

Лабораторная работа 1

Тема: Создание и запуск первого Java-приложения. Знакомство со средой разработки IntelliJ IDEA.

Цель: Изучение процесса создания программ на языке JAVA и знакомство со средой программирования IntelliJ IDEA.

Задачи: 1. Изучение интерфейса среды программирования IntelliJ IDEA, структуры проекта.

2. Создание простейших консольных приложений.

3. Изучение способов отладки программ.

Теоретическая часть

Java — объектно-ориентированный язык программирования, разработанный компанией Sun Microsystems (в последующем приобретённой компанией Oracle) в 1995 году.

Программы на Java транслируются в байт-код, выполняемый виртуальной машиной Java (JVM) — программой, обрабатывающей байтовый код и передающей инструкции оборудованию как интерпретатор. Достоинством подобного способа выполнения программ является полная независимость байт-кода от операционной системы и оборудования, что позволяет выполнять Java-приложения на любом устройстве, для которого существует соответствующая виртуальная машина. Другой важной особенностью технологии Java является гибкая система безопасности, в рамках которой исполнение программы полностью контролируется виртуальной машиной. Любые операции, которые превышают установленные полномочия программы (например, попытка несанкционированного доступа к данным или соединения с другим компьютером), вызывают немедленное прерывание.

Компания Sun Microsystems при выпуске языка заложила 5 парадигм:

- простота, объектная ориентированность;
- надёжность и безопасность;
- переносимость и независимость от платформы;
- высокая производительность;
- интрепретируемость, поточность и динамичность.

Основные возможности языка JAVA

- автоматическое управление памятью;
- расширенные возможности обработки исключительных ситуаций;
- богатый набор средств фильтрации ввода-вывода;
- набор стандартных коллекций: массив, список, стек и т. п.;
- наличие простых средств создания сетевых приложений (в том числе с использованием протокола RMI);
- встроенные в язык средства создания многопоточных приложений;
- унифицированный доступ к базам данных:
 - на уровне отдельных SQL-запросов — на основе JDBC, SQLJ;
 - на уровне концепции объектов, обладающих способностью к хранению в базе данных — на основе Java Data Objects (англ.) и Java Persistence API;
- поддержка обобщений;

- поддержка лямбд, замыканий, встроенные возможности функционального программирования;
- параллельное выполнение программ.

Пространство имён JAVA

Идея пространств имён воплощена в Java-пакетах.

Внутри пакета есть два независимых пространства имен: переменные и методы.

Java package (пакет Java) — механизм, позволяющий организовать Java классы в пространстве имен аналогично модулям.

Обычно в пакеты объединяют классы одной и той же категории, либо предоставляющие сходную функциональность.

- Каждый пакет предоставляет уникальное пространство имен для своего содержимого.
- Допустимы вложенные пакеты.

Основные идеи JAVA

Примитивные типы

В языке Java определены следующие примитивные (скалярные, простые) типы: *boolean, byte, char, short, int, long, float, double*. Существует также вспомогательный девятый примитивный тип — *void*, однако переменные и поля такого типа не могут быть объявлены в коде, а сам тип используется только для описания соответствующего ему класса, для использования при рефлексии (процесс, во время которого программа может отслеживать и модифицировать собственную структуру и поведение во время выполнения).

Длины и диапазоны значений примитивных типов определяются стандартом, а не реализацией, и приведены в таблице.

Тип	Длина (в байтах)	Диапазон или набор значений
boolean	1 в массивах, 4 в переменных	true, false
byte	1	-128..127
char	2	0.. $2^{16}-1$, или 0..65535
short	2	$-2^{15}..2^{15}-1$, или -32768..32767
int	4	$-2^{31}..2^{31}-1$, или -2147483648..2147483647
long	8	$-2^{63}..2^{63}-1$, или примерно $-9.2 \cdot 10^{18}..9.2 \cdot 10^{18}$
float	4	$-(2 \cdot 2^{-23}) \cdot 2^{127}..(2 \cdot 2^{-23}) \cdot 2^{127}$, или примерно $-3.4 \cdot 10^{38}..3.4 \cdot 10^{38}$, а также $-\infty, \infty, \text{NaN}$
double	8	$-(2 \cdot 2^{-52}) \cdot 2^{1023}..(2 \cdot 2^{-52}) \cdot 2^{1023}$, или примерно $-1.8 \cdot 10^{308}..1.8 \cdot 10^{308}$, а также $-\infty, \infty, \text{NaN}$

Инициализация переменных

Все переменные или требуют явного определения, или автоматически заполняются нулями (0, null, массивом нулей). Таким образом, исчезают ошибки, связанные со случайным использованием неинициализированной памяти, характерные для низкоуровневых языков вроде C.

