Лабораторная работа № 1: Введение в Java

1. Запустите онлайн среду разработки "Tutorials Point" с рабочим пространством для языка программирования Java версии 8 по ссылке:

`https://www.tutorialspoint.com/compile_java8_online.php`

Интерфейс среды разработки состоит из трех областей: с левой стороны располагается каталог текущих файлов проекта, с правой стороны – редактор кода программы, под редактором кода находится терминал для ввода команд. В ходе выполнения лабораторных работ будет необходимо создавать программы и запускать их командами из терминала. Для сохранения файлов проекта можно использовать локальный диск ("Project" → "Download project") или диск Google ("Project" → "Save project" → "Save on Google Drive").

Язык программирования Java

Java является объектно-ориентированным языком программирования. Практически все данные в Java представляют собой объекты, за исключением примитивных типов данных. Это строго типизированный язык программирования с трансляцией разработанных приложений в байт-код, что позволяет использовать Java на любой платформе. Java обладает большим количеством преимуществ, включая автоматическое управление памятью, набор встроенных коллекций (массивы, списки, стеки и т.д.), средства создания сетевых приложений и многое другое.

Простейшая программа на языке Java:

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

В приведенном примере объявлен общедоступный (`public`) класс `HelloWorld`, имеющий общедоступный статический (`static`) метод `main` без возвращаемого значения (`void`), который осуществляет вывод на экран текста `"Hello, World!"`.

Компиляция программ осуществляется с использованием команды `javac`, после которой указывается имя файла:

```
javac HelloWorld.java
```

Запуск программ осуществляется с использованием команды `java`, после которой указывается имя запускаемого класса:

```
java HelloWorld
```

Для вывода информации в консоль используется класс `System`, реализующий стандартные методы работы с входными и выходными потоками данных. Для вывода информации в консоль может быть использован метод `println()` поля класса для выходного потока `out`, осуществляющий вывод переданного в скобках значения с добавлением перевода строки.

Наличие главного метода класса определяет, является ли данный класс запускаемым. Главный метод характеризуется спецификатором доступа `public` (является

общедоступным), модификатором `static` (метод статичен), типом `void` (метод без возвращаемого значения) именем `main` и аргументом `String[] args`, представляющим собой массив значений, переданных в командной строке при вызове метода.

2. Создайте файл `FirstClass.java`:

```
public class FirstClass {
    public static void main(String[] args) {
        int i = 5;
        i += 7;
        System.out.println("i = " + i);
        double d = (double) i / 8;
        System.out.println("d = " + d);
        char c1 = 'n';
        char c2 = 110;
        char c3 = 111;
        System.out.println("c1=" + c1 + " & " + "c2=" + c2);
        boolean b1 = (c1 == c2);
        boolean b2 = (c1 == c3);
        System.out.println(b1);
        System.out.println(b2);
    }
}
```

Запустите программу и удостоверьтесь в ее работоспособности.
Ознакомьтесь с выведенной информацией. Пример результата выполнения программы приведен ниже:

```
i = 12
d = 1.5
c1=n & c2=n
true
false
```

Примитивные типы

Язык программирования Java оперирует понятиями объектов и классов, реализующих методы, которые могут быть выполнены над соответствующими объектами. Исключение составляют 8 примитивных типов, представленных в таблице, для которых возможно использовать лишь базовые арифметические и логические операции. Простые типы являются упрощением базовых классов, реализующих основные методы работы с данными. При необходимости можно использовать приведение типов путем указания нужного типа в скобках перед выражением.

Примитивный тип	Диапазон значений	Основной класс
`boolean`	`true, false`	Boolean
`char`	Unicode: 0 .. 2^{16}-1	Character
`byte`	-128 .. 127	Byte
`short`	-2^{15} .. 2^{15}-1	Short

`int`	$-2^{31} .. 2^{31}-1$	Integer
`long`	$-2^{63} .. 2^{63}-1$	Long
`float`	IEEE 754	Float
`double`	IEEE 754	Double
`void`	-	Void

Основным отличием примитивных типов от классов является отсутствие методов работы с переменными таких типов, а также формат их записи: названия классов в языке Java всегда начинаются с прописной буквы, в то время как названия примитивных типов прописываются строчными буквами.

