


Нижегородский государственный университет им. Н.И. Лобачевского

Институт информационных технологий, математики и механики

Нижегородский государственный университет им. Н.И. Лобачевского

Институт информационных технологий, математики и механики

Введение в Java

Обзор возможностей Java. Компиляция и запуск.

Козинов Е.А.
Кафедра МОСТ

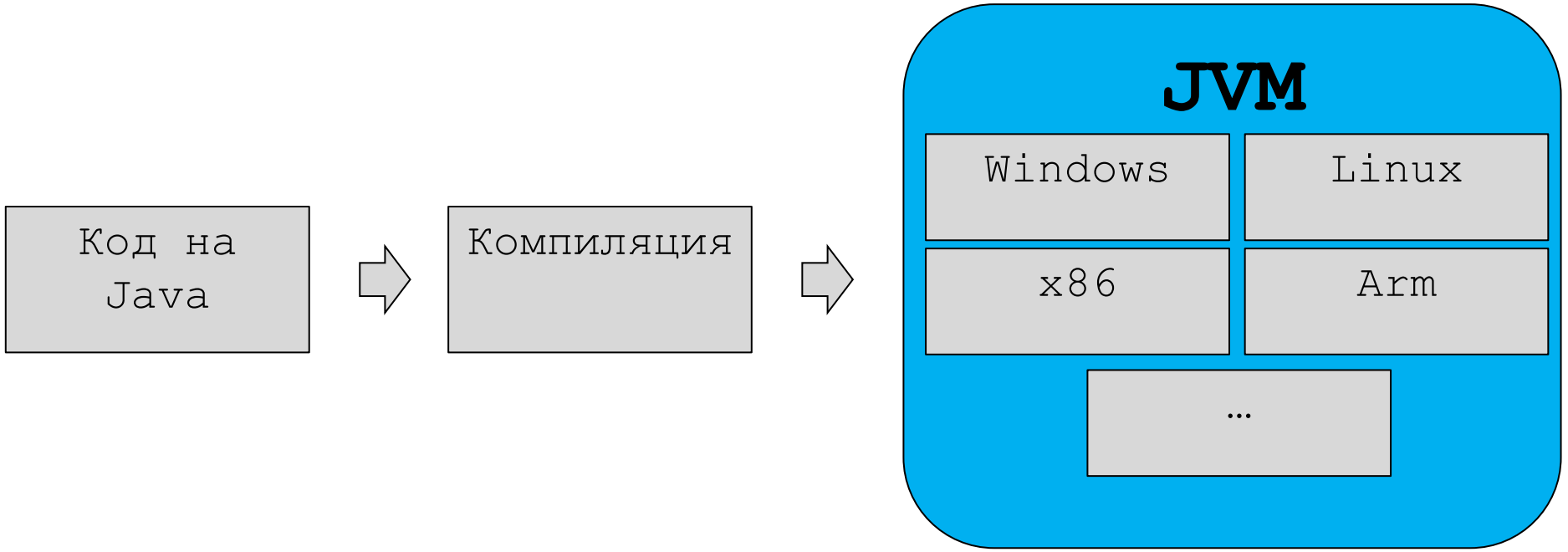
Содержание

- Назначение Java
- Пример программы «Hello» в сравнении с C++
- Компиляция программ на Java
- Системы сборки приложений
  - Ant
  - Maven
- Ввод/Вывод

НАЗНАЧЕНИЕ JAVA

Зачем нужна java? (некоторые преимущества)

- Основное преимущество Java – **переносимость** (платформенная независимость)
  - Переносимость обеспечена за счет наличия **Java Virtual Machine (JVM)**
  - Нет необходимости в повторной компиляции приложений при смене операционной системы или типа устройства



Как исполняются приложения

- Код на C/C++

Набор файлов:
***.cpp и *.h**

↓

Компилятор:
cl, icl, gcc

↓

Набор файлов:
***.exe, *.dll, *.lib**

↓

Способ запуска:
***.exe**

↓

Исполнение на CPU:
***.exe, *.dll**

- Код на Java

Набор файлов:
***.java**

↓

Компилятор:
javac

↓

Набор файлов:
***.class, *.jar**

↓

Способ запуска:
java <опции> <имя класса>

↓

Исполнение на CPU:
java (JVM)

Зачем нужна java? (некоторые преимущества)

- Процесс разработки приложений упрощен
  - Более «**жесткая**» **проверка кода** на этапе компиляции (инициализация переменных, проверка исключений, т.д.)

```
// Java
int a, b, c;
c = a + b; // Ошибка компиляции
           // (в C/C++, как правило, предупреждение)
```

```
// Java
int a, b, c;
a = 10;
b = 20;
c = a + b; // Нет ошибки компиляции
```

