# *Лекция 1. Язык программирования Java*

**Java** – [объектно-ориентированный язык программирования](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F" \o "Объектно-ориентированный язык программирования), разработанный компанией [Sun Microsystems](https://ru.wikipedia.org/wiki/Sun_Microsystems" \o "Sun Microsystems), и впоследствии приобретённый компанией [Oracle](https://ru.wikipedia.org/wiki/Oracle). Первая версия языка была представлена компанией [Sun Microsystems](https://ru.wikipedia.org/wiki/Sun_Microsystems) в 1994 году (Java 1.0). Актуальная версия на сегодняшний день – Java 8 (2014 г.).

# *Свойства языка программирования Java*

Язык Java обладает следующими характеристиками:

1. *Объектно-ориентированный*.

Язык Java разработан согласно *объектно*-*ориентированной* *парадигме* *программирования* (ООП), в центре внимания которого находятся данные  (объекты) и работа с ними (методы).

Объектно-ориентированное программирование предполагает использование трех концепций – *инкапсуляции*, *наследования* и *полиморфизма*. Соблюдение этих концепций позволяет разрабатывать ясную, гибкую, замкнутую архитектуру приложений.

1. *Интерпретируемый*.

Приложения Java [транслируются](https://ru.wikipedia.org/wiki/%D0%A2%D1%80%D0%B0%D0%BD%D1%81%D0%BB%D1%8F%D1%82%D0%BE%D1%80) в [байт-код](https://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B9%D1%82-%D0%BA%D0%BE%D0%B4), исполняемый [виртуальной машиной Java](https://ru.wikipedia.org/wiki/Java_Virtual_Machine" \o "Java Virtual Machine) (Java Virtual Machine – JVM), JVM интерпретирует байт-код в машинный код. Именно интерпретируемость байт-кода позволяет программам исполняться в любой среде, в которой установлен интерпретатор Java – JVM.

1. *Высокопроизводительный*.

Хотя интерпретируемый байт-код имеет достаточную производительность, бывают ситуации, в которых требуется более высокая эффективность. Именно поэтому в языке Java реализован ряд механизмов оптимизации процессов разработки и исполнения программного кода. Так, например, для оптимизации программы на этапе разработки в библиотеках языка предусмотрены возможности оптимизации чтения данных, ввода-вывода и пр.; на этапе исполнения кода также возможно использование механизмов оптимизации ряда операций (конкатенация).

1. *Многопоточный*.

Java предоставляет средства распараллеливания процессов в приложении.

1. *Надежный*.

Основное внимание в языке Java уделяется обнаружению ошибок, еще на этапе компиляции программы, а не вовремя исполнения программы (runtime).

1. *Распределенный*.

Язык Java обладает большой библиотекой программ для передачи данных по сети на основе таких протоколов TCP/IP (протокол управления передачей/интернет-протокол), как HTTP (протокол передачи гипертекста) или FTP (протокол передачи файлов).

1. *Динамичный*.

Java был разработан специально для подстройки под изменяющееся окружение. В то время как компилятор Java на этапе компиляции и статических проверок не допускает никаких отклонений, процесс сборки и выполнения сугубо динамический. Классы связываются только тогда, когда в этом есть необходимость. Новые программные модули могут подключаться из любых источников, в том числе, поставляться по сети.

# *Комплект средств разработки Java*

*Software Development Kit* (SDK) – комплект средств разработки для разработки приложений, состоящий из программного обеспечения базовых средств разработки, аппаратной платформы, компьютерной системы, консолей, операционных систем и прочих платформ.

SDK от [Oracle](https://ru.wikipedia.org/wiki/Oracle" \o "Oracle) для разработки Java-приложений называется *[Java Development Kit](https://ru.wikipedia.org/wiki/Java_Development_Kit" \o "Java Development Kit)* (JDK). JDK включает в себя [компилятор](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D0%B8%D0%BB%D1%8F%D1%82%D0%BE%D1%80" \o "Компилятор) (*javac*), стандартные библиотеки классов, примеры, документацию, различные утилиты и исполнительную систему Java ([JRE](https://ru.wikipedia.org/wiki/JRE" \o "JRE)). В состав JDK не входит [интегрированная среда разработки](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D1%80%D0%B5%D0%B4%D0%B0_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8).