Описание основных арифметических и логических операций можно найти в спецификации языка Java:

`https://docs.oracle.com/javase/specs/jls/se8/html/jls-4.html#jls-4.2.2`

3. Дополните файл `FirstClass.java`. Создайте целочисленную переменную `year` со значением года Вашего рождения и переменную `div` типа `double`, записав в нее результат деления значения переменной `year` на 5. Выведите значение переменной `div` на экран.

4. Дополните файл `FirstClass.java`. Создайте переменные логического типа `l1` и `l2` со значениями `true` и `false` соответственно. Поочередно осуществите вывод на экран результаты выполнения всех возможных логических операций над переменными `l1` и `l2`.

5. Создайте файл `SecondClass.java`:

```
public class SecondClass {
    public static void main(String[] args) {
        String s = "13.7";
        Double a = new Double(s);
        char c = "qwe".charAt(2); // символ со 2-й позиции
        System.out.println(a);
        System.out.println(c);
    }
}
```

Запустите программу и удостоверьтесь в ее работоспособности.
Ознакомьтесь с выведенной информацией. Пример результата выполнения программы приведен ниже:

```
13.7
e
```

Объекты

Классы в языке Java определяют типы создаваемых объектов. Создание объекта какого-либо класса осуществляется с помощью вызова конструктора класса с использование ключевого слова `new`. Для основных классов, представленных в таблице, и класса

String, предназначенного для работы со строками, возможно опускать использование ключевого слова new. Таким образом, объекты таких классов могут быть созданы несколькими способами, представленными далее, первый из которых является общим для создания объектов всех классов. Все объекты должны быть объявлены. При объявлении объекта без указания его значения, присваивается пустое значение null.

```
Integer i; // объекту присвоено значение null
int a = 5;
Integer b = new Integer(a);
Integer c = 6;
```

Для работы со строками предназначен класс String, объекты которого так же могут быть объявлены различными способами. Вызов методов классов осуществляется с помощью указания имени метода через точку от имени объекта данного класса:

```
String s1 = "Hello, ";
String s2 = s1.concat("World!"); // конкатенация строк
System.out.println(s2); // вывод в консоль значения s2
```

Следует отметить, что под "работой с объектами" в языке Java подразумевается работа со ссылками на объекты. Полная информация, включающая описание встроенных классов языка Java и соответствующих методов доступна по ссылке:

https://docs.oracle.com/javase/8/docs/api/overview-summary.html

6. Дополните файл SecondClass.java. Создайте новый объект iS целочисленного типа Integer из строки "135" двумя способами: с помощью конструктора с использованием ключевого слова new и с помощью метода класса Integer получения числового значения из строки.

7. Дополните файл SecondClass.java. Создайте строку s1 "Java is one of the best languages!". Осуществите вывод на экран результатов следующих действий, выполненных с использованием методов класса String:

- получение символа, находящегося на 5 позиции;
- сравнение строки со строкой "Java is one of the most beautiful languages!";
- поиска в строке подстроки "best".

8. Создайте файл ThirdClass.java:

```
public class ThirdClass {
    public static void main(String[] args) {
        double[] arr = {0.5, 1.3, 2.7, 0.2};
        Arrays.sort(arr);
        System.out.println(Arrays.toString(arr));
        System.out.println(arr[2]);
        System.out.println(arr.length);
    }
}
```

Запустите программу и удостоверьтесь в ее работоспособности.

Ознакомьтесь с выведенной информацией. Пример результата выполнения программы приведен ниже:

```
[0.2, 0.5, 1.3, 2.7]
1.3
4
```

Массивы

В языке программирования Java существует множество различных коллекций. Для примера рассмотрим класс `Arrays`, реализующий методы работы с массивами в Java. Объявление массивов происходит аналогично другим языкам программирования. Индексация элементов массива начинается с 0.

```
int[] arr;
arr = new int[3]; // резервируем память на 3 элемента массива
// заполняем массив значениями
arr[0] = 5;
System.out.println(arr[0]);
// заполнение массива при инициализации
int[] a = {0, 1, 2, 3};
```

Класс `Arrays` содержит статичные методы (подробнее режимы доступа и модификаторы рассмотрены в следующем блоке), принимающие на вход объекты массивов, над которыми необходимо выполнять действия. Статичные поля и методы классов могут быть вызваны напрямую от самого класса. Для использования методов работы с массивами необходимо подключить класс `Arrays` пакета `java.util` с помощью ключевого слова `import`:

```
import java.util.Arrays; // подключение Arrays
```

Пример работы с массивами:

```
double[] arr = {0.5, 1.3, 2.7, 0.2};
Arrays.sort(arr); // сортировка массива
System.out.println(Arrays.toString(arr)); // вывод значений
```

9. Дополните файл `ThirdClass.java`. Создайте массив строк `stringArray`. Заполните его значениями, осуществив разделение строки "Peter Piper picked a peck of pickled peppers" по символу пробела. Осуществите вывод на экран результатов выполнения следующих действий:

- вывод значений всего массива, используя метод класса Arrays;
- вывод на экран значения пятого элемента массива;
- сортировка массива, используя метод класса Arrays;
- поиск элемента "peppers" в массиве, используя метод класса Arrays.

