Урок 1

Введение в платформу Java

Введение в платформу Java, инструменты разработчика, написание первой программы. Переменные, типы данных, арифметические операции. Методы. Условные операторы.

Оглавление

Особенности платформы Java

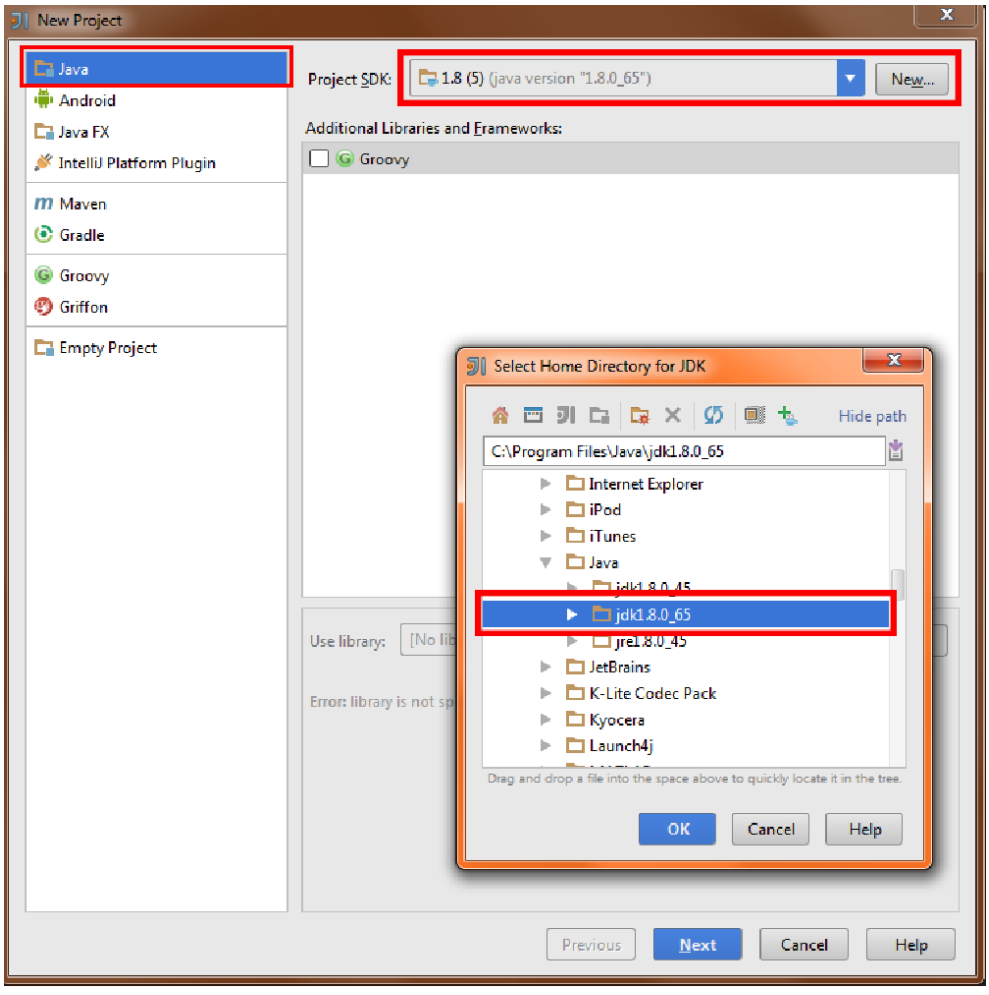
Простота	Язык Java обладает лаконичными, тесно связанными друг с другом и легко усваиваемыми языковыми средствами, был задуман как простой в изучении и эффективный в употреблении язык программирования
Безопасность	Java предоставляет безопасные средства для создания интернет-приложений
Переносимость	Программы на Java могут выполняться в любой среде, для которой имеется исполняющая система Java (например, Windows, Linux, Android, MacOS и т.д.)
Объектно-ориентированный характер программирования	В Java воплощена современная философия объектно-ориентированного программирования
Надежность	Java уменьшает вероятность появления ошибок в программах, благодаря строгой типизации данных и выполнению соответствующих проверок во время выполнения. Java исключает ошибки по работе с памятью за счет автоматического управления резервированием и освобождением памяти.
Многопоточность	Язык Java обеспечивает встроенную поддержку многопоточного программирования и предоставляет множество удобных средств для решения задач синхронизации многих процессов, которое позволяет строить устойчиво работающие интерактивные системы.
Архитектурная независимость	Язык Java не привязан к конкретному типу вычислительной машины или архитектуре операционной системы и следует принципу "написано однажды, выполняется везде, в любое время и всегда".
Интерпретируемость и высокая производительность	Java предоставляет байт-код, обеспечивающий независимость от платформы. Выполняя компиляцию программ в промежуточное представление, называемое байт-кодом, Java позволяет создавать межплатформенные программы, которые будут выполняться в любой системе, на которой реализована виртуальная машина JVM. Байт-код Java максимально оптимизируется для повышения производительности

