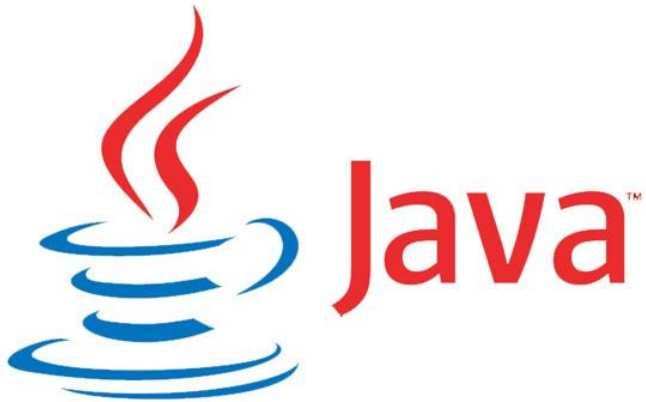
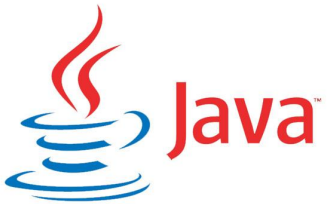
# Тема 1.
# Введение в язык программирования Java.

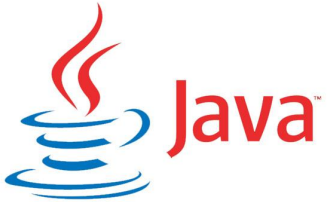


# История




- Начало разработке языка было положено в 1991 г. Джеймсом Гослингом и его напарниками Патриком Ноутоном, работавшими в компании Sun Microsystems, Inc.
- Изначально язык создавался для программирования бытовых электронных устройств.
- Разработка первой работающей версии заняла 18 месяцев.
- Первым названием языка было Oak (Дуб).
- В 1995 был переименован в Java.
- 23 мая 1995 г. - Компания Sun официально представила Java на выставке SunWorld '95.

# Причины создания




Патрик Ноутон, Джеймс Гослинг  работали над проектом "Green", целью которого было разработать язык для программирования бытовых электронных устройств (например, контроллеров, для переключения каналов телевидения).

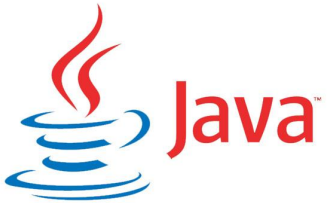
Бытовые устройства потребляют мало энергии (отсюда и кодовое название Green) и имеют небольшие микросхемы памяти.

Следовательно, программы, написанные для них, должны быть небольшими.

Кроме того, была поставлена задача создания кода, который единожды читается на любой машине, а не пишется под конкретное устройство.

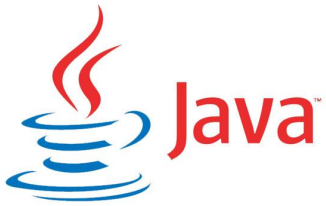
Так и родился Java.

# Основные качества:




- Простота и мощь;
- Безопасность;
- Объектная ориентированность;
- Надежность;
- Интерактивность;
- Архитектурная независимость;
- Возможность интерпретации;
- Высокая производительность;
- Легкость в изучении.

# Простота и мощь

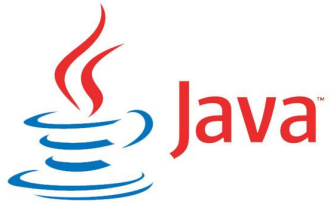



В языке Java для решения задачи имеется совсем немного вариантов.

Стремление к простоте зачастую приводило к созданию неэффективных и невыразительных языков типа командных интерпретаторов.

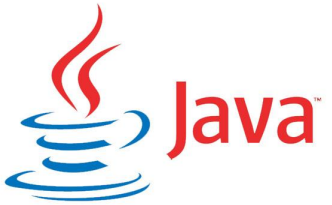
Java к числу таких языков не относится – для Вас вся мощность ООП и библиотек классов.

# Надежность




- В Java накладывается ряд ограничений в нескольких наиболее важных областях, что вынуждает разработчиков выявлять ошибки на ранних этапах создания программы.
- Java избавляет от необходимости беспокоится по поводу наиболее часто встречающихся ошибок программирования
- Java строго типизированный язык, и проверка кода выполняется во время компиляции
- Проверка кода в Java выполняется и во время выполнения программы, в результате чего многие трудно обнаруживаемые программные ошибки, часто приводящие к с трудом воспроизводимым ситуациям, попросту невозможны
- Предсказуемость кода в разных ситуациях, одна из особенностей Java

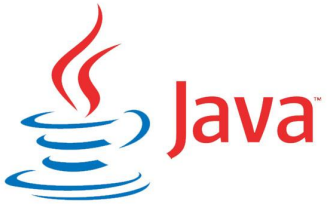
# Независимость от архитектуры




Основной задачей, которую ставили перед собой разработчики Java, было обеспечение долговечности и переносимости кода.

Одной из главных трудностей, стоявших перед разработчиками, когда они создавали Java, было отсутствие всяких гарантий что код, написанный сегодня, будет успешно выполняться завтра - даже на одном и том же компьютере.