*Java Runtime Environment* (JRE) – среда выполнения Java-программ, состоящая из минимальной реализации [виртуальной машины](https://ru.wikipedia.org/wiki/%D0%92%D0%B8%D1%80%D1%82%D1%83%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D0%BC%D0%B0%D1%88%D0%B8%D0%BD%D0%B0) без компилятора и других средств разработки и [библиотеки Java-классов](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B0%D0%BD%D0%B4%D0%B0%D1%80%D1%82%D0%BD%D0%B0%D1%8F_%D0%B1%D0%B8%D0%B1%D0%BB%D0%B8%D0%BE%D1%82%D0%B5%D0%BA%D0%B0).

*Java Virtual Machine* (JVM) — [виртуальная машина](https://ru.wikipedia.org/wiki/%D0%92%D0%B8%D1%80%D1%82%D1%83%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D0%BC%D0%B0%D1%88%D0%B8%D0%BD%D0%B0) Java, основная часть [JRE](https://ru.wikipedia.org/wiki/JRE). Виртуальная машина Java исполняет *[байт-код](https://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B9%D1%82-%D0%BA%D0%BE%D0%B4_Java" \o "Байт-код Java)*, полученный из исходного кода программы после компиляции. JVM может также использоваться для выполнения программ, написанных на других языках программирования. Например, исходный код на языке [Ada](https://ru.wikipedia.org/wiki/%D0%90%D0%B4%D0%B0_(%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)" \o "Ада (язык программирования)) может быть откомпилирован в байт-код Java, который затем может выполниться с помощью JVM.

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](https://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B9%D1%82-%D0%BA%D0%BE%D0%B4" \o "Байт-код) для многих платформ позволяет описать Java как «скомпилировано однажды, запускается везде» (compile once, run anywhere).

Виртуальные машины Java обычно содержат [интерпретатор](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BF%D1%80%D0%B5%D1%82%D0%B0%D1%82%D0%BE%D1%80" \o "Интерпретатор) байт-кода, однако, для повышения производительности во многих машинах также применяется [JIT](https://ru.wikipedia.org/wiki/JIT" \o "JIT)-компиляция часто исполняемых фрагментов байт-кода в машинный код.

# *Классификация платформ Java*

В Java выделены несколько платформ, согласно задачам и областям применения, в которых используется для разработки программных продуктов:

* *Java Standard Edition* ([Java SE](https://ru.wikipedia.org/wiki/Java_SE)) – основное издание Java, содержит компиляторы, API, [Java Runtime Environment](https://ru.wikipedia.org/wiki/Java_Runtime_Environment" \o "Java Runtime Environment); подходит для создания пользовательских приложений, в первую очередь – для настольных систем.
* *Java Enterprise Edition* ([Java EE](https://ru.wikipedia.org/wiki/Java_EE)), представляет собой набор спецификаций для создания программного обеспечения уровня предприятия.
* *Java Micro Edition* ([Java ME](https://ru.wikipedia.org/wiki/Java_ME)), создана для использования в устройствах, ограниченных по вычислительной мощности, например, в [мобильных телефонах](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B1%D0%B8%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D1%82%D0%B5%D0%BB%D0%B5%D1%84%D0%BE%D0%BD), [КПК](https://ru.wikipedia.org/wiki/%D0%9A%D0%B0%D1%80%D0%BC%D0%B0%D0%BD%D0%BD%D1%8B%D0%B9_%D0%BF%D0%B5%D1%80%D1%81%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80), встроенных системах;
* [*JavaFX*](https://ru.wikipedia.org/wiki/JavaFX) — технология, являющаяся следующим шагом в эволюции Java как Rich Client Platform; предназначена для создания графических интерфейсов корпоративных приложений и бизнеса.
* [*Java Card*](https://ru.wikipedia.org/wiki/Java_Card) — технология предоставляет безопасную среду для приложений, работающих на смарт-картах и других устройствах с очень ограниченным объёмом памяти и возможностями обработки.