10. Создайте файл `FourthClass.java`:

```
public class FourthClass {
    public static void main(String[] args) {
        int p = 0;
        for (int i = 0; i < 30; i++) {
```

```
            if (i % 2 == 0) {
                double d = (double) i / 4;
                System.out.print(d+"; "); // вывод в одну строку
            }
        }
        System.out.println();
        int year = 2016;
        switch (year) {
            case 2014:
                System.out.println("You're 3rd year student");
                break;
            case 2015:
                System.out.println("You're 2nd year student");
                break;
            case 2016:
                System.out.println("You're 1st year student");
                break;
        }
    }
}
```

Запустите программу и удостоверьтесь в ее работоспособности.
Ознакомьтесь с выведенной информацией. Пример результата выполнения программы приведен ниже:

```
0.0 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0 5.5 6.0 6.5 7.0
You're 1st year student
```

Базовые конструкции Java

Условный оператор:

```
if (<условие>) {
    <операторы>
}
else if (<условие>) {
    <операторы>
}
else {
    <операторы>
}
```

Оператор `switch`:

```
switch (<переменная>) {
    case <значение>:
        <операторы>
        break;
    . . .
    default:
        <операторы>
        break;
}
```

Оператор цикла `for`:

```
for (<переменная>;<условие>;<оператор>) {
    <операторы>
}
```

Оператор цикла `while`:

```
while (<условие>) {
    <операторы>
}
```

Оператор цикла `do..while`:

```
do {
    <операторы>
} while (<условие>);
```

Для операторов цикла могут быть использованы операторы выхода из цикла (`break`) и перехода на следующую итерацию (`continue`).

Области видимости

Области видимости переменных примитивных типов определяются фигурными скобками `{}`. Таким образом, внутренние переменные условий, циклов и других выделенных фигурными скобками блоков будут не доступны в блоках, являющихся внешними по отношению к данным. При этом внутренние блоки могут использовать переменных внешних блоков. В случае с объектами видимость так же ограничивается блоками фигурных скобок, однако объект, созданный с использованием ключевого слова `new`, остается в памяти и после завершения блока.

Одним из главных преимуществ языка программирования Java является отсутствие необходимости удаления (разрушения) объектов после окончания работы с ними. Для этого существует "Сборщик мусора" (garbage collector), который автоматически очищает память, занимаемую объектами, созданными с помощью `new`, когда ссылка на них, записанная в переменную перестает использоваться программой.

11. Дополните файл `FourthClass.java`. Осуществите вывод на экран таблицы истинности для функции трех логических переменных $x1 \wedge x2 \vee \neg x3$ с использованием трех видов циклов.

12. Дополните файл `FourthClass.java`. Осуществите последовательный вывод на экран чисел кратных 7 и меньших 130. Остановите работу цикла с помощью оператора остановки по достижении значения в три раза превосходящего число 23.

13. Создайте файл `Planet.java`:

```
public class Planet {

}
```

Класс

Для создания собственного класса Java необходимо создать файл <Имя_класса>.java. Имя файла должно полностью совпадать с именем класса. Имена классов принято начинать с прописной буквы. Объявление класса происходит с помощью ключевого слова class.

```
class MyClass {
    . . .
}
```

Объявление объектов определенного класса осуществляется следующим образом:

```
MyClass obj = new MyClass();
```

14. Создайте общедоступный класс Satellite.

15. Модифицируйте класс Planet:

```
public class Planet {

    String name;
    Double radius;
    Double sunDistance;

}
```

Поля класса

Для класса определяются поля определенных типов, которые должен содержать данный класс. Для каждого поля обязательно указывается его тип. Поля класса рекомендуется определять сразу после строки объявления класса, например:

```
class MyClass {
    int century;
    int year;
    . . .
}
```

Если значения полей класса не заданы, для них будут установлены стандартные нулевые значения соответствующих типов.