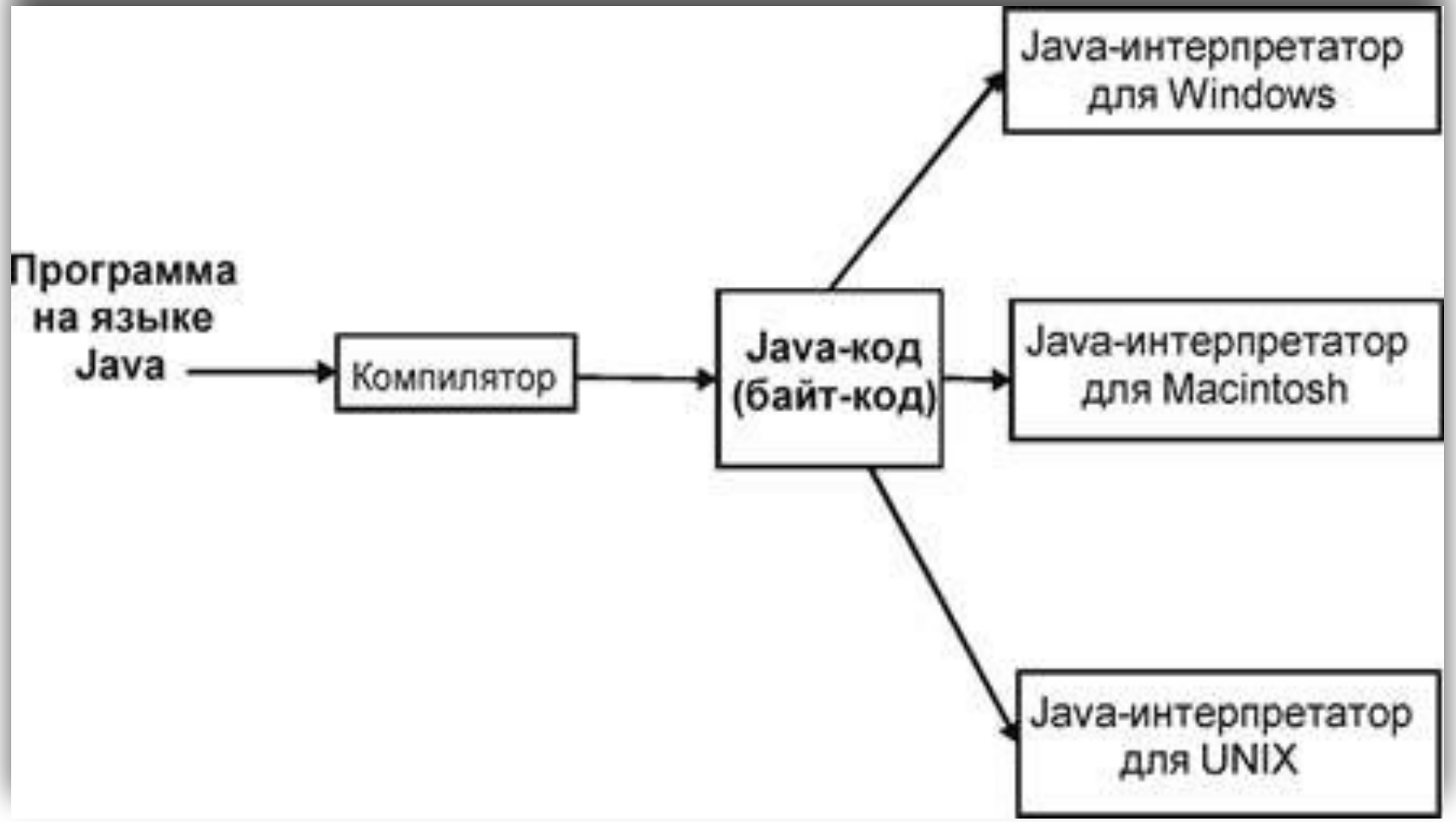
# Независимость от архитектуры



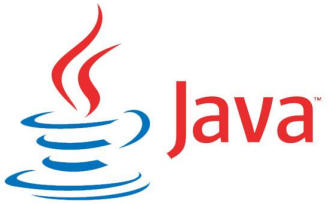

Операционные системы и процессоры постоянно совершенствуются, и любые изменения в основных системных ресурсах могут стать причиной неработоспособности программ.

Пытаясь каким-то образом изменить это положение, разработчики приняли ряд жестких решений в самом языке и виртуальной машине Java. Они поставили перед собой следующую цель: "**написано однажды, выполняется везде, в любое время и всегда**". И эта цель была в значительной степени достигнута.

# Java Virtual Machine



# Java Virtual Machine




**Виртуальная машина** - это программное обеспечение, основанное на понятиях и идее воображаемого компьютера, который имеет логический набор команд, определяющих операции этого компьютера.

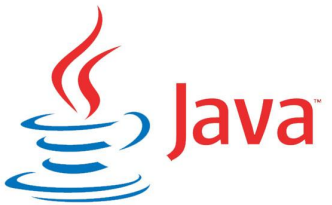
Это, можно сказать, небольшая операционная система.

Она формирует необходимый уровень абстракции, где достигается независимость от платформы и используемого оборудования.

Исполняет **байт-код Java**, предварительно созданный из исходного текста Java-программы компилятором Java (**javac**).

**JVM** может также использоваться для выполнения программ, написанных на других языках программирования. Например, исходный код на языке Ada (названный в честь первой программистки - Ады Лавлэйс) может быть откомпилирован в **байт-код** Java, который затем может выполниться с помощью **JVM**.

# Java Virtual Machine



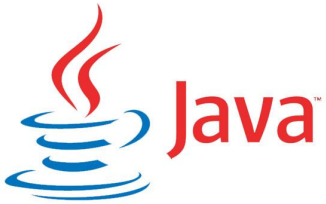

JVM является ключевым компонентом платформы Java.

Так как виртуальные машины Java доступны для многих аппаратных и программных платформ, Java может рассматриваться и как связующее программное обеспечение, и как самостоятельная платформа. Использование одного байт-кода для многих платформ позволяет описать Java как **«скомпилировано однажды, запускается везде»** (compile once, run anywhere).

Виртуальные машины Java обычно содержат **Интерпретатор** байт-кода, однако, для повышения производительности во многих машинах также применяется **JIT (Just-in-time)** - компиляция часто исполняемых фрагментов байт-кода в машинный код.

# Интерпретация

# Интерпретация

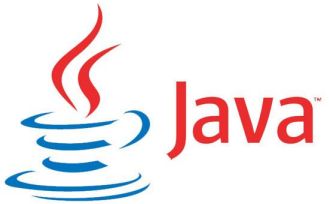



Необычайная способность Java исполнять свой код на любой из поддерживаемых платформ достигается тем, что ее программы транслируются в некое промежуточное представление, называемое **байт-кодом (bytecode)**.

Байт-код, в свою очередь, может интерпретироваться в любой системе, в которой есть среда времени выполнения Java.

Большинство ранних систем, в которых пытались обеспечить независимость от платформы, обладало огромным недостатком — потерей производительности (Basic, Perl). Несмотря на то, что в Java используется интерпретатор, байт-код легко переводится непосредственно в “родные” машинные коды (Just In Time compilers) “на лету”. При этом достигается весьма высокая производительность.