# *Что пишут-то?*

Java написана на Java =)

Среды разработки Java-приложений написаны на Java =)

Графический интерфейс Maple(математический пакет вычислений)

Серверная часть Google+

[OpenOffice](http://www.openoffice.org/ru/)

Poseidon for UML

# *Практическое занятие 1.*

# *Установка JDK*

**Шаг 1. Установка JDK.**

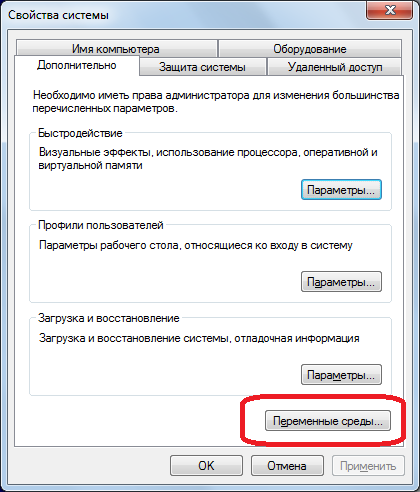
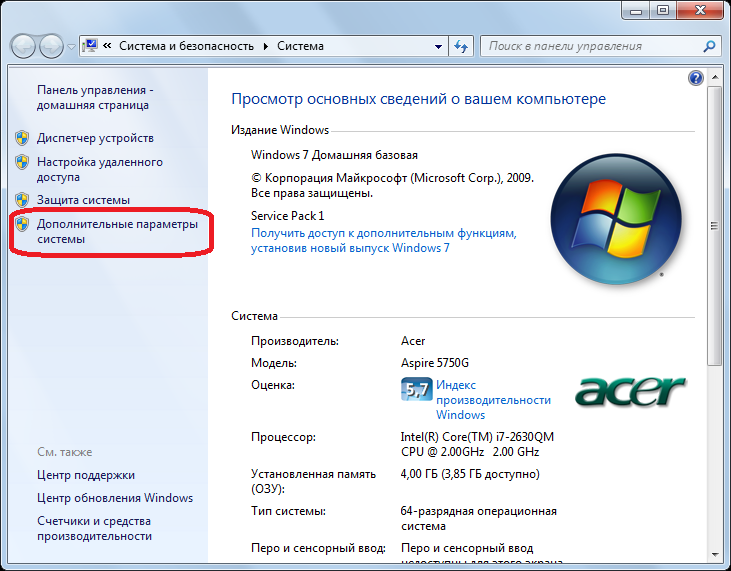
Скачать установочный файл с официального сайта Oracle и запустить инсталлятор и указать путь установки JDK.



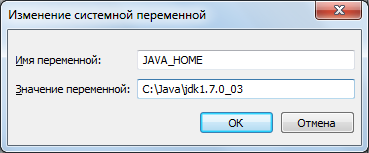
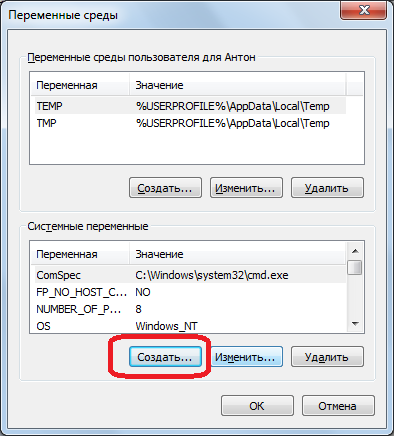
**Шаг 2. Настройка переменных среды операционной системы.**

Настройка переменных среды (Path, JAVA\_HOME) позволит Java-приложениям быть исполненными JRE.

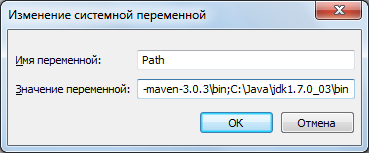
Меню «Пуск» → правой кнопкой мыши по «Компьютер» → «Свойства» → «Дополнительные параметры системы».



В окне «Переменные среды» создать новую переменную «JAVA\_HOME». В окне «Изменение системной переменной» в поле «Имя переменной» ввести «JAVA\_HOME», а в поле «Значение переменной» ввести путь, по которому был установлен JDK. Нажать «ОК».



В списке «Системные переменные» найти переменную «Path» и нажать кнопку «Изменить». Значения переменной задаются через «;». Необходимо снова добавить путь к JDK + «\bin» после чего снова «;» и путь к JDK БЕЗ папки bin. Так как переменной JAVA\_HOME присвоено значение пути к JDK, можно использовать имя этой переменной: %JAVA\_HOME%\bin; %JAVA\_HOME%.



# *Среда разработки Java-приложений*

Integrated Development Environment (IDE) – интегрированная среда разработки приложений.

Первые IDE представляли собой текстовый редактор, который служил для набора и редактирования исходных кодов, остальную работу, начиная от компиляции и заканчивая окончательной сборкой проекта, приходилось выполнять вручную.