Доступ к полям объекта класса осуществляется с указанием названия поля класса через точку после имени объекта:

```
obj.century = 21;
```

16. Дополните класс Satellite. Добавьте классу Satellite следующие поля:
- name – название;
- radius – содержит значение радиуса спутника;
- period – период обращения спутника (в часах).

17. Модифицируйте класс `Planet`:

```
public class Planet {

    String name;
    Double radius;
    Double sunDistance;

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public double toThousandKm(String param){
        double result = 0;
        switch (param){
            case "sunDistance":
                result =  this.sunDistance / 1000;
            case "radius":
                result =  this.radius / 1000;
        }
        return result;
    }
}
```

Методы класса

Методы класса определяют действия, которые могут быть выполнены над его объектами. Для каждого метода класса обязательно указывается тип возвращаемого значения или `void`, в случае, если метод ничего не возвращает, и имя метода. Формат описания метода представляется следующим образом:

```
<тип_возвращаемого_значения> <имя_метода> (<аргументы>) {
    <операторы>
    [return <значение>;]
}
```

Например:

```
class MyClass {
    int century;
    int year;

    double division (double value) {
        return this.year / value;
    }
}
```

В описанном примере метод `division` возвращает остаток от деления значения поля `year` объекта класса `MyClass` на переданное методу в качестве аргумента значение. Ключевое слово `this`, использованное внутри метода представляет собой текущий объект класса, для которого был вызван метод `division`.

18. Дополните класс `Satellite`. Добавьте классу `Satellite` следующие методы:
    - `getPeriod` –возвращает значение периода обращения спутника;
    - `getPeriodInDays` – осуществляет перевод и возвращает значение периода обращения спутника в днях;
    - `print` – осуществляет вывод информации об объекте (значения всех полей объекта класса), не возвращает значений;

19. Модифицируйте класс `Planet`:

```
public class Planet {

    String name;
    Double radius;
    Double sunDistance;

    public Planet(String name, Double radius, Double sunDistance)
{
        this.name = name;
        this.radius = radius;
        this.sunDistance = sunDistance;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public double toThousandKm(String param){
        double result = 0;
        switch (param){
            case "sunDistance":
                result =  this.sunDistance / 1000;
            case "radius":
                result =  this.radius / 1000;
        }
        return result;
    }
}
```

20. Создайте файл `FifthClass.java`:

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

```
        Planet planet = new Planet("Earth-01", 6371.0, 149.6);
        System.out.println(planet.name);
        planet.setName("Earth");
        System.out.println(planet.getName());
        System.out.println(planet.toThousandKm("radius"));
    }
}
```

Запустите программу и удостоверьтесь в ее работоспособности.
Ознакомьтесь с выведенной информацией. Пример результата выполнения программы приведен ниже:

```
Earth-01
Earth
6.371
```

Конструкторы класса

Для каждого класса по умолчанию создается пустой конструктор. Определение собственных конструкторов происходит следующим образом:

```
class MyClass {
    int century;
    int year;

    public MyClass(int century) {
        this.century = century;
    }

    double division (double value) {
        return this.year / value;
    }
}
```

В приведенном примере общедоступный конструктор `MyClass()` принимает на вход значение целочисленного типа `century` и записывает его в качестве значения поля `century` создаваемого объекта класса.

21. Дополните класс `Satellite`. Создайте конструктор для класса, принимающий на вход значения полей `name`, `radius` и `period`.

22. Дополните класс `Planet`. Добавьте поле `satellite` типа `Satellite` и метод `getSatelliteInfo`, осуществляющий вывод информации о спутнике планеты на экран и не возвращающий значений.

23. Дополните метод `main` класса `FifthClass`, создав объект класса `Satellite` и реализовав проверку всех созданных методов.

24. Есть три стержня, на которых размещены `n` дисков таким образом, что на диске большего диаметра находится диск меньшего диаметра. В начальный момент все диски находятся на первом стержне. Напишите программу, осуществляющую перемещение дисков с первого стержня на третий, при условии, что на диске большего диаметра всегда должен находится диск меньшего диаметра.

25. Основное задание:

Создайте набор классов, описывающий систему в соответствии с вариантом. В классе, предназначенном для описания всей системы, реализуйте следующие методы:

- получение (вывод в терминал) информации о свойствах системы
- редактирование свойств системы
- получение информации об установленных компонентах
- удаление установленных компонентов
- добавление новых компонентов системы

В классах, предназначенных для описания компонентов, реализуйте следующие методы:

- получение (вывод в терминал) информации о свойствах компонента
- редактирование свойств компонента

Создайте запускаемый класс, в котором осуществите проверку всех свойств и методов реализованных классов системы и ее компонентов.