# Объектная ориентированность

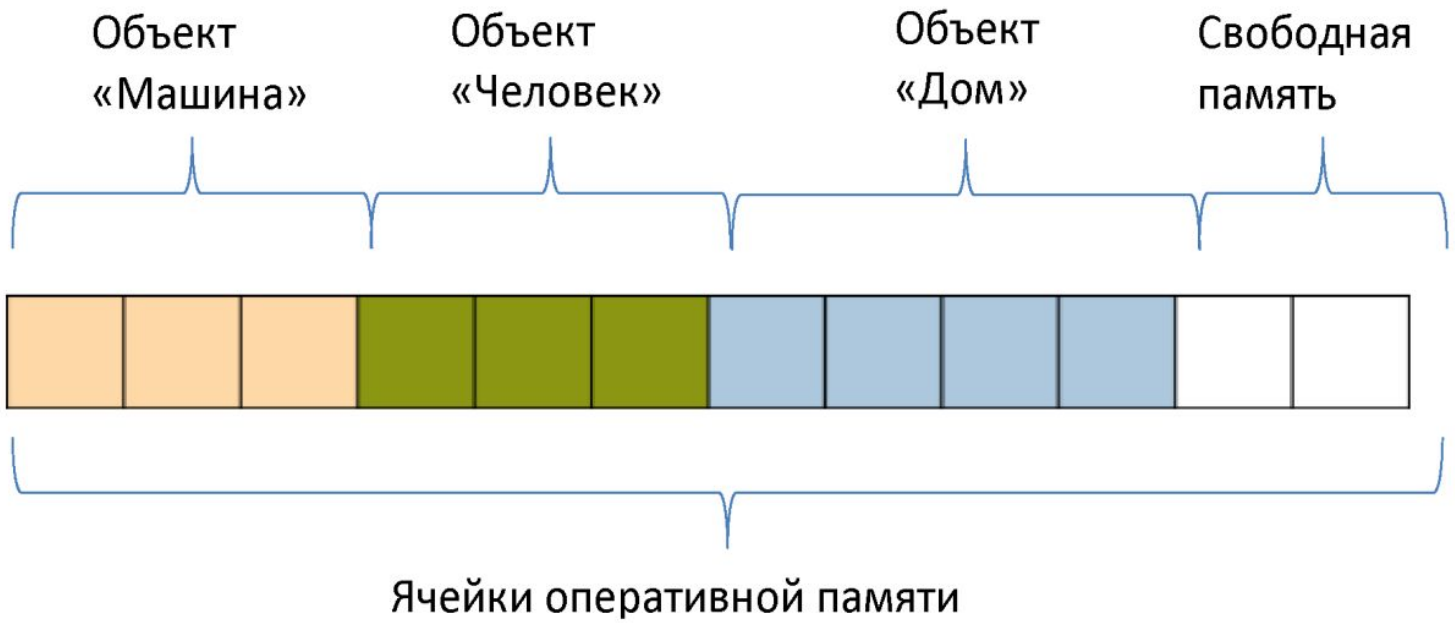



Java выделяется четким, практичным и прагматичным подходом к объектам.

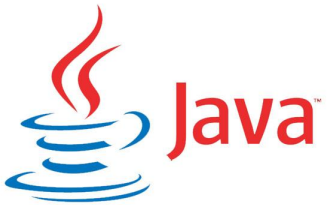
Объектная модель Java проста и легко расширяема.

В то же время такие элементарные типы данных, как целочисленные, сохраняются в виде высокопроизводительных компонентов, не являющихся объектами.

# Организация оперативной памяти в Java

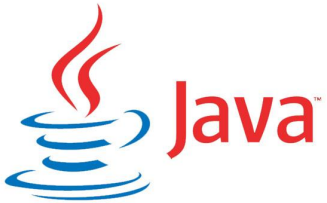


# Организация оперативной памяти в Java



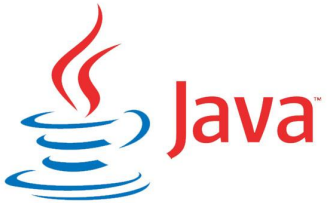

- В C++ есть прямой доступ к памяти, в Java – нет;
- Виртуальная машина Java самостоятельно управляет объектами в памяти;
- Сборщик мусора избавляет программиста Java от проблем управления памятью вручную. Он удаляет старые объекты, на которых уже нет ссылок в программе. Периодичность его работы динамична и заранее неизвестна;
- Запись объектов в память происходит при выполнении программы, а не при написании кода.

# Безопасность




Программы на Java не могут вызывать глобальные функции и получать доступ к произвольным системным ресурсам, что обеспечивает в Java уровень безопасности, недоступный для других языков.

# Простота изучения

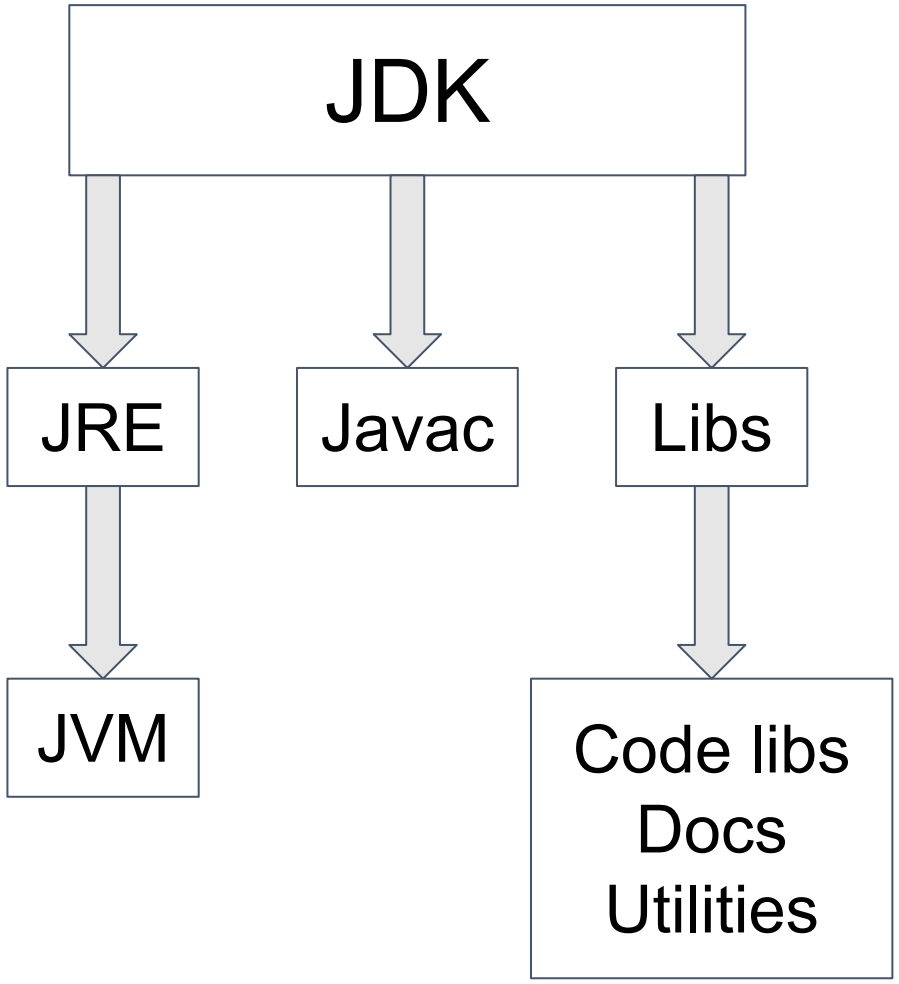



Объектная модель в Java одновременно проста и выразительна, что позволяет быстро освоиться с объектно-ориентированным стилем создания программ.

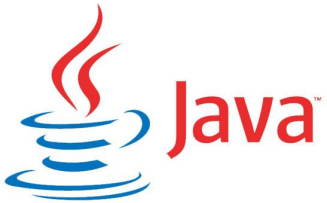
Среда Java содержит встроенный набор ключевых классов, содержащих основные абстракции реального мира, с которыми будет иметь дело Ваша программа.

Основой популярности Java стали классы-абстракции, сделавшие его языком, действительно независимым от платформы.