Современные IDE являются многофункциональными программы, включающие в себя и сам редактор для написания исходного кода, и ряд дополнительных возможностей, как то автоматическое создания Javadoc, рефакторинга кода, навигация по коду, профайлинг, UML-проектирование, клиент для подключения к любой СУБД и даже проверка орфографии.

# *«Hello world !!!» в Eclipse IDE*

В Eclipse существует понятие рабочего пространства (workspase) – это место, в котором определено множество проектов, объединенных по какому-либо принципу (размещение проектов в одной и той же папке на диске, логическая связность проектов, зависимость проектов друг от друга).

При работе с Eclipse прежде всего необходимо создать рабочее пространство, в котором будут храниться настройки рабочего пространства в целом, и проектов, в частности. При этом лучше разделить рабочее пространство и фактическое расположение проекта на диске, т.е. проект, находящийся в рабочем пространстве должен храниться отдельно от самого рабочего пространства.

1. **Создание рабочего пространства в Eclipse.**

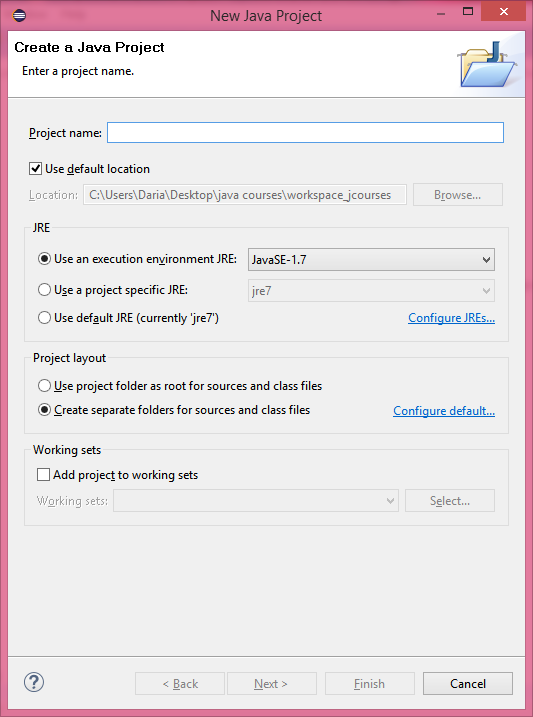
Перейдите в папку, в которой будет храниться рабочее пространство. Создайте папку Eclipse WS, в ней подпапку JS\_TA.

Откройте Eclipse и укажите загрузчику путь к папке JS\_TA. После этого действия перейдите в папку JS\_TA: в них появились подпапки настроек рабочего пространства.

1. **Создание Java – проекта.**

На уровне с папкой Eclipse WS создайте папку Java Projects.

В рабочем пространстве выберете File – New – Java Project.