Инструменты разработчика

Для написания программ необходимо установить инструменты разработчика Java Development Kit (JDK). Этот комплект свободно предоставляется компанией Oracle. На момент написания методического пособия (последний квартал 2016 года) текущей является версия JDK 8, применяемая на платформе Java SE 8 (*Standard Edition*). JDK можно загрузить по следующему адресу: http://www.oracle.com/technetwork/java/javase/downloads/index.html

Среду разработки IntelliJ IDEA можно скачать с сайта разработчиков (для обучения достаточно бесплатной версии Community Edition): https://www.jetbrains.com/idea/#chooseYourEdition

При создании первого проекта в IntelliJ IDEA необходимо указать путь к установленному JDK, как показано на рисунке ниже. Project SDK -> New -> JDK -> путь к папке jdk1.x.x_xxx.



Первая программа

```
/**
 * Created by GeekBrains on 15.11.2016.
 */
public class MainClass {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}
```

В первых строчках кода мы видим так называемые комментарии, они могут быть как однострочные (начинаются с символов //), так и многострочные (блок текста, заключенный в /* … */). Комментарии предназначены для упрощения работы с кодом, разработчик может делать всяческие пометки, которые не будут влиять ни на размер, ни на скорость выполнения полученной программы.

```
// Однострочный комментарий
/*
…
Блочный комментарий
…
*/
```

После комментариев идет объявление класса MainClass (название класса, должно обязательно совпадать с названием файла в котором он создан).

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

Далее идет объявление метода main(), выполнение любой Java программы начинается с вызова этого метода. Ключевое слово void сообщает компилятору, что метод main() не возвращает никаких значений. Как будет показано далее, методы могут также возвращать конкретные значения. Следует иметь в виду, что в Java учитывается регистр символов, и Main не равнозначно main. Что такое модификаторы доступа (public, private), ключевое слово static, и String[] args будет объяснено позже, когда зайдет разговор об основах ООП и массивах.

В следующей строке кода System.out.println("Hello, World!"), которая находится в теле метода main(), в консоль выводится текстовая строка "Hello, World!" с последующим переводом каретки на новую строку. На самом деле вывод текста на экран выполняется встроенным методом println(), который кроме текста может также выводить числа, символы, значения переменных и т.д.

В языке Java все операторы обычно должны оканчиваться символом ; . Причина отсутствия точки с запятой после закрытия кодовых блоков (там где блок закрывается скобкой }) состоит в том, что формально эти фигурные скобки не являются операторами.

Переменные и типы данных

Переменные представляют собой зарезервированную область памяти для хранения данных, при этом, в зависимости от типа переменной, операционная система выделяет память и решает, что именно должно в ней храниться.

В Java существует две группы типов данных:

- Примитивные;
- Ссылочные (объектные).

Существует 8 примитивных типов данных:

- byte – 8-битное знаковое целое число.

Значения: от -128 до 127.

Пример объявления переменной: byte a = 10;

- short – 16-битное знаковое целое число.

Значения: от -32768 до 32767.

Пример объявления переменной: short a = 2404;

- int – 32-битное знаковое целое число.