# Компоненты Java



# Компоненты Java

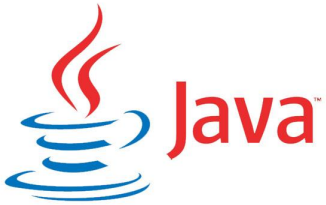



**JDK** (Java Development Kit) - комплект разработчика приложений на языке Java, включающий в себя компилятор, стандартные библиотеки классов Java, примеры, документацию, различные утилиты и исполнительную систему JRE.

**JRE** (Java Runtime Environment) - минимальная реализация виртуальной машины, необходимая для исполнения Java -приложений, без компилятора и других средств разработки. Состоит из виртуальной машины и библиотек Java классов.

**JVM** (Java Virtual Machine) - виртуальная машина Java - основная часть исполняющей системы Java, так называемой Java Runtime Environment (JRE). Виртуальная машина Java исполняет байт-код Java, предварительно созданный из исходного текста Java-программы компилятором Java (javac). JVM обеспечивает платформо-независимый способ выполнения кода. Программисты могут писать код не задумываясь как и где он будет выполняться.

# Основные библиотеки




**java.lang** - Классы ядра языка (типы, работа со строками, тригонометрические функции, обработка исключений)

**java.io** - Классы для различных типов ввода-вывода

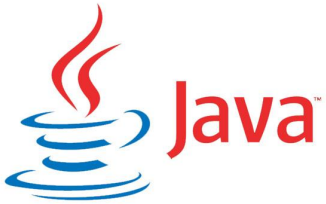
**java.system** - Классы для работы с системой, консолью/командной строкой и т.д.

**java.math** - Классы для арифметических операций произвольной точности

**java.net** - Классы для работы в сети Интернет (сокеты, протоколы, URL)

**java.util** - Разноообразные полезные типы данных (стеки, словари, хэш-таблицы), даты, генератор случайных чисел

# Классификация платформ Java




- **Java SE** — Java Standard Edition, основное издание Java, содержит компиляторы, API, Java Runtime Environment; подходит для создания пользовательских приложений, в первую очередь — для настольных систем.
- **Java EE** — Java Enterprise Edition, представляет собой набор спецификаций для создания программного обеспечения уровня предприятия.
- **Java ME** — Java Micro Edition, создана для использования в устройствах, ограниченных по вычислительной мощности, например, в мобильных телефонах, КПК, встроенных системах;

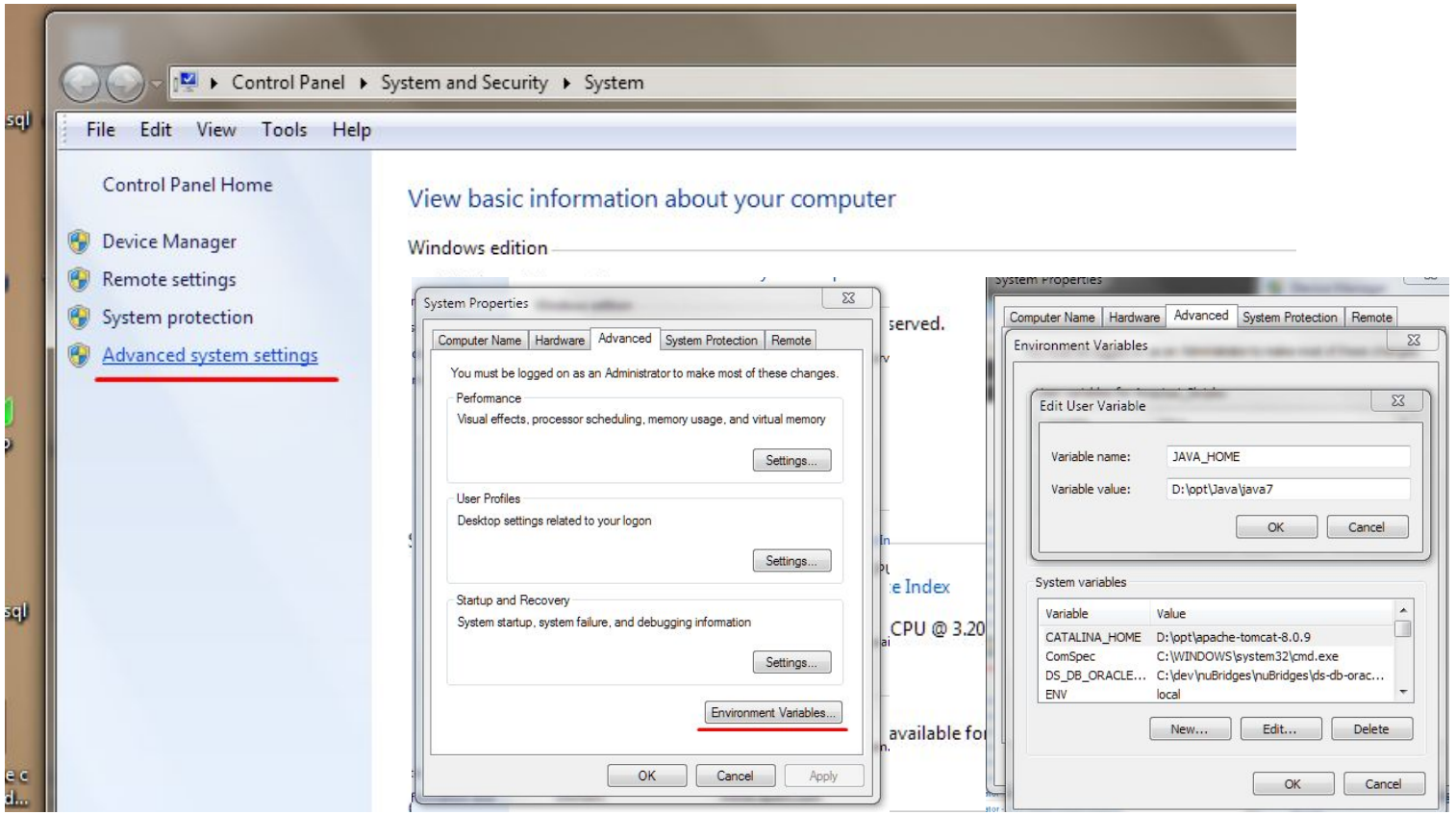
# Установка Java

Скачать Java с официального сайта https://www.oracle.com/technetwork/java/javase/downloads/index.html


| | | |
|---|---|---|
| Windows x64 Installer | 162.11 MB | jdk-14.0.2_windows-x64_bin.exe |