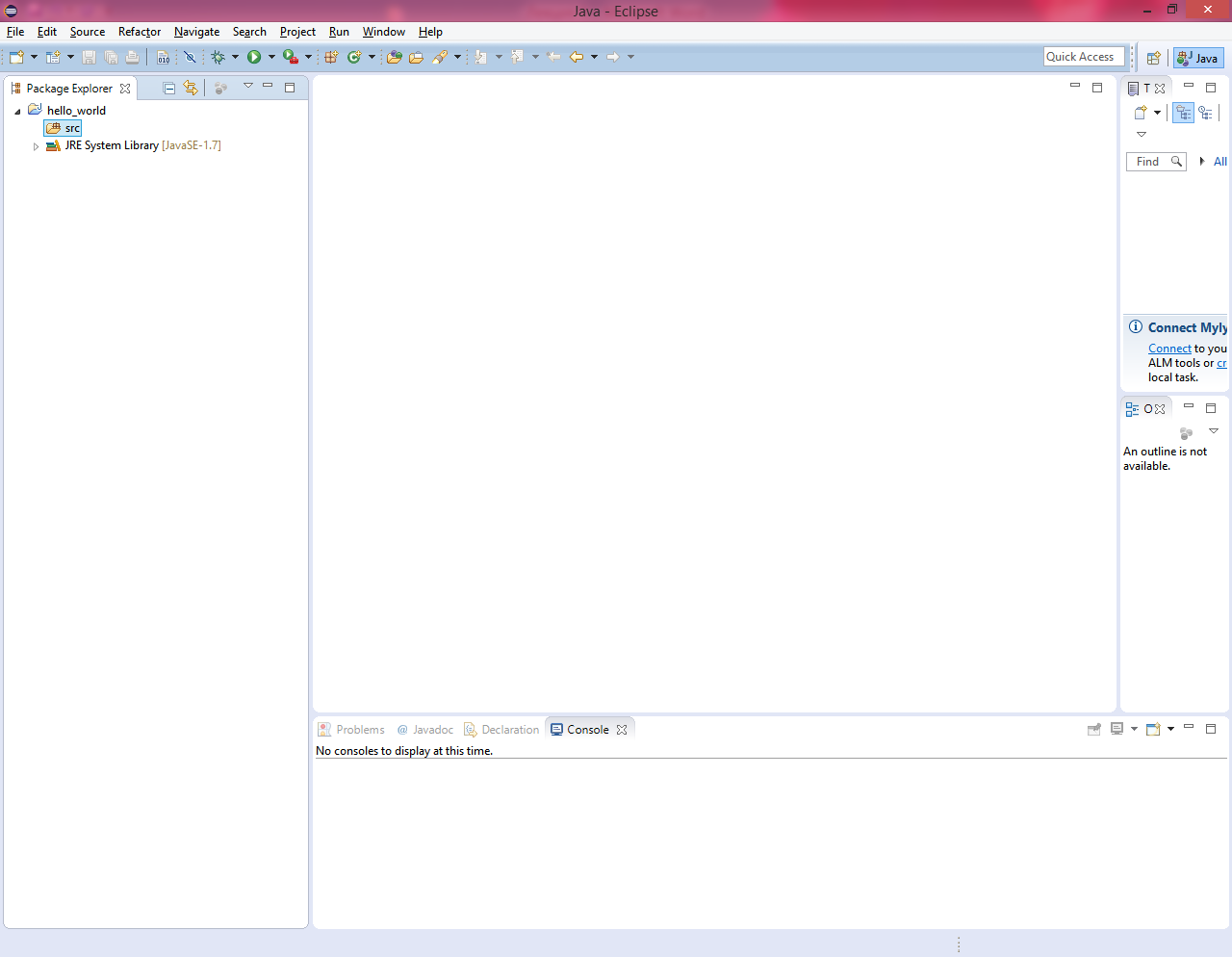


В открывшемся окне необходимо заполнить:

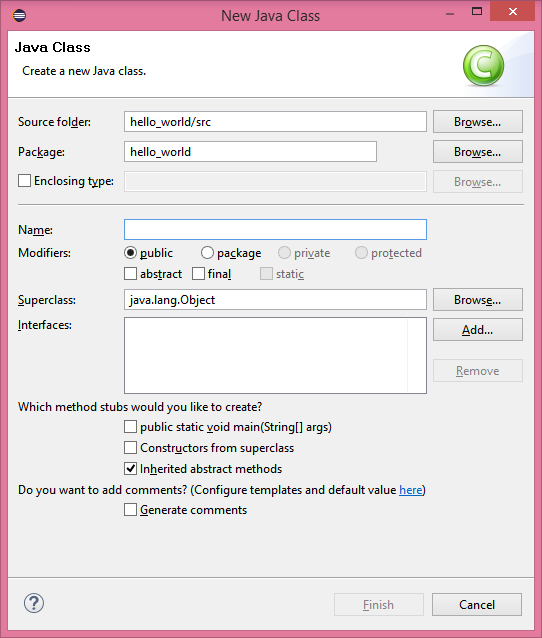
|  |  |  |
| --- | --- | --- |
| **Настройка проекта** | **Обозначение** | **Значение** |
| Project Name | Имя проекта | hello\_world |
| Location | Местоположение проекта | Папка Java-проектов – Java Projects |

1. **Создание класса в проекте.**

После создания Java-проекта, в проекте hello\_world появился каталог src – это папка, в которой будут храниться исходные коды приложения.