Значения: от -2147483648 до 2147483647.

Пример объявления переменной: int a = 200000;

- long – 64-битное знаковое целое число.

Значения: от -9223372036854775808 до 9223372036854775807.

Пример объявления переменной: long a = 15000L; (после числа ставится буква L)

- float – 32-битное знаковое число с плавающей запятой одинарной точности.

Пример объявления переменной: float a = 120.0f; (после числа ставится буква f)

- double – 64-битное знаковое число с плавающей запятой двойной точности.

Пример объявления переменной: double a = 15.72775;

- boolean – принимает два значения: true и false.

Пример объявления переменной: boolean a = true;

- char – 16-битный тип данных, предназначенный для хранения символов в кодировке Unicode.

Принимает значения от '\u0000' или 0 до '\uffff' или 65,535.

Пример объявления переменной: char c = 'A';

Насчет ссылочных типов данных разговор будет идти на следующих занятиях.

Общую структуру объявления переменной можно описать как:

```
[тип_данных] [идентификатор(имя_переменной)] = [начальное_значение];
```

```
int a = 20;
float b;
b = 2.25f;
```

Идентификаторы – имена переменных, должны начинаться с буквы, $ или _, а затем может идти любая последовательность символов. Идентификаторы чувствительны к регистру. Ключевые слова Java не могут быть идентификаторами.

К ключевым словам Java относятся:

abstract assert boolean break byte case catch char class const continue default do double else enum extends final finally float for goto if implements import instanceof int interface long native new package private protected public return short static strictfp super switch synchronized this throw throws transient try void volatile while

Для того, чтобы переменная не могла менять свое значение в процессе выполнения программы, можно определить её как константу с помощью ключевого слова final, если написать его перед указанием типа данных переменной:

```
final int a = 20;
```

Динамическая инициализация переменных:

Начальные значения переменных (volume) могут рассчитываться на основе значений других переменных (radius, height):

```
Public static void main(String args[]) {
    float radius = 2.0f, height = 10.0f;
    // volume инициализируется динамически во время выполнения программы
    float volume = 3.1416f * radius * radius * height;
    System.out.println("Объем цилиндра равен " + volume);
}
```

Инициализация нескольких переменных в одну строку:

```
int x, y, z;
x = y = z = 10; // присвоить значение 10 переменным x, y и z
float d = 2.2f, e = 7.2f;
```

Если требуется несколько переменных одного типа, их можно объявить в одном операторе через запятую.

Арифметические операции

Операция	Описание
+	Сложение
-	Вычитание
*	Умножение
/	Деление
%	Деление по модулю
++	Инкремент (приращение на 1)
+=	Сложение с присваиванием
-=	Вычитание с присваиванием
*=	Умножение с присваиванием
/=	Деление с присваиванием
%=	Деление по модулю с присваиванием
--	Декремент (отрицательное приращение на 1)

Еще одна простая программа

Вот так может выглядеть еще одна программа, написанная на языке Java.

```
public class MainClass {
    public static void main(String args[]) {
        int a;
        int b;
        a = 128;
        System.out.println("a = " + a);
        b = a / 2;
        System.out.println("b = a / 2 = " + b);
    }
}
```

Первые две строки в методе main() означают объявление двух целочисленных переменных с идентификаторами a и b. Затем в переменную a записывается число 128, и выводится сообщение в консоль – «a = 128». Затем значение переменной b вычисляется через значение переменной a, как b = a / 2 (т.е. b = 128 / 2 = 64), и в консоль выводится сообщение «b = a / 2 = 64».

Методы

Общая форма объявления метода выглядит следующим образом:

```
тип_метода имя_метода (список_параметров) {
   тело_метода;
}
```

Тип_метода обозначает конкретный тип данных (int, float, char, boolean, String и т.д.), возвращаемых методом. Если метод ничего не должен возвращать, указывается ключевое слово void. Для возврата значения из метода используется оператор return:

```
return значение;
```

Для указания имени метода служит идентификатор имя. Список параметров обозначает последовательность пар "тип_данных + идентификатор", разделенных запятыми. По существу, параметры – набор данных, необходимых для работы метода. Если у метода отсутствуют параметры, то список параметров оказывается пустым.