Зачем нужна java? (некоторые преимущества)

- Процесс разработки приложений упрощен
  - Динамическую **память контролирует JVM**
    - Нет «утечек» памяти

```
// C++
int * getArray()
{
  int * a = new int [10];
  return a;
}

…

int b = getArray();
delete[] b; // часто забывают
```

Зачем нужна java? (некоторые преимущества)

- Процесс разработки приложений упрощен
  - **Контроль за доступом к памяти** во время исполнения

```
// Java
  int a[];
  a = new int [10];
  a[10] = 0;
```

  - Результат исполнения

```
Exception in thread "main"
   java.lang.ArrayIndexOutOfBoundsException:
   Index 10 out of bounds for length 10
      at example.maven.mainClass.main(mainClass.java:36)
```

Зачем нужна java? (некоторые преимущества)

- Процесс разработки приложений упрощен
  - В java есть **большая библиотека классов**
    - Реализованы базовые структур данных
    - Упрошенная разработка визуальных приложений (JavaFX, Swing)
    - Большое количество открытых библиотек https://mvnrepository.com/
    - …
  - Удобное управление многопоточными приложениями
    - В последствии заимствовано многими языками
  - Большое число технологий разработки сетевых приложений
  - Поддержка обобщений (шаблонов)
  - Поддержка Лямбда-выражений
  - Большое сообщество разработчиков

Назначение языков программирования (субъективная точка зрения)

- Назначение Perl, Python
  - Автоматизация рутинных процессов (запуск тестов, сбор статистики,…)
- Назначение C/C++, Fortran
  - Высокопроизводительные вычисления
- Назначение SQL
  - Доступ к базам данных
- Назначение Java, C#, …
  - Реализация многофункциональных программных систем и комплексов, предназначенные для автоматизации ключевых бизнес-функций и процессов внутри компании.
- **ВАЖНО**
  - Решение задач может требовать нескольких языков программирования
  - Части программных комплексов могут быть реализованы на разных языках программирования

Некоторые недостатки Java

- JVM должна быть реализована для каждой платформы
  - **Корректно (!)**
  - **Эффективно (!)**
- Исполняемы файлы «интерпретируется»
  - Как следствие исполнение медленней бинарного кода
- JVM пытается минимизировать разницу в скорости исполнения за счет подготовки байткода к исполнению
  - Перед исполнением подготовленный код «компилируется» под конкретную платформу (JIT-компиляция)

Основные семейства Java

- **Java SE** (Standard Edition) - содержит компиляторы, основные API, Runtime;
  - подходит для создания большинства пользовательских приложений
  - ***Будем использовать в обучении***
- Java EE (Enterprise Edition) - набор спецификаций для создания распределенных и масштабируемых приложений
  - сейчас переименован в Jakarta EE
- Java ME (Micro Edition) – версия Java для устройств с малой вычислительной мощностью
- Java Card - среда работы со смарт-картами и другими устройствами с очень ограниченным объёмом памяти и возможностями обработки
- Android SDK – содержит специальную версию Java для смартфонов

ПРИМЕР ПРОГРАММЫ НА JAVA

Пример программы «Hello world!»

- Пример на C++

```
#include <iostream>

int main(int argc, char **argv)
{
    std::cout << "Hello world!";
    return 0;
}
```

Точка входа в программу

В Java нет функций

- Пример на Java

```
public class Main {
    public static void main(String[] argv) {
        System.out.println("Hello world!");
    }
}
```

Поток вывода

КОМПИЛЯЦИЯ ПРОГРАММ НА JAVA

Возможные способы сборки приложений

- Использовать компилятор Java
- Использовать системы сборки приложений (например, в C/C++ можно использовать CMake)
  - Ant
    - Описание процесса сборки и запуска на локальном компьютере
  - Maven
    - Описание процесса сборки и запуска с использование репозитория открытых библиотек https://mvnrepository.com/ (наиболее востребованный вариант)
- Все варианты требую установки JDK
  - https://www.oracle.com/technetwork/java/javase/downloads/index.html

Сборка и запуск приложения стандартными средствами

- Компиляция

```
D:\temp\2020\JavaHello>dir
25.01.2020  14:29                194 Main.java

D:\temp\2020\JavaHello>javac.exe Main.java

D:\temp\2020\JavaHello>dir
25.01.2020  14:33               414 Main.class
25.01.2020  14:29                194 Main.java
```

- Запуск

```
D:\temp\2020\JavaHello>java.exe Main
Hello world!
```

Система сборки Maven

- Структура директорий

```
Maven_Example\pom.xml
Maven_Example\src\main\java\example\example_maven\Main.java
```

- Исходный код

```
package example.example_maven;

public class Main {
    public static void main(String[] argv) {
        System.out.println("Hello world");
    }
}
```