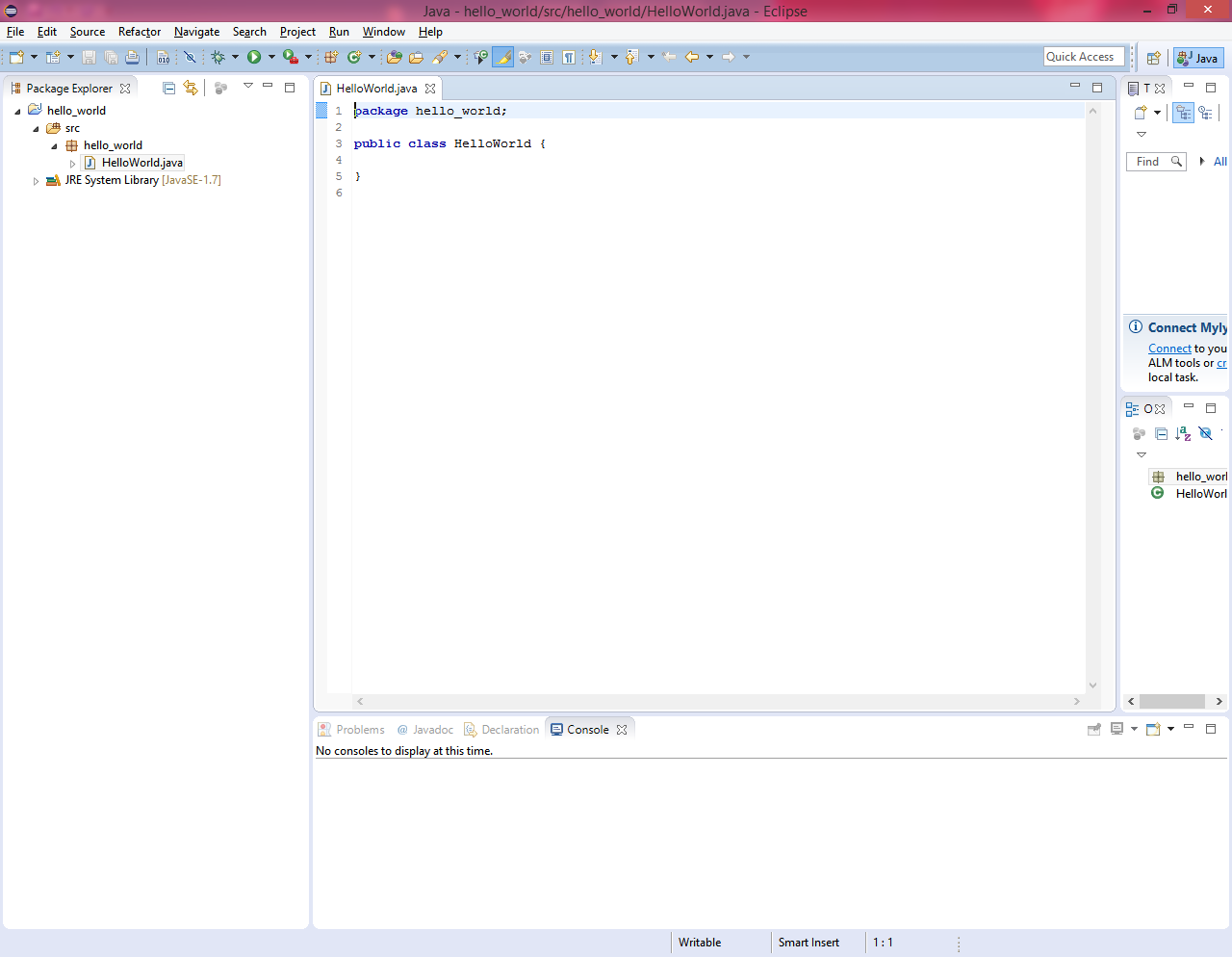


Правой кнопкой мыши по папке src – New – Class.



В данном окне важно положение создаваемого класса – он находится в папке src, в пакете hello\_world. (Пакет – это префикс имени создаваемого класса).

В поле Name задается имя создаваемого класса: HelloWorld.



Создался пустой класс HelloWorld в подпапке hello\_world в проекте hello\_world.