IntelliJ IDEA

IntelliJ IDEA – это коммерческая среда разработки приложений. Существует бесплатная версия “Community Edition” с ограниченным функционалом и полная коммерческая версия “Ultimate Edition”. Версия “Community” распространяется на основе лицензии Apache и включает инструменты тестирования, средства контроля версий, сборки ПО, поддерживает языки Java, Java ME, Groovy, Scala и Clojure.

Также в ограниченной версии поддерживается разработка программ для системы Android, имеются средства разработки пользовательского интерфейса, редактор XML-кода, регулярных выражений, проверка синтаксиса, импорт и экспорт проектов Eclipse. “IntelliJ IDEA Community Edition” легко интегрируется с системами отслеживания ошибок.

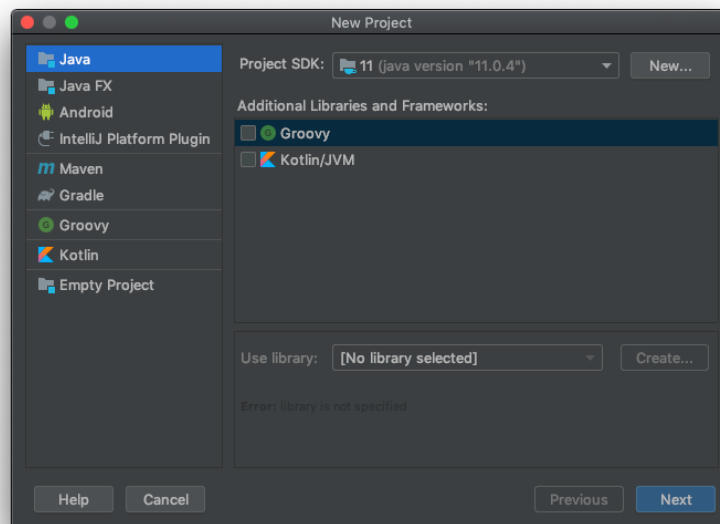
Версия “Ultimate” помимо стандартного набора языков программирования версии “Community”, поддерживает PHP, SQL, Ruby, CSS, Python, HTML, JS. Работа с технологией Java EE и фреймворками Hibernate, Rails, Google Web Toolkit, Spring также присутствуют. Среди средств интеграции Microsoft Team Foundation Server, Rational Clear Case и Perforce.

Создание проекта

Для начала создадим консольное приложение. Консольное приложение Java представляет собой откомпилированный класс, содержащий точку входа:

Метод `void main()` и модификаторами `public static`.

1. Если ни один проект в данный момент не открыт, нажмите кнопку **Create New Project** на экране приветствия. В противном случае, выберите пункт **New Project** в меню **File**. В результате откроется мастер создания проекта.
2. В левой панели выберите **Java**.

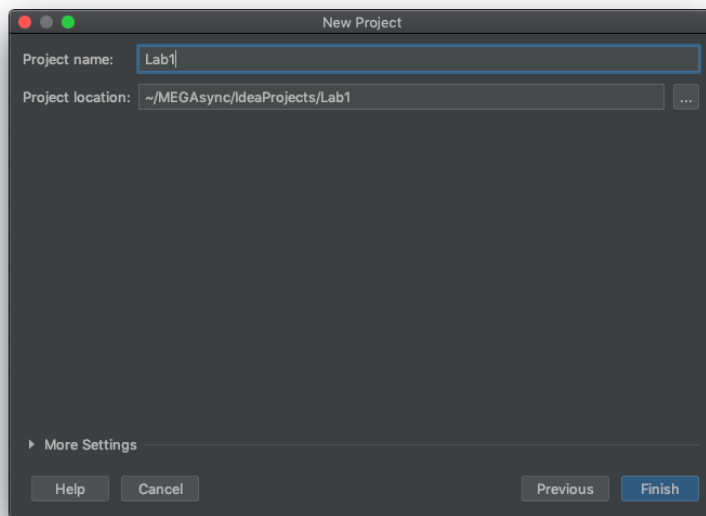


3. Если в поле Project SDK стоит **<None>**, то необходимо выбрать версию JDK. Вместо **<None>** нажмите **New** и в подменю выберите **JDK**. В окне **Select Home Directory for JDK** выберите директорию, в которую устанавливалось **JDK** и нажмите **OK**.