Несколько примеров работы с методами:

```
public static void main(String[] args) {
    System.out.println(summ(5, 5)); // для вызова метода необходимо передать ему 2 аргумента типа int, результатом работы будет целое число, которое напечатается в консоль
    printSomeText(); // для вызова метода не нужно передавать ему никаких аргументов, и он не возвращает никаких данных (метод объявлен как void)
    printMyText("Java"); // для вызова метода передаем ему в качестве аргумента строку "Java", которую он выведет в консоль
}

// метод возвращает целое число, принимает на вход два целых числа
public static int summ(int a, int b) {
    return a + b; // возвращаем сумму чисел
}

// метод ничего не возвращает, не требует входных данных
public static void printSomeText() {
    System.out.println("Hello"); // печатаем Hello в консоль
}

// метод ничего не возвращает, принимает на вход строку
public static void printMyText(String txtToPrint) {
    System.out.println(txtToPrint); // выводим строку txtToPrint в консоль
}
```

Условный оператор if

Условный оператор if позволяет выборочно выполнять отдельные части программы. Ниже приведена простейшая форма оператора if.

```
if (условие) {
   последовательность_операторов;
}
```

где условие обозначает логическое выражение. Если условие истинно(true), последовательность операторов выполняется, если ложно (false) - не выполняется. Например:

```
if (5 < 10) {
   System.out.println("5 меньше 10");
}
```

В данном примере числовое значение 5 меньше 10, и поэтому условное выражение принимает логическое значение true, а следовательно, выполняется метод println().

Рассмотрим еще один пример с противоположным условием.

```
if (10 < 5) {
   System.out.println("Это сообщение никогда не будет выведено");
}
```

Теперь числовое значение 10 не меньше 5, а следовательно, метод println() не вызывается, и в консоль ничего не выводится.

Могут быть использованы следующие операторы сравнения:

Оператор	Значение
<	Меньше
<=	Меньше или равно
>	Больше
>=	Больше или равно
==	Равно
!=	Не равно

Следует обратить внимание на то, что для проверки на равенство указывается два знака равно. Для проверки значения типа boolean используется запись:

```
public static void main(String args[]) {
    boolean bool = true;
    if (bool) { // если bool == true
        // ...
    }
    if (!bool) { // если bool == false
        // ...
    }
}
```

Ниже приведен пример программы, демонстрирующий применение оператора if.

```
public static void main(String args[]) {
    int a, b, c;
    a = 2;
    b = 3;
    if (a < b) System.out.println("a меньше чем b");
    if (a == b) System.out.println("a равно b. Это сообщение не будет
выведено");
    c = a - b; // переменная c = 2 - 3 = -1
    System.out.println("c = -1");
    if (c >= 0) System.out.println("c не отрицательно");
    if (c < 0) System.out.println("c отрицательно");
    c = b - a; // переменная c = 3 - 2 = 1
    System.out.println("c = 1");
    if (c >= 0) System.out.println("c не отрицательно");
    if (c < 0) System.out.println("c отрицательно");
}
```

Ниже приведен результат выполнения данной программы.
a меньше чем b
c = -1
c отрицательно
c = 1
c не отрицательно
Еще один вариант условного оператора if-else представлен ниже. Если условие верно, выполняется последовательность_операторов_1, если нет – последовательность_опреторов_2(из блока else).

```
if (условие) {
  последовательность операторов 1
} else {
  последовательность операторов 2
}
```

При использовании условий можно составлять более сложные конструкции, с помощью логических операторов И(&&) и ИЛИ(||).

```
if (условие1 && условие2) {
   ...
}
if (условие1 || условие2) {
   ...
}
if ((условие1 && условие2) || условие3) {
   ...
}
```

В первом случае (логическое И) для выполнения кода из блока if, необходимо чтобы и условие1, и условие2 одновременно были верны. Во втором случае (логическое ИЛИ) – достаточно чтобы хотя бы одно из условий было верно.

