным
>=	Проверяет, является ли значение левого операнда больше или равно значению правого операнда, если да, то условие становится истинным
<=	Проверяет, если значение левого операнда меньше или равно значению правого операнда, если да, то условие становится истинным

# Побитовые операторы

Оператор	Описание
& (побитовое и)	Бинарный оператор AND копирует бит в результат, если он существует в обоих операндах.
(побитовое или)	Бинарный оператор OR копирует бит, если он существует в любом из операндов.
^ (побитовое логическое или)	Бинарный оператор XOR копирует бит, если он установлен в одном операнде, но не в обоих.
~ (побитовое дополнение)	Бинарный оператор дополнения и имеет эффект «отражения» бит.
<< (сдвиг влево)	Бинарный оператор сдвига влево. Значение левых операндов перемещается влево на количество бит, заданных правым операндом.
>> (сдвиг вправо)	Бинарный оператор сдвига вправо. Значение правых операндов перемещается вправо на количество бит, заданных левых операндом.
>>> (нулевой сдвиг вправо)	Нулевой оператор сдвига вправо. Значение левых операндов перемещается вправо на количество бит, заданных правым операндом, а сдвинутые значения заполняются нулями.

# Логические операторы

Оператор	Описание
&&	Называется логический оператор «И». Если оба операнда являются не равны нулю, то условие становится истинным
	Называется логический оператор «ИЛИ». Если любой из двух операндов не равен нулю, то условие становится истинным
!	Называется логический оператор «НЕ». Использование меняет логическое состояние своего операнда. Если условие имеет значение true, то оператор логического «НЕ» будет делать false

# Операторы присваивания

Оператор	Описание
=	Простой оператор присваивания, присваивает значения из правой стороны операндов к левому операнду
+=	Оператор присваивания «Добавления», он присваивает левому операнду значения правого
-=	Оператор присваивания «Вычитания», он вычитает из правого операнда левый операнд
*=	Оператор присваивания «Умножение», он умножает правый операнд на левый операнд
/=	Оператор присваивания «Деление», он делит левый операнд на правый операнд
%=	Оператор присваивания «Модуль», он принимает модуль, с помощью двух операндов и присваивает его результат левому операнду
<<=	Оператор присваивания «Сдвиг влево»
>>=	Оператор присваивания «Сдвиг вправо»
&=	Оператор присваивания побитового «И» («AND»)
^=	Оператор присваивания побитового исключающего «ИЛИ» («XOR»)
=	Оператор присваивания побитового «ИЛИ» («OR»)

# If

```
if (true) {  
    // Code  
} else {  
    // Another code  
}
```

# switch

```
switch(выражение){  
    case значение :  
        //Операторы  
        break; //необязательно  
    case значение :  
        //Операторы  
        break; //необязательно  
    //Здесь может быть любое количество операторов case.  
    default : //необязательно  
        //Операторы  
}
```



# while

```
while (выражение) {  
    // код цикла  
}
```

# for

```
for (int i=0; i<10; i++) {  
    // Код  
}
```

# foreach

```
for (String arg: args) {  
    // Код  
}
```

# do...while

```
do
{
|    // Код
} while (выражение);
```

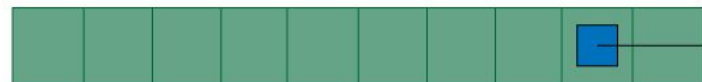
# Массив

```
String[] names = new String[] {"name1", "name2"};
```

Индекс первого элемента

0 1 2 3 4 5 6 7 8 9

Индексы



Элемент массива  
(по индексу 8)

← Массив на 10 элементов →

# Методы

- модификатор
- возвращаемый тип
- имя метода
- аргументы
  - тип
  - имя

```
public static int main(String[] args) {  
    return 0;  
}
```

# Вспомогательные материалы и связь

- <http://proglang.su/java>
- Герберт Шилдт. Java 8. Руководство для начинающих
- <https://t.me/joinchat/MRQ8u6A5inQ2NzU6>