Вариант	**Описание сложного объекта**
1	Система: «Автомобиль» Компоненты: «Двигатель», «Кузов», «Колеса»
2	Система: «Судно» Компоненты: «Двигатель», «Корпус», «Мачты»
3	Система: «Самолет» Компоненты: «Двигатель», «Корпус», «Топливный бак»
4	Система: «Компьютер» Компоненты: «Монитор», «Клавиатура», «Мышь»
5	Система: «Плита» Компоненты: «Конфорки», «Духовка», «Корпус»
6	Система: «Кафедра» Компоненты: «Образовательные программы», «Преподаватели»
7	Система: «Факультет» Компоненты: «Дирекция», «Кафедры», «Деканат»
8	Система: «Университет» Компоненты: «Административные подразделения», «Образовательные подразделения», «Научные подразделения»
9	Система: «Программное обеспечение» Компоненты: «Документация», «Программный код», «Разработчики»
10	Система: «Холодильник» Компоненты: «Холодильная камера», «Морозильная камера», «Корпус»

11	Система: «Кинотеатр» Компоненты: «Экран», «Колонки», «Зал»
12	Система: «Театр» Компоненты: «Сцена», «Оркестровая яма», «Зал»
13	Система: «Ресторан» Компоненты: «Кухня», «Зал»
14	Система: «Завод» Компоненты: «Оборудование», «Материалы»
15	Система: «Мобильный телефон» Компоненты: «Аккумулятор», «Материнская плата», «Корпус»
16	Система: «Лаборатория» Компоненты: «Помещения», «Реагенты»
17	Система: «Строительная компания» Компоненты: «Проекты», «Оборудование»
18	Система: «Часы» Компоненты: «Корпус», «Стрелки», «Механизм»
19	Система: «Мультимедиа-студия» Компоненты: «Помещения», «Аудиооборудование», «Видеооборудование»
20	Система: «Морской порт» Компоненты: «Судна», «Расписание», «Причалы»
21	Система: «Космический аппарат» Компоненты: «Двигатели», «Кабина», «Корпус»
22	Система: «Банк» Компоненты: «Помещения», «Валюта», «Окна»
23	Система: «Страховая компания» Компоненты: «Специализация», «Юридическая служба», «Администрация»
24	Система: «Кухня» Компоненты: «Мебель», «Оборудование», «Посуда»
25	Система: «Ванная комната» Компоненты: «Сантехника», «Мебель»
26	Система: «БРИКС» Компоненты: «Члены», «Наблюдатели», «Саммиты»
27	Система: «ООН» Компоненты: «Члены», «Руководство», «Официальные документы»
28	Система: «Город» Компоненты: «Районы», «Транспорт», «Администрация»
29	Система: «Федеративное государство» Компоненты: «Субъект федерации», «Органы государственной власти»
30	Система: «Унитарное государство» Компоненты: «Области», «Органы государственной власти»
31	Система: «Склад» Компоненты: «Специализация», «Помещения», «Администрация»
32	Система: «Автомобильная заправочная станция» Компоненты: «Бензоколонки», «Пункты заправки электромобилей», «Платежный терминал»
33	Система: «Аэропорт» Компоненты: «Самолеты», «Терминалы», «Расписание»
34	Система: «Поезд» Компоненты: «Локомотив», «Вагоны»
35	Система: «Железнодорожный вокзал» Компоненты: «Платформы», «Поезда», «Расписание»

36	Система: «Школа» Компоненты: «Кабинеты», «Предметы», «Администрация»
37	Система: «Гостиница» Компоненты: «Номера», «Ресторан», «Рецепция»
38	Система: «Супермаркет» Компоненты: «Еда», «Напитки», «Хозяйственные товары»
39	Система: «Спортзал» Компоненты: «Тренажеры», «Бассейн», «Фитнес-зал»
40	Система: «Колледж» Компоненты: «Профессии», «Административные подразделения», «Образовательные подразделения»
41	Система: «Автопарк» Компоненты: «Транспортные средства», «Диспетчерская служба», «Ремонтная служба»
42	Система: «Министерство» Компоненты: «Департаменты», «Руководство