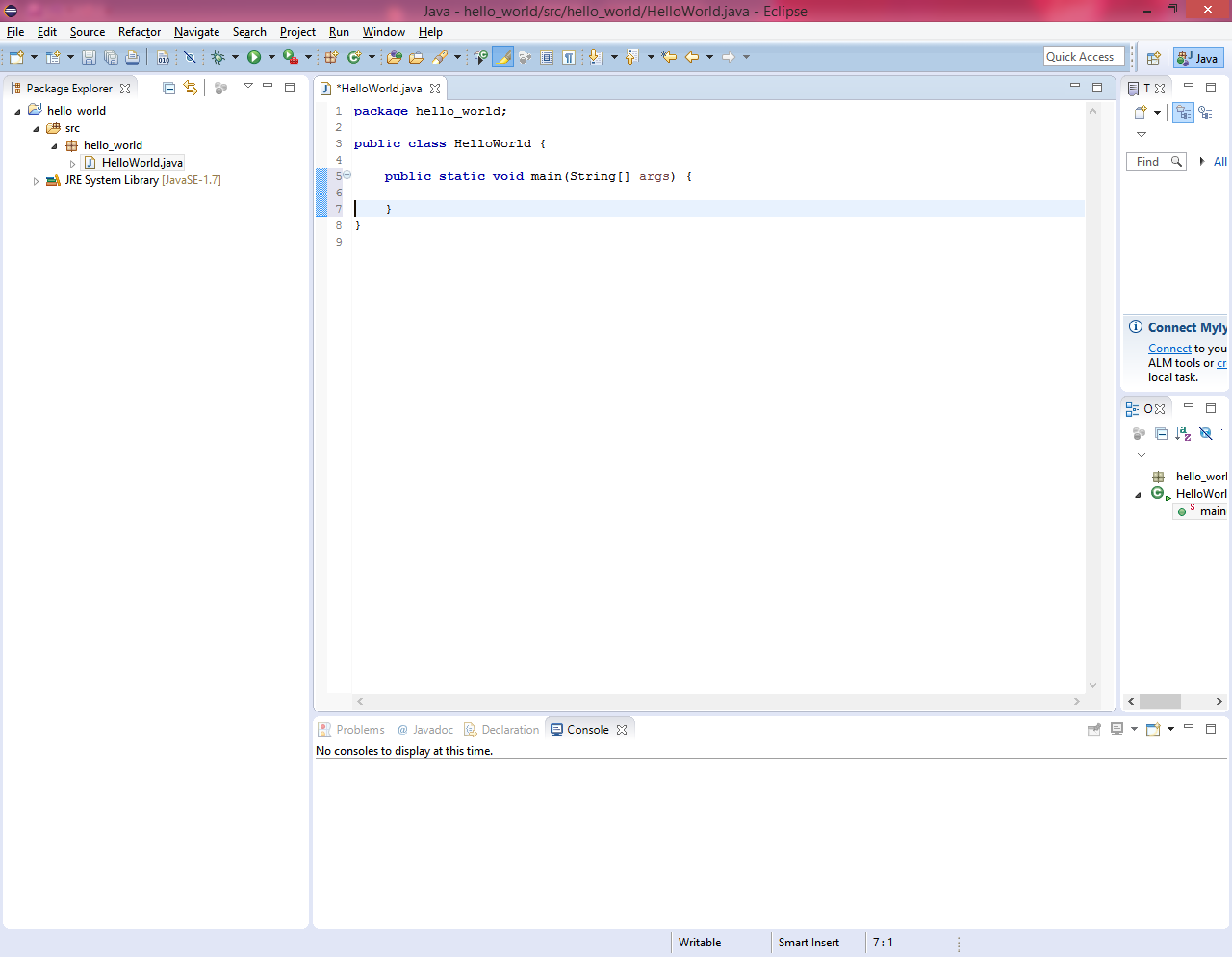
Зайдите в папку Java Projects – hello\_word –src – hello\_world – HelloWorld.java – это файл исходного кода java. Все исходные коды, хранящиеся в файлах, носят расширение «.java».

1. **Исходный код программы.**

Точкой входа для программы в Java служит метод (функция) main.

Метод – это набор инструкций в Java-программе, разделенный областью видимости. Область видимости метода – это открывающаяся в начале метода скобка, и закрывающаяся в конце, служащая также идентификатором окончания инструкций.