Версия JDK, которую вы выбрали, появится в поле **Project SDK**.

Кликните **Next**. Учтите, что указанная версия JDK будет связана по умолчанию со всеми проектами и Java модулями, которые в дальнейшем будут создаваться.

В правой части страницы, в поле **Project name**, введите название проекта: *Lab1*.



Изучение структуры проекта

В дереве проекта видим две директории верхнего уровня:

- **Lab1**. Это узел, содержащий Java модуль. Папки `.idea` и файлы внутри используются для хранения данных конфигурации вашего проекта и модулей соответственно. SRC папки содержат исходный код.
- **External Libraries** (внешние библиотеки). Это категория, которая представляет все «внешние» ресурсы, необходимые для вашего проекта. В настоящее время в этой категории файлы `.jar`, из выбранного нами JDK.

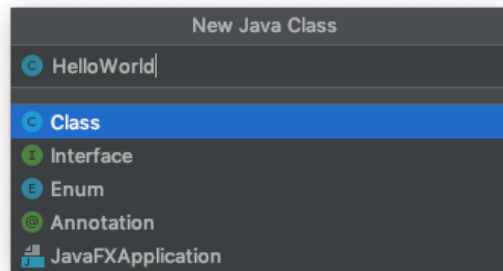
Создание пакета

Теперь мы собираемся создать пакет для класса `Lab1`. Назовем этот пакет `com.example.lab1`.

1. В окне инструментов **Project** выберите папку SRC и нажмите ALT + INSERT. (В качестве альтернативы, вы можете выбрать **File -> New**, или **New** из контекстного меню для папки SRC).
2. В меню **New** выберите **Package**.
3. В открывшемся окне **New Package** введите имя пакета (`com.example.lab1`). Нажмите кнопку OK (или клавишу ENTER).

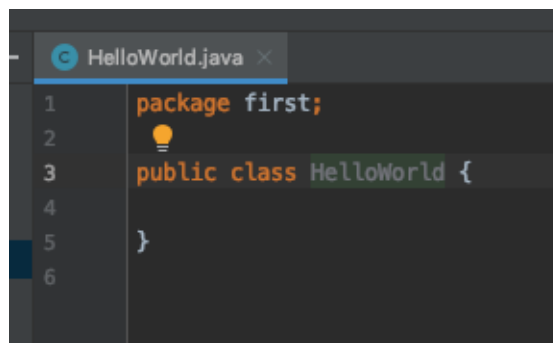
Создание класса

1. Нажмите ALT + INSERT. В окне **New** из списка доступных действий с выделенным пакетом `com.example.lab1` выбираем **Java Class** и нажимаем Enter.
2. В появившемся окне **Create New Class** в поле **Name** вводим имя `HelloWorld`. В поле **Kind** оставляем тип **Class** и нажимаем Enter, чтобы подтвердить создание класса.



Созданный класс HelloWorld появляется в структуре проекта:

После создания класса соответствующий ему файл HelloWorld.java открывается в редакторе.



Обратите внимание на оператор пакета в начале файла, а также объявление класса. При создании класса, IntelliJ IDEA использует файл шаблона для класса Java.

Также обратите внимание на желтую лампочку. Эта лампа указывает, что в IntelliJ IDEA есть предложения для текущего контекста. Нажмите на лампочку или ALT + ENTER, чтобы увидеть список доступных действий.

Написание кода для класса HelloWorld

Код в конечном состоянии будет выглядеть следующим образом:

```
package com.example.lab1;
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}
```

Здесь класс HelloWorld используется только для того, чтобы определить метод main(), который и является точкой входа и с которого начинается выполнения программы интерпретатором Java. Метод main() содержит аргументы-параметры командной строки String[] args в виде массива строк и является открытым (public) членом класса. Это означает, что метод main() виден и доступен любому классу. Ключевое слово static объявляет методы и переменные класса, используемые для работы с классом в целом, а не только с объектом класса. **Символы верхнего и нижнего регистров в Java различаются.**

Кроме того, этот метод можно использовать для печати значений переменных - как по отдельности, так и со строками символов, например:

```
System.out.println("Symbol array");  
int i=7; System.out.println(i);           //Вывод 7  
int j=10; System.out.println("j="+j+(i+1)); // Вывод j=108
```

Все классы являются производными (или подклассами) от существующих классов. В случае класса HelloWorld не было явно указано, от какого он класса он произошел. В таком случае -если не определен суперкласс, то по умолчанию предполагается, что таким суперклассом является Object.