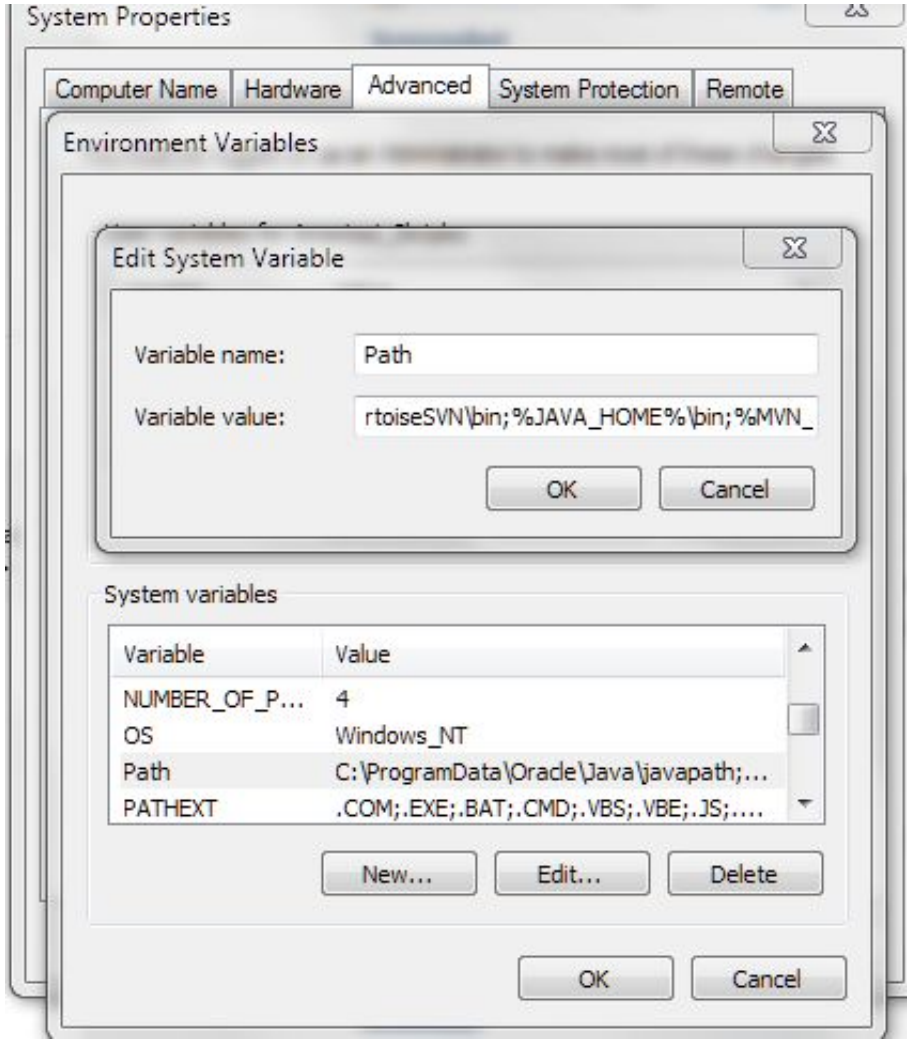
# Установка Java

Создать новую переменную среду с именем JAVA_HOME, которая будет ссылаться на директорию, в которую была произведена установка java.

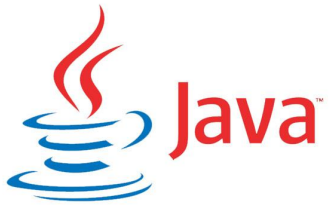


# Установка Java

Добавить переменную JAVA_HOME в Path:



# Установка Java



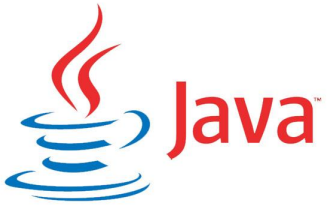

Проверить успешность установки вызовом команды java -version.

```
Microsoft Windows [Version 10.0.17134.885]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>java -version
java version "1.8.0_144"
Java(TM) SE Runtime Environment (build 1.8.0_144-b01)
Java HotSpot(TM) 64-Bit Server VM (build 25.144-b01, mixed mode)

C:\WINDOWS\system32>
```

# Структура программы на Java

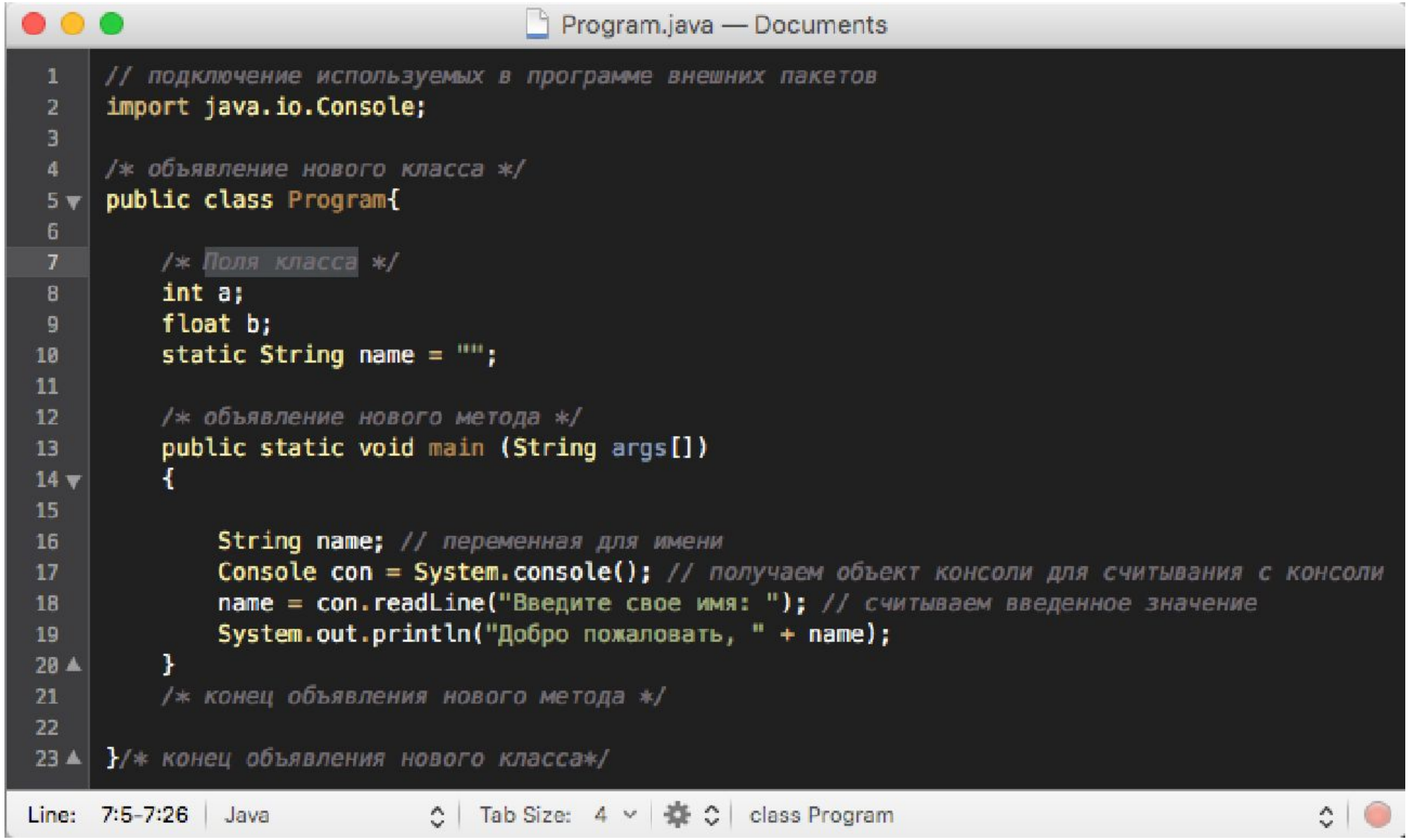



Любая программа на Java состоить из классов, которые, по мере надобности, а иногда и все сразу, загружаются в память JVM.