Домашнее задание

1. Создать пустой проект в IntelliJ IDEA и прописать метод main();
2. Создать переменные всех пройденных типов данных, и инициализировать их значения;
3. Написать метод вычисляющий выражение a * (b + (c / d)) и возвращающий результат, где a, b, c, d – входные параметры этого метода;
4. Написать метод, принимающий на вход два числа, и проверяющий что их сумма лежит в пределах от 10 до 20(включительно), если да – вернуть true, в противном случае – false;
5. Написать метод, которому в качестве параметра передается целое число, метод должен напечатать в консоль положительное ли число передали, или отрицательное; *Замечание: ноль считаем положительным числом.*
6. Написать метод, которому в качестве параметра передается целое число, метод должен вернуть true, если число отрицательное;
7. Написать метод, которому в качестве параметра передается строка, обозначающая имя, метод должен вывести в консоль сообщение «Привет, указанное_имя!»;
8. * Написать метод, который определяет является ли год високосным, и выводит сообщение в консоль. Каждый 4-й год является високосным, кроме каждого 100-го, при этом каждый 400-й – високосный.

*Если выполнение задач вызывает трудности, можете обратиться к последней странице методического пособия. Для задач со * не нужно искать решение в интернете, иначе вы теряете весь смысл их выполнения.*

Рекомендации по оформлению кода домашнего задания:

```
public class MainClass {
   public static void main(String[] args) {
      task1(); // при вызове этого метода, он сам напечатает результат в консоль
      System.out.println(task2(10)); // если метод что-то возвращает, его вызов
лучше закинуть в System.out.println(), чтобы увидеть результат его работы в
консоли
      // …
   }

   public static void task1() { // задача1, метод ничего не возвращает
      // …
      System.out.println(…); // печатается результат работы метода
   }

   public static int task2(int x) { // задача2, метод обязан вернуть целое число
      return x * 2; // метод просто возвращает результат, без печати
   }

   // …
}
```

Либо конечно можете сами выбирать стиль оформления кода, все зависит от вашего опыта программирования. Вместо taskN лучше даже подставлять имена, которые будут говорить о том, что делает данный метод.

Комментарии проставлены для объяснения работы кода, так их прописывать при выполнении домашнего задания не обязательно.

Дополнительные материалы

1. К. Сьерра, Б. Бейтс Изучаем Java // Пер. с англ. – М.: Эксмо, 2012. – 720 с.
2. Кей С. Хорстманн, Гари Корнелл Java. Библиотека профессионала. Том 1. Основы // Пер. с англ. - М.: Вильямс, 2014. - 864 с.
3. Брюс Эккель Философия Java // 4-е изд.: Пер. с англ. – СПб.: Питер, 2016. – 1168 с.
4. Г. Шилдт Java 8. Полное руководство // 9-е изд.: Пер. с англ. - М.: Вильямс, 2015. - 1376 с.
5. Г. Шилдт Java 8: Руководство для начинающих. // 6-е изд.: Пер. с англ. - М.: Вильямс, 2015. - 720 с.

Подсказки по домашнему заданию

1) Объявление метода main():
   public static void main(String[] args) { }

2) Создать переменные типов: byte, short, int, long, float, double, char, boolean;

3) public static int calculate(int a, int b, int c, int d) {
   …
   }

4) public static boolean task10and20(int x1, int x2) {
   …
   }

5) public static void isPositiveOrNegative(int x) {
   if(…) {
   System.out.println(…);
   } else {
   System.out.println(…);
   }
   }

6) public static boolean isNegative(int x) {
   if(…) {
   return true;
   }
   return false;
   }

7) public static void greetings(String name) {
   System.out.println(…);
   }

Вместо … подставляете ваш код. Представлены не все существующие решения, возможно вы найдете свое.

Как пропишите все методы, метод main() может выглядеть так:

```
public static void main(String[] args) {
  System.out.println(calculate(2, 2, 2, 2));
  System.out.println(task10and20(5, 6));
  isPositiveOrNegative(-30);
  …
}
```