Так как класс HelloWorld объявлен как public, то имя файла, в котором содержится его исходный код, должно совпадать с именем класса. Для классов, не объявленных как public, имена содержащих их исходные тексты файлов могут быть любыми (расширение обязательно .java).

В классе Hello объявляется метод main() со строковым параметром args, который будет содержать аргументы командной строки, передаваемые при запуске приложения:

```
public static void main(String args[]) {
```

Без функции main() ее интерпретатор не сумеет понять, откуда начинать выполнение приложение (метод main() является точкой входа приложения). Java-приложения могут запускаться с аргументами командной строки. Хотя необходимо обязательно включать параметр args в определение метода main(), но использовать аргументы командной строки необязательно.


Объявление пакета и класса уже есть, теперь добавим недостающие пару строк.

Теперь можно добавить оставшиеся строки кода

```
System.out.println («Hello, World!»);.
```

Построение проекта

Опции построения проекта или его части доступны в меню **Build**.

Многие из этих опций доступны также в контекстном меню в окне **Project** и в редакторе для HelloWorld.java. Также есть значок на панели инструментов, которая соответствует команде **Make Project** (). Теперь давайте построим проект.

Строительство в данном конкретном случае- просто компиляция исходного файла Java в файл класса. Таким образом, любой из вариантов в меню **Build (Make Project, Make Module 'HelloWorld', или Compile 'HelloWorld.java')** могут быть использованы для этой цели.

Клавиатурный эквивалент для команды построения проекта - CTRL + F9. Когда этот процесс компиляции будет завершен, соответствующая информация отображается в строке состояния.

Теперь, если вы перейдете в папку модуля вывода (по умолчанию это папка \out\production\, в нашем случае, и папка и называются HelloWorld), вы увидите там структуру папок для пакета com.example.helloworld и HelloWorld.class файл в папке HelloWorld.

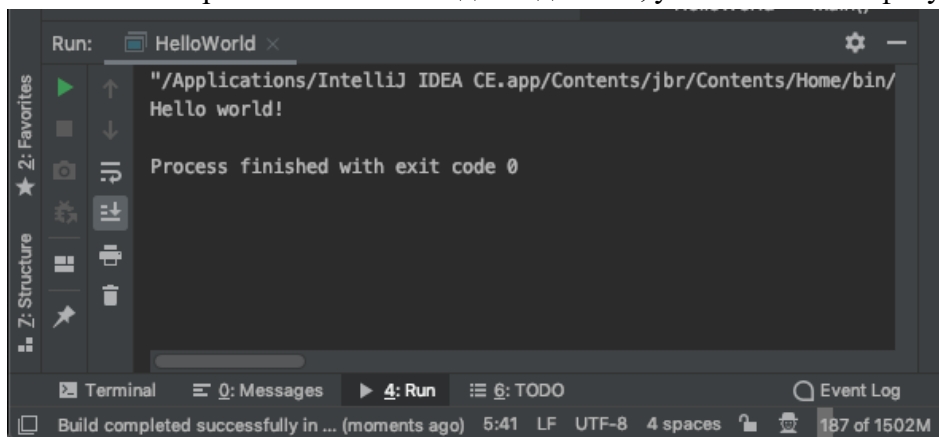
Запуск приложения

Приложение IntelliJ IDEA выполняется согласно тому, что называется конфигурацией запуска/отладки (Run/Debug). Такая конфигурация, как правило, должна быть создана до запуска приложения.

В случае класса HelloWorld, нет необходимости создавать конфигурацию запуска и отладки заранее. Класс содержит метод **main()**. Такие классы могут быть запущены сразу, прямо из редактора. Для этой цели существует команда **Run** '**<ClassName>.main()**' в контекстном меню для класса.

Таким образом, чтобы запустить класс, щелкните правой кнопкой мыши где-нибудь в области редактирования и выберите **Run 'HelloWorld.main ()'**.

В результате выполнения команды **Run** появляется окно в нижней части экрана. Оно отвечает за отображение всех выходных данных, указанных в конфигурации команды.



Первая строка в окне содержит командную строку IntelliJ IDEA, используемую для запуска класса, включая все опции и аргументы. Последняя строка показывает, что процесс завершился нормально, бесконечных циклов не произошло.