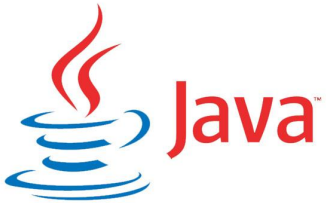
**Классы описываются по определенному шаблону и имеют следующие основные блоки**:

- Область подключения внешних пакетов
- Объявление класса
- Поля класса
- Описание конструктора класса
- Описание методов класса

# Структура программы на Java

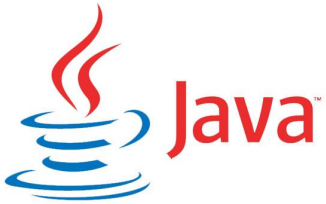


# Структура программы на Java



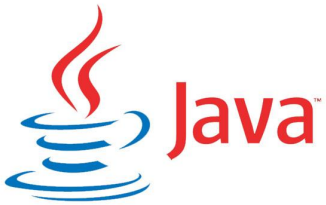

- В начале файла идет секция с подключенными внешними пакетами с помощью директивы import, после которой идут названия подключаемых **пакетов** и классов. **Пакеты** представляют собой организацию классов и интерфейсов в общие группы или блоки.
- Каждая строка завершается **точкой с запятой**, а каждый блок кода помещен в фигурные скобки.
- Далее идет определение класса программы. Классы объявляются следующим образом: модификатор доступа public, указывающий, что данный класс будет доступен из любого места программы и мы сможем запустить его из командной строки; затем идет ключевое слово class, а потом – название класса и тело самого класса в фигурных скобках.

# Структура программы на Java



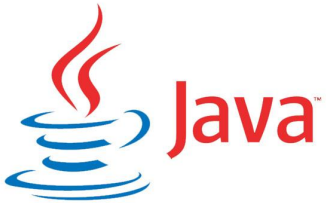

- Классы – это «кирпичики», из который строится программа на Java. К именам классов (идентификаторам) предъявляются определенные требования: именем класса может быть произвольная последовательность алфавитных и цифровых символов, а также символа подчеркивания, однако при этом названия не должны начинаться с цифры; идентификаторы не должны быть представлены зарезервированными ключевыми словами (такими, как class, int и т.д.)
- Класс может содержать различные переменные и методы.
- Выполнение любой программы начинается с метода main, который обязательно должен присутствовать в коде.

# Структура программы на Java




- Метод main также имеет модификатор public.
- Слово static указывает, что метод main – статический, а слово void – что он не возвращает никакого значения.
- Все это станет более понятно чуть позже ☺
- Далее в скобках у нас идут параметры метода – String[] args – это массив args, который хранит значения типа String (строкового). В данном случае они нам пока не нужны, но в реальной программе – это те строковые параметры, которые передаются при запуске программы из командной строки.
- В фигурных скобках располагается тело метода, содержащее инструкции, которые будут выполнятся при запуске программы.

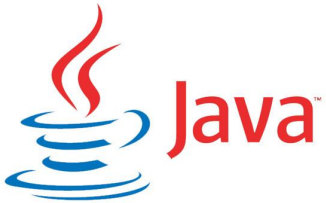
# Структура программы на Java




- В начале тела класса объявляем переменную **name**, которая будет хранить строку (т.е. тип **String**). Java является регистрозависимым языком, поэтому следующие два объявления переменных будут не эквивалентны: **String name** и **String Name**. В данном случае будут объявлены 2 разные переменные.

- Далее идет создания переменной консоли, которая даст возможность взаимодействовать с консолью: **Console con = System.console();** . Так как класс **Console** находится в библиотеке классов в пакете **java.io**, то чтобы его использовать, необходимо этот пакет подключить, используя директиву import: **import java.io.Console**.

# Структура программы на Java

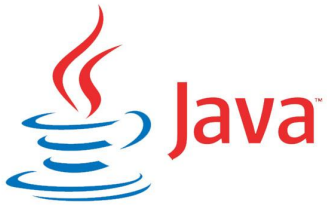



- Все это станет более понятно чуть позже ☺

- Далее в скобках у нас идут параметры метода – **String[] args** – это массив args, который хранит значения типа String (строкового). В данном случае они нам пока не нужны, но в реальной программе – это те строковые параметры, которые передаются при запуске программы из **командной строки**.

- В фигурных скобках располагается тело метода, содержащее инструкции, которые будут выполнятся при запуске программы.

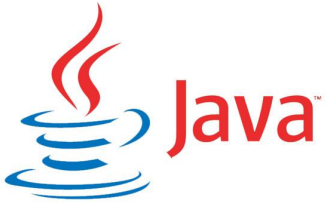
# Создание первой программы




Чтобы создать простейшую программу на Java вы должны первым делом создать текстовый файл с именем **Hello.java**, и следующим содержанием:

```
1   import java.io.Console;
2
3   public class Program{
4       static String name = "";
5
6       public static void main(String[] args){
7           String name;
8           Console con = System.console();
9           name = con.readLine("Please, enter your name: ");
10          System.out.println("Welcome to Java, " + name);
11      }
12  }
```

# Создание первой программы

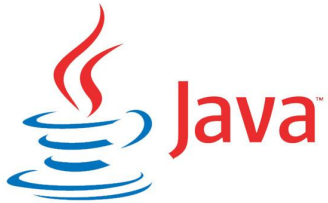



Далее необходимо в консоли набрать команду вида:

```
javac Hello.java
```

Для успешного выполнения этой команды вы должны находиться в том же каталоге, что и ваш файл.