Система сборки Maven

- pom.xml (содержит описание исходных кодов)

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns=http://maven.apache.org/POM/4.0.0
         xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
         http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>example</groupId>
  <artifactId>Example_Maven</artifactId>
  <version>1.0</version>
  <packaging>jar</packaging>
  <properties>
   <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
   <maven.compiler.source>1.8</maven.compiler.source>
   <maven.compiler.target>1.8</maven.compiler.target>
  </properties>
</project>
```

Система сборки Maven

- Компиляция приложения

```
set JAVA_HOME=C:\Program Files\Java\jdk-12.0.2
mvn package
[INFO] Scanning for projects...
[INFO] ---------------------------
[INFO] Building Example_Maven 1.0
[INFO] ---------------------------
[INFO] --- maven-resources-plugin:2.6:resources
[INFO] --- maven-compiler-plugin:3.1:compile
[INFO] --- maven-resources-plugin:2.6:testResources
[INFO] --- maven-compiler-plugin:3.1:testCompile
[INFO] --- maven-surefire-plugin:2.12.4:test
[INFO] --- maven-jar-plugin:2.4:jar
[INFO] Building …\target\Example_Maven-1.0.jar
[INFO] ---------------------------
[INFO] BUILD SUCCESS
```

Система сборки Maven

- Запуск приложения

```
D:\temp\2020\Maven_Example>mvn exec:java
-Dexec.mainClass="example.example_maven.Main"
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] Building Example_Maven 1.0
[INFO] ------------------------------------------------------------------------
[INFO]
[INFO] --- exec-maven-plugin:1.6.0:java (default-cli) @ Example_Maven ---
Hello world
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
```

ВВОД/ВЫВОД

Потоки ввода/вывода

- В Java есть три потока работы с консолью
  - System.in – ввод
  - System.out – вывод
  - System.err – вывод ошибок
- Потоки имеют стандартный интерфейс
- Стандартный интерфейс System.in (основные методы)
  - int read(byte[])
  - int read(byte[], int offset, int length)
  - Интерфейс позволяет работать с потоком байт.

Расширение интерфейса потоков

- Возможности потоков можно расширить
  - Для того используются специальные контейнеры (декораторы)

```
public static void main(String[] args)
    throws IOException {
  String name;
  DataInputStream dis = new DataInputStream(System.in);
  name = dis.readLine();

  System.out.println("Hello, " + name + "!");
  // также вывод можно реализовать следующим образом
  DataOutputStream dos = new DataOutputStream(System.out);
  dos.writeUTF("Hello, " + name + "!");