Работа с консолью (Ввод и вывод данных)

Самая важная особенность программ — это умение считывать некоторые данные, с которыми она в дальнейшем выполняет какие-либо операции. Для того, чтобы считывать данные из консоли применяют класс `Scanner`. Для того, чтобы применять данный класс необходимо подключить `import java.util.Scanner;`

Конструктор класса `Scanner` — `public Scanner(InputStream source)`

Создает новый сканер, который создает значения, отсканированные из указанного входного потока. Байты из потока преобразуются в символы с использованием кодировки по умолчанию базовой платформы.

Параметры: источник — входной поток для сканирования.

Класс `java.lang.System` является `final`, все поля и методы являются статическими (`static`), поэтому мы не можем создать подкласс и переопределить его методы используя наследование. Класс `Java System` не предоставляет каких-либо публичных конструкторов, поэтому мы не можем создать экземпляр этого класса.

Класс `System` содержит три поля — `in`, `out` и `err`. Они используются для чтения данных из `InputStream` и записи данных в `OutputStream`.

Методы класса `Scanner`:

`next()`: считывает введенную строку до первого пробела

`nextLine()`: считывает всю введенную строку

`nextInt()`: считывает введенное число `int`

`nextDouble()`: считывает введенное число `double`

`nextBoolean()`: считывает значение `boolean`

`nextByte()`: считывает введенное число `byte`

`nextFloat()`: считывает введенное число `float`

`nextShort()`: считывает введенное число `short`

Для проверки на наличие корректных данных имеется следующая группа методов: `hasNextInt()` — проверяет, является ли следующая порция введенных данных числом, или нет (возвращает, соответственно, `true` или `false`).

`hasNextLine()` — проверяет, является ли следующая порция данных строкой.

`hasNextByte()`

`hasNextShort()`

`hasNextLong()`

`hasNextFloat()`

`hasNextDouble()`

Пример:

```
public class Main {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        System.out.println("Введите число:");
        if (in.hasNextInt()) {
            int number = in.nextInt();
        }
    }
}
```



```

        System.out.printf("Введено число %d", number);
    } else {
        System.out.println("Введено некорректное число");
    }
}
}

```

Задание на лабораторную работу

Создать программу, реализующий вычисление значения переменных по заданным расчетным формулам и наборам введенных данных. На экран вывести значения вводимых исходных данных и результаты вычислений, сопровождая ввод и вывод поясняющими комментариями.

Вариант задания	Расчетные формулы
1	$a = \frac{\sin(x)+4}{\cos(y^2)}; b = \sqrt{a} * \frac{x}{y}$
2	$a = \frac{\cos(x)+\sin(y^2)}{\cos(y^2)}; b = \sqrt{a} + \frac{x}{y}$
3	$a = \frac{\cos(x)+\sin(y^2)}{2}; b = \sqrt{a^{\frac{2}{4}} + \frac{4x}{y}}$
4	$a = \frac{1^y+\sin(y^2)}{2}; b = \sqrt{a^{\frac{2}{4}} + \frac{4x}{y}}$
5	$a = y^x + \frac{\sin(y^2)}{2x}; b = \sqrt{a^{\frac{2}{4}} + \frac{y}{4}}$
6	$a = y^x + \frac{1}{2y+3/x}; b = \sqrt{a^{\frac{2}{4}} + \cos x}$
7	$a = y^x + \frac{\sqrt{4}}{\sqrt{2y+3/x}}; b = \sqrt{a^{\frac{2}{4}} - \cos x}$
8	$a = y^{x-0.1} + \frac{\sqrt{4}}{\sqrt{2y+3/x}}; b = \sqrt{a^{\frac{2}{4}} - \frac{\cos x}{3y}}$
9	$a = y^x + \frac{\sqrt{4}}{\sqrt{2y+3/x}}; b = \sqrt{a^{\frac{2}{4}} - \frac{\cos x}{3y}}$
10	$a = y^x + \frac{1}{2y*3/x}; b = a^x + \frac{\sqrt{4}}{\sqrt{2+3/x}}$
11	$a = \sqrt{a^{\frac{2}{4}} + \frac{4y^3x}{y}}; b = \sqrt{a^{\frac{2}{4}} - \frac{4x}{y}}$
12	$a = \frac{\sin(x)+4}{\cos(y^2)}; b = \sqrt{a^{\frac{2}{4}} - \frac{4x}{y}}$