# Создание первой программы



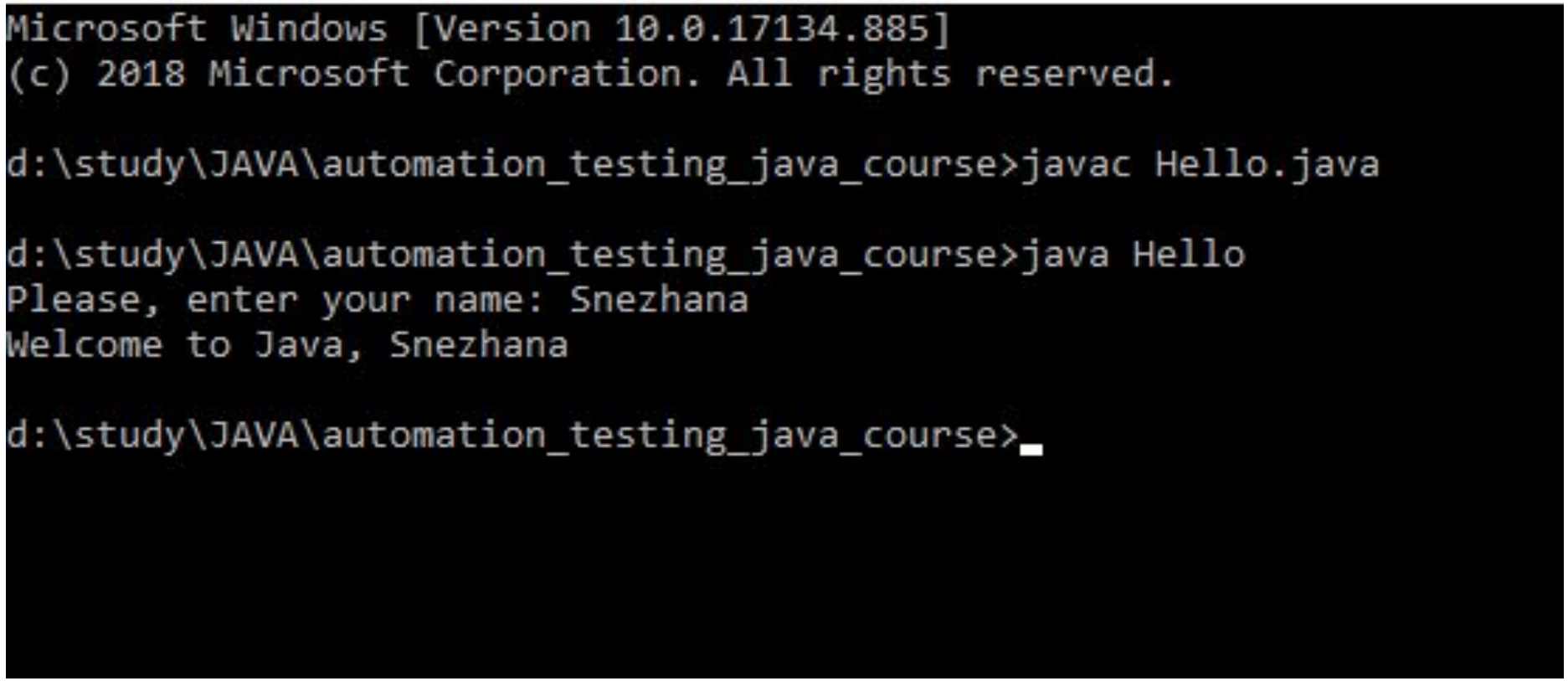

- В результате выполнения данной команды рядом с вашим файлом должен появиться другой файл с таким же именем и с расширением **.class**. Если он был успешно создан, программу можно запускать.

- Делается это командой:

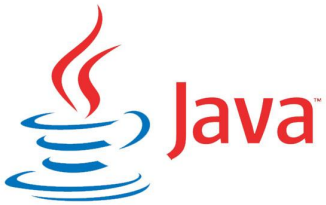
- *java Hello*

# Создание первой программы

- Если вы нигде не допустили ошибок, в консоли вы должны увидеть следующий текст:



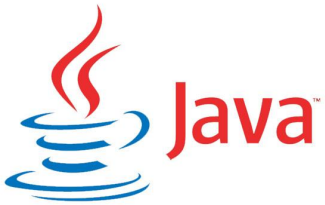
# Среды разработки на Java




**Интегри́рованная среда́ разрабо́тки (англ. Integrated development environment — IDE)** — комплекс программных средств, используемый программистами для разработки программного обеспечения, включающий в себя:

- редактор кода,
- компилятор и/или интерпретатор,
- средства автоматизации сборки,
- отладчик.

# Наиболее популярные среды разработки на Java



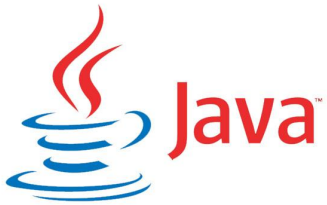

- **IntelliJ IDEA**

Одна из самых функциональных сред для java разработки, созданная компанией JetBrains.

Оснащена системой интеллектуальной помощи в написании кода, включает в себя огромное количество плагинов и надстроек под любую задачу, функцию автодополнения, имеет современный интерфейс.

Существуют две версии IntelliJ — **Community Edition**, которая является бесплатной, и **Ultimate Edition**, которая полностью признана и требует использования оплачиваемых лицензий.

# Наиболее популярные среды разработки на Java




- **Eclipse**

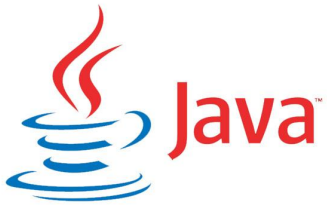
Одна из самых популярных IDE, не только для Java, но и для C ++ с PHP, созданная компанией Eclipse Foundation.

Это инструмент с открытым исходным кодом, имеющий отличное сообщество разработчиков.

В нем имеется огромная библиотека плагинов, созданная самими пользователями, а также - множество версий, самая популярная из которых — Eclipse Oxygen.

Является бесплатной и не требует лицензии.

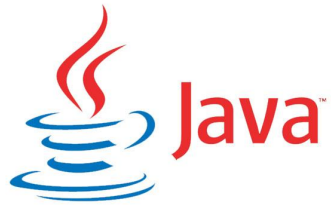
# Наиболее популярные среды разработки на Java




Среди прочих можно выделить также:

- NetBeans
- JDeveloper
- Dr. Java
- BlueJ
- jCreator
- jGrasp
- Greenfoot
- Codenvy (облачная IDE)
- и др.

# Что будем использовать?

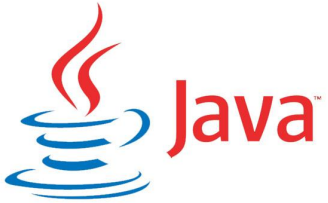



Для наших целей будем использовать IntelliJ IDEA  Community версию.

Скачать IntelliJ IDEA  Community можно с официального сайта пройдя по ссылке
https://www.jetbrains.com/idea/download/#section=windows

Далее необходимо запустить установочный файл и выполнить все шаги в открывшемся окне установки.

# Создаем первый проект в IntelliJ IDEA  Community




Для наших целей будем использовать IntelliJ IDEA  Community версию.

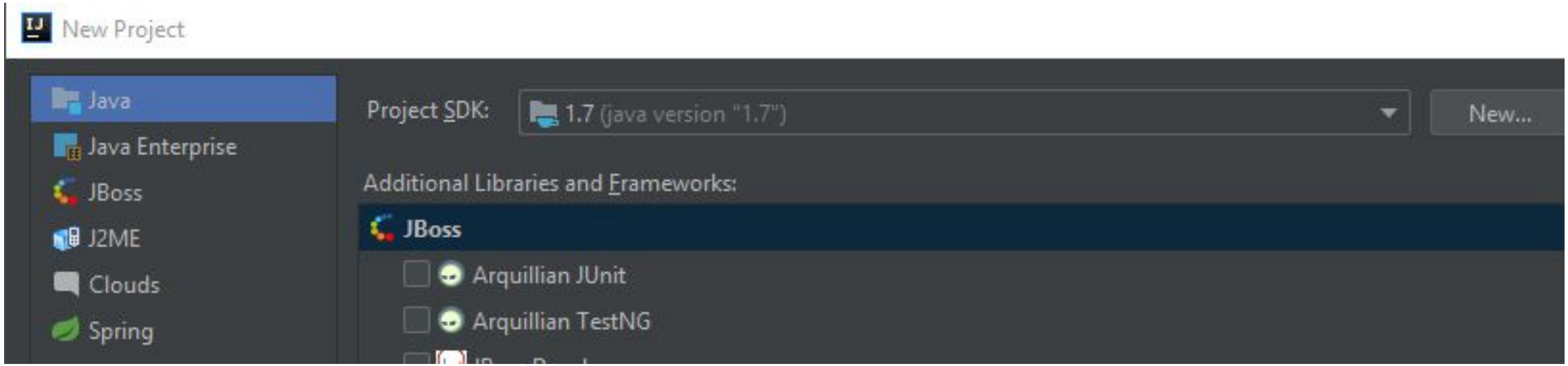
Скачать IntelliJ IDEA  Community можно с официального сайта пройдя по ссылке
https://www.jetbrains.com/idea/download/#section=windows

Далее необходимо запустить установочный файл и выполнить все шаги в открывшемся окне установки.

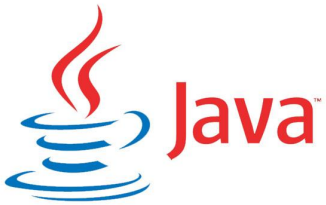
# Создаем первый проект в IntelliJ IDEA Community




В верхнем левом углу жмем File / New / Project и в открывшемся окне New Project выбираем Java.



# Создаем первый проект в IntelliJ IDEA Community



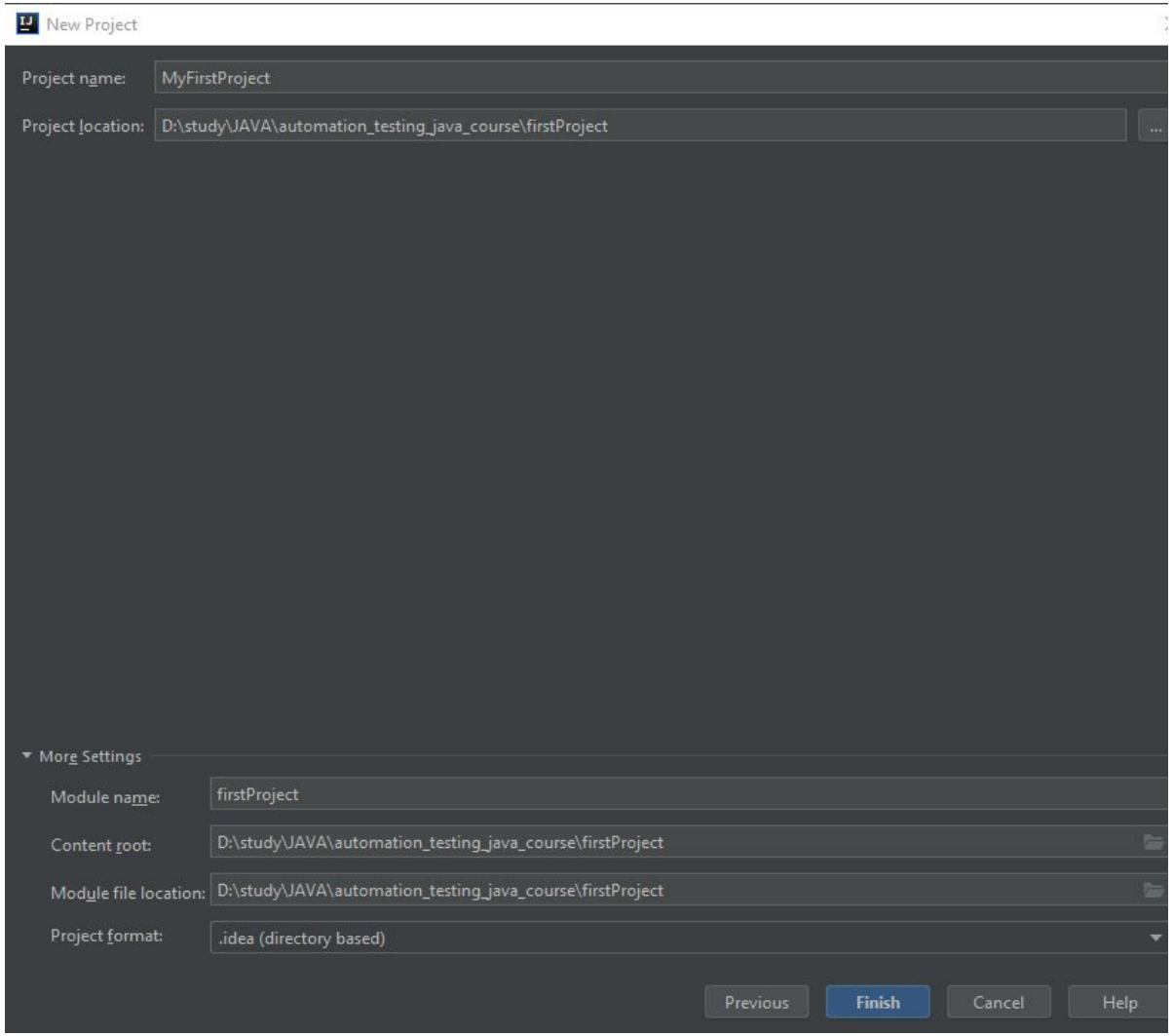

Жмем Next, оставляем все как есть в следующем окне и снова жмем Next.

На диске D: создаем папку и называем ее firstProject.

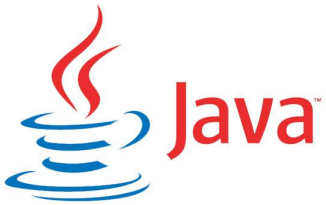
В открывшемся окне указываем имя проекта в поле Project Name - MyFirstProject

В поле Project Location указываем созданную Вами директорию, куда будем сохранять проект - firstProject.

# Создаем первый проект в IntelliJ IDEA  Community



# Создаем первый проект в IntelliJ IDEA Community




Нажимаем Finish, после чего созданный проект должен открыться в IDE.