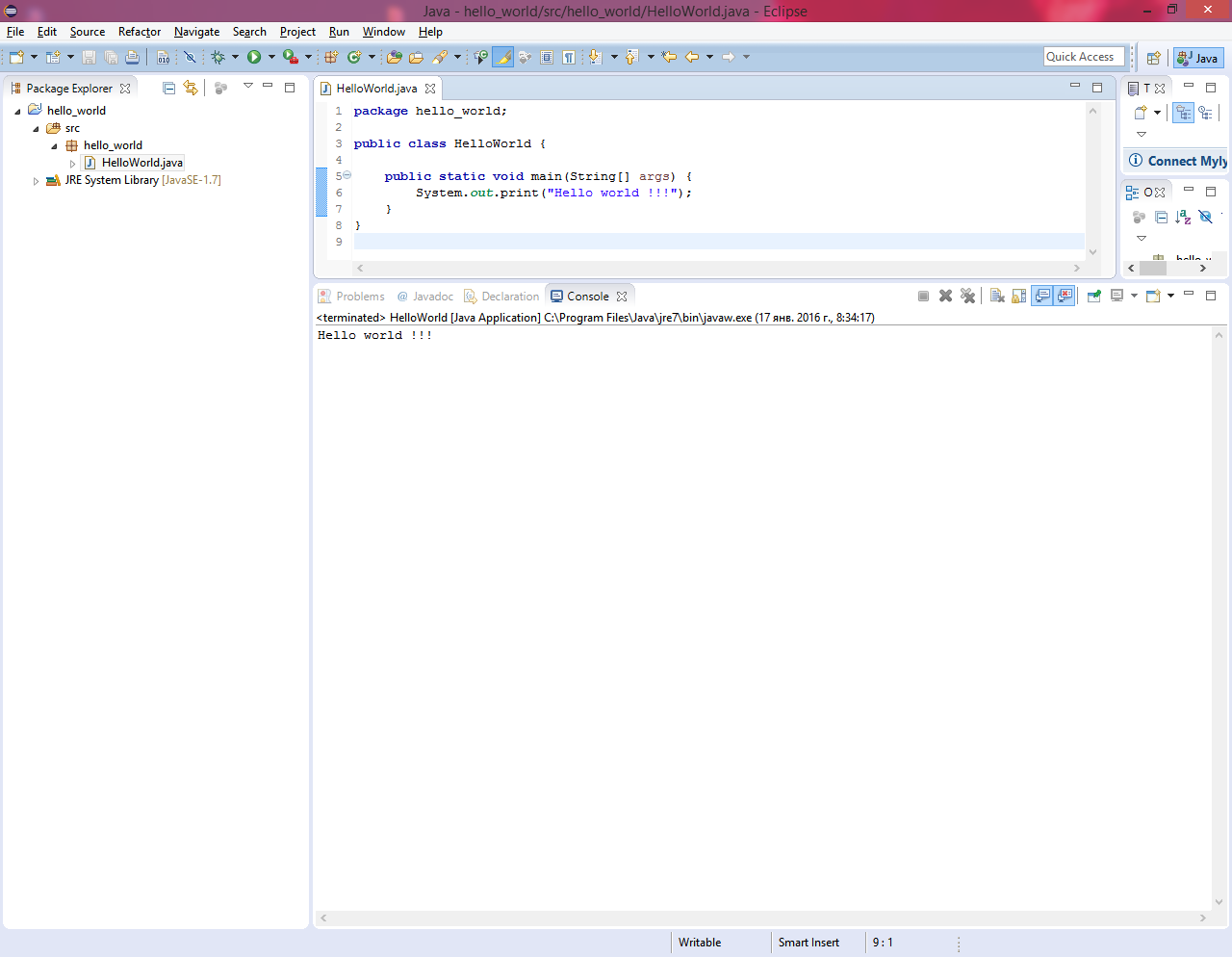
Метод main имеет декларированную структуру: чтобы Java-машина исполнила инструкции в методе main, он должен быть объявлен, согласно его декларации:



Имя метода – main, в него передается массив параметров типа String (строки) – args.

В теле метода (между открывающейся и закрывающейся скобкой) впишите инструкцию вывода строки «Hello world !!!» на экран: для этого используется специальная функция в Java – System.out.print(String toPrint), где toPrint – это строка, которую необходимо вывести в консоль.

Запускаем код на исполнение: правой кнопкой по исходному файлу – Run As – Java Application.



**Самостоятельно:**

1. Найти в папке приложения скомпилированный код. Какое расширение имеют скомпилированные файлы?
2. Запустить код на исполнение через командную строку Windows.

Для запуска файла необходимо использовать следующую команду:

java **–classpath** “ПУТЬ\_К\_ПАПКЕ\_ГДЕ\_ЛЕЖАТ\_СКОМПИЛИРОВАННЫЕ\_ФАЙЛЫ” ИМЯ\_ПАКЕТА.ИМЯ\_КЛАССА.

сlasspath – это переменная, в которой JVM ищет файлы исполнения.