}
```

Суммирование двух чисел

- Первый вариант кода

```
public static void main(String[] args)
    throws IOException {
  String s;
  int a, b, c;
  DataInputStream dis = new DataInputStream(System.in);
  s = dis.readLine();
  a = Integer.parseInt(s);
  s = dis.readLine();
  b = Integer.parseInt(s);
  c = a + b;
  System.out.println(a + " + " + b + " = " + c);
}
```

  - Какие могут возникнуть проблемы в коде?
  - Что можно улучшить?

Суммирование двух чисел

- Второй вариант кода

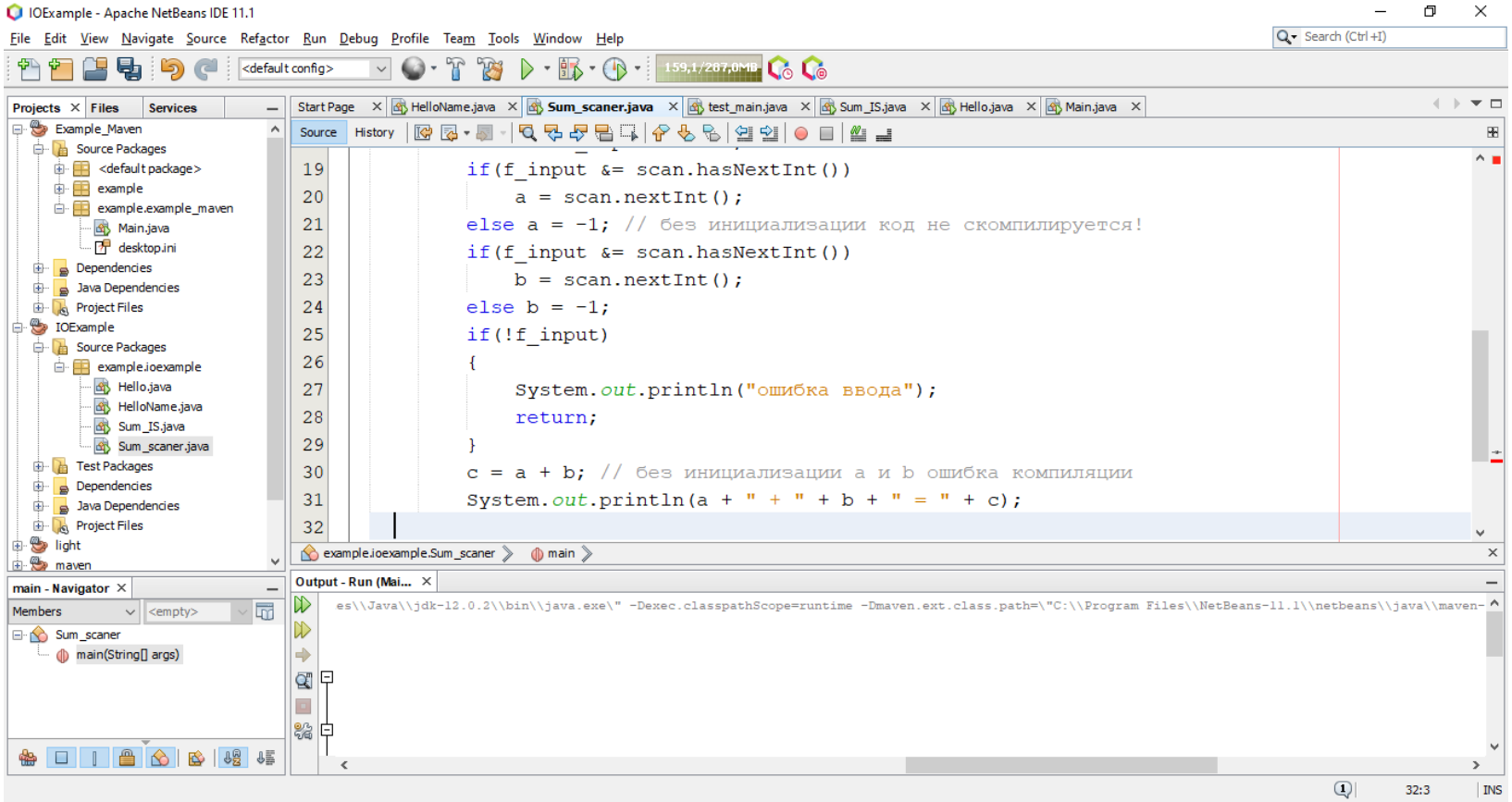
```
public static void main(String[] args) {
    int a, b, c;
    Scanner scan = new Scanner(System.in);
    boolean f_input = true;
    if(f_input &= scan.hasNextInt())
        a = scan.nextInt();
    if(f_input &= scan.hasNextInt())
        b = scan.nextInt();
    if(!f_input) {
        System.out.println("ошибка ввода"); return;
    }
    c = a + b; // ОШИБКА КОМПИЛЯЦИИ!
    System.out.println(a + " + " + b + " = " + c);
}
```

Суммирование двух чисел

- Третий вариант кода

```
public static void main(String[] args) {
    int a, b, c;
    Scanner scan = new Scanner(System.in);
    boolean f_input = true;
    if(f_input &= scan.hasNextInt())
        a = scan.nextInt();
    else a = -1; // без инициализации код не скомпилируется!
    if(f_input &= scan.hasNextInt())
        b = scan.nextInt();
    else b = -1;
    if(!f_input) {
        System.out.println("ошибка ввода"); return;
    }
    c = a + b; // ошибки компиляции нет!
    System.out.println(a + " + " + b + " = " + c);
}
```

ДЕМОНСТРАЦИЯ РАЗРАБОТКИ ПРИЛОЖЕНИЙ В СРЕДЕ NETBEANS



ЗАКЛЮЧЕНИЕ

Заключение

- В лекции рассмотрены основные понятия связанные с Java
- В лекции рассмотрен простейший пример на Java, а также способы компиляции и запуска приложений
- Для прохождения практики рекомендуется
  - Установить JDK
  - Установить одну из сред разработки
    - NetBeans https://netbeans.org/
    - IntelliJ IDEA https://www.jetbrains.com/ru-ru/idea/

Литература

1. Программирование на Java - http://www.intuit.ru/studies/courses/16/16/info
2. Построение распределенных систем на Java - http://www.intuit.ru/studies/courses/633/489/info
3. Дистрибутивы средств разработки ПО - http://www.oracle.com/technetwork/java/javase/downloads/index.html
4. Официальная документация по языку программирования Java - https://docs.oracle.com/javase/tutorial/

Контакты

Нижегородский государственный университет
http://www.unn.ru

Институт информационных технологий, математики и механики
http://www.itmm.unn.ru

Козинов Е.А.
evgeny.kozinov@itmm.unn.ru