ТЕХНОАТОМ

Java

Основы языка

Сергей Рыбалкин

Немного о себе

ТЕХНОАТОМ

Java – 5+ лет
C++ – 3 года

Люблю backend и зеленые билды.
Не люблю javascript.

s.rybalkin@corp.mail.ru
http://stackoverflow.com/users/6375041
https://github.com/rybalkinsd

Немного о вас

ТЕХНОАТОМ

План лекции

ТЕХНОАТОМ

1. Формат курса
2. Архитектура языка
3. Базовый синтаксис
4. Классы

План лекции

ТЕХНОАТОМ

1. Формат курса
2. Архитектура языка
3. Базовый синтаксис
4. Классы

Формат курса

ТЕХНОАТОМ

- Java intro
- Java web
- Java persistence
- Java client – server communication
- Итоговый проект

Правила игры

ТЕХНОАТОМ

Total: 100

- 3 рубежных контроля = 16 + 12 + 12
- задачи семинаров = 30
- сдача итогового проекта = 30

Сертификаты:

3: 40+

4: 60+

5: 80+

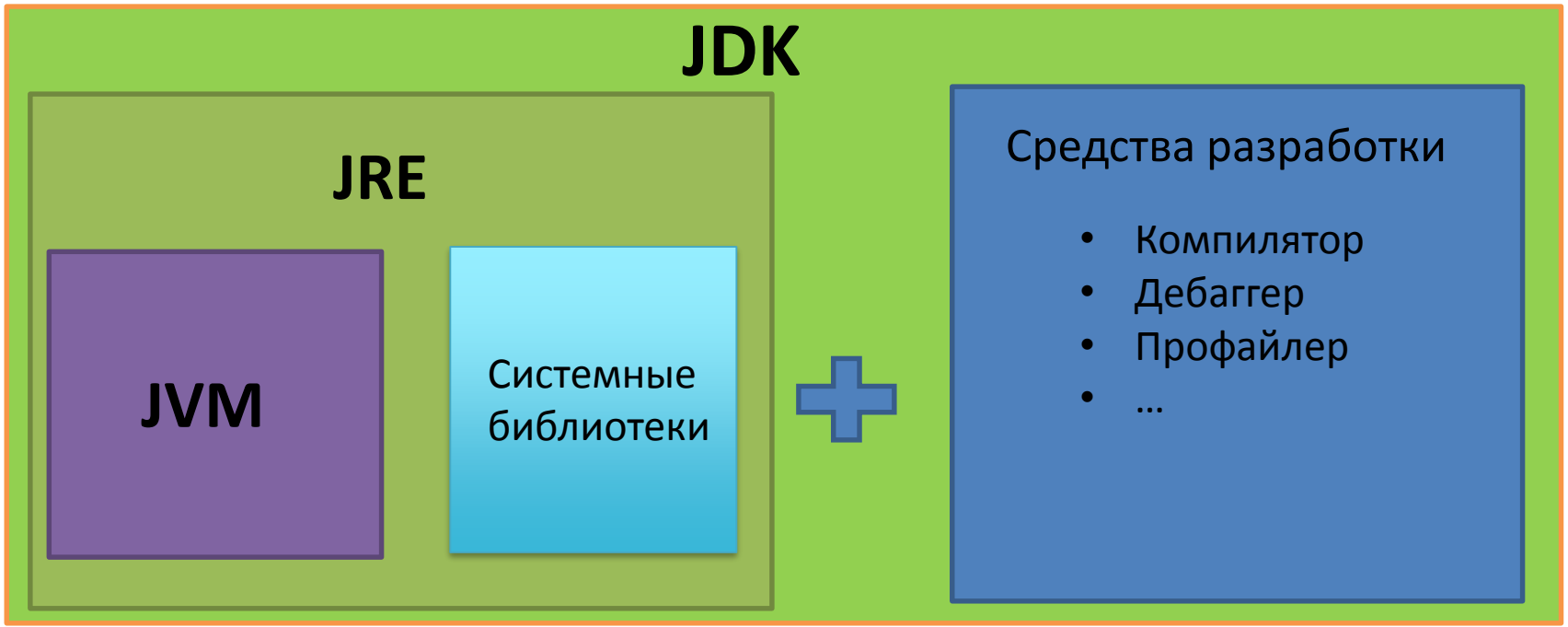
План лекции

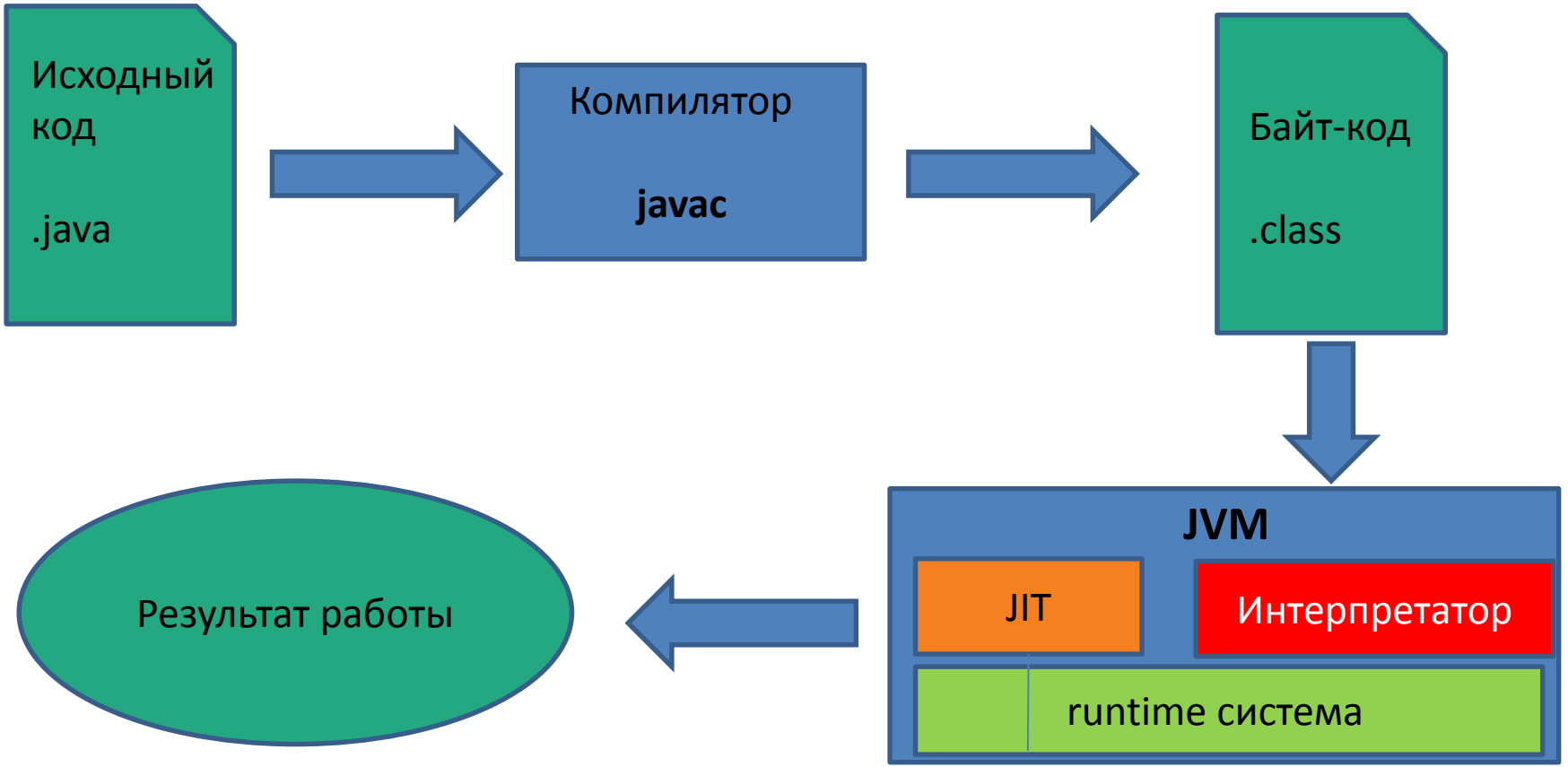
ТЕХНОАТОМ

1. Формат курса
2. Архитектура языка
3. Базовый синтаксис
4. Классы

- Языку Java 20 лет
- Является языком ООП
- Код транслируется не в машинные команды, а в байт-код, который затем выполняет JVM
- Строгая типизация
- Автоматическое управление памятью

Типы распространения




Исходный код
.java
Компилятор
javac
Байт-код
.class
JVM
JIT
Интерпретатор
runtime система
Результат работы

План лекции

ТЕХНОАТОМ

1. Формат курса
2. Архитектура языка
3. Базовый синтаксис
4. Классы

Типы данных

ТЕХНОАТОМ

- Примитивные типы
- Ссылочные типы

Type	Size	Range	Type	Size	Range
boolean	undefined	true/false	int	4 bytes	$-2^{31} – 2^{31}- 1$
byte	1 byte	-128 – 127	long	8 bytes	$-2^{63} – 2^{63} - 1$
char	2 bytes	\u0000 – \uffff	float	4 bytes	IEEE 754
short	2 bytes	-32’768 – 32’767	double	8 bytes	IEEE 754

Операторы

ТЕХНОАТОМ

Operator type	Operator
Assignment	=, +=, *= …^=
Arithmetical	+, -, *, /, %
Relational	<, >, <=, >=, ==, !=
Logical	&&, ||
Bitwise	&, |, ^, >>, <<, >>>
Unary	++, --, +, -, !
Relational2	instanceof

Выражения и блоки

ТЕХНОАТОМ

```
1. int value = 0;

2. array[0] = 100;

3. System.out.println("Hello, world!");

4. int result = 1 + 2;

5. if (value1 == value2)
     System.out.println("value1 == value2");
```

Block defines variable scope

```
1. int commonVariable = 0;
2. if (commonVarialbe > -42) { // ← начало блока
3.     int innerVariable = commonVariable + 1;
4.     System.out.println(String.format("Inner variable is %d",
5.                                      innerVariable));
6. } // ← конец блока
7. /*
8.     а здесь innerVariable уже нет
9. */
```

ТЕХНОАТОМ

if – else if – else

```
1.if (18 == yourAge){
2.     // у вас всё хорошо
3.} else if (yourAge > 18){
4.     // бывало и лучше
5.} else {
6.     // ¯\_(ツ)_/¯
7.}
```

ТЕХНОАТОМ

switch - case

```
switch (countOfApple){
    case 1: // у нас есть 1 яблоко
        break;
    case 2: // у нас есть 2 яблока
        break;
    …
    default:
        // прочие случаи
        break;
}
```

```
while (expression) statement

do { statement } while (expression)

for (initialization; termination; increment)
    statement
```

Циклы. for

ТЕХНОАТОМ

```
1.for (int i = 0; i < numberOfObjects; i++) {
2.     // цикл выполнится numberOfObjects раз,
3.     // если numberOfObjects >= 0
4.}
```

```
1.int[] digits = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
2.for (int i : digits ) {
3.     System.out.println("Digit: " + digit);
4.}
```

Циклы. Для хипстеров

ТЕХНОАТОМ

```
1. IntStream.range(0, 10)
2.          .forEach(digit -> System.out.println(digit));
```

```
1. IntStream.range(0, 10)
2.          .forEach(System.out::println);
```

* Java 8, stream, lambda, method reference

ТЕХНОАТОМ

Method signature – method name + argument list.

```
1.public int getCountOfApples(List<Integer> boxes,
2.                            Integer[] numberOfBoxes)
3.        throws Throwable {

4.    Integer sumOfApples = 0;
5.    for (Integer i : numberOfBoxes) {
6.        sumOfApples += boxesWithApples.get(i);
7.    }
8.    return sumOfApples;
9.}
```

Методы

ТЕХНОАТОМ

- Access modifier → public
- Return type → int
- Method name → getCountOfApples
- Parameter list → (…)
- Exception list → Throwable
- Method body → { … }

Методы. Overload

ТЕХНОАТОМ

```
1. void sayDigit(int digit){
2.     System.out.println(String.format("The digit is %d", digit));
3. }
4.
5. void sayDigit(float digit){
6.     System.out.println(String.format("The digit is %f", digit));
7. }
```

Перегрузка может осуществляться только по набору аргументов.

План лекции

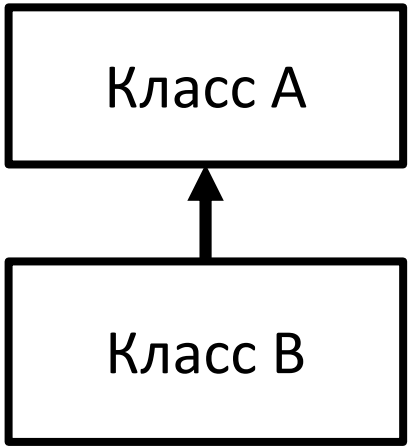
ТЕХНОАТОМ

1. Формат курса
2. Архитектура языка
3. Базовый синтаксис
4. Классы

Everything is an object.
Nothing outside of a class.

```
1.class Clazz extends SuperClass
2.                implements YourInterface {
3.
4.    private int id;
5.    public Clazz() { … }
6.}
```

Классы. Наследование




```
1. class A{
2.     // поля, конструкторы,
3.     // методы и блоки инициализации
4. }
```

```
1. class B extends A{
2.     // поля, конструкторы,
3.     // методы и блоки инициализации
4. }
```

Отношение Is A

Классы. Наследование

ТЕХНОАТОМ

```
1. class A {
2.     int i;
3.     public A(int i){
4.         this.i = i;
5.     }
6.     public int getI(){
7.         return i;
8.     }
9. }
```

```
1. class B extends A {
2.     int i;
3.     public B(int i){
4.        super(i);
5.        this.i = i / 2;
6.     }
7.     @Override
8.     public int getI(){
9.         return this.i;
10.    }
11.    public int getSuperI(){
12.        return super.i;
13.    }
14.}
```

Классы. Object

ТЕХНОАТОМ

java.lang.Object is a superclass for all classes.

```
1. class Object {
2.     protected Object clone() throws CloneNotSupportedException

3.     public boolean equals(Object obj)
4.
5.     protected void finalize() throws Throwable

6.     public final Class getClass()

7.     public int hashCode() {//some logic }

8.     public String toString()
9.     …
10.}
```

```
1. class Human extends Animal {
2.     public static short AVERAGE_HEIGHT = 170;
3.     public static final long COUNT_OF_POPULATION;
4.     static {
5.         COUNT_OF_POPULATION = 7_000_000_000;
6.     }
7.     private int luckyNumber;
8.
9.     private Human(int myLuckyNumber){
10.         this.luckyNumber = myLuckyNumberl;
11.     }
12.
13.     public Human(int[] luckyNumbersCandidats){
14.         this(selectLuckyNumber(luckyNumbersCandidats));
15.     }
16. }
```

Классы. Модификаторы

ТЕХНОАТОМ

- **static**
- **final**
- **abstract**
- **default**
- **transient –** маркирует поля, которые не будут сериализоваться
- **volatile –** гарантирует атомарность операций чтения/записи
- **synchronized –** обеспечивает синхронизацию выполнения блоков кода
- **native –** метод реализован в нативном коде

Классы. Абстрактный класс

ТЕХНОАТОМ

```
1. abstract class AbstractBird {
2.     private int age;
3.     public AbstractBird(int age) {
4.         this.age = age;
5.     }

6.     public abstract void sayHi();}
7. }

8. new AbstractBird() {
9.     @Override
10.    public void sayHi() {
11.        // payload
12.    }
13.}
```

One class – one superclass.
One class – many interfaces.

```
1.class B extends A
2.         implements Writable, Readable, Mutable {
3.
4.     // payload
5.}
```

```
1. interface Talking {
2.     void sayHello();
3.
4.     default void sayHi(){
5.         System.out.println(“Hi!”);
6.     }
7. }

8. class Parrot implements Talking {
9.     public void sayHello(){
10.        System.out.println(“Hello!”);
11.    }
12.}
```

	Interface	Abstract class
Inheritance	Implement many	Extend one
Fields	public static only	no limits
Access modifiers	public only	no abstract private methods
Constructor	no constructors	no limits

ТЕХНОАТОМ

Спасибо за внимание!

Сергей Рыбалкин

s.rybalkin@corp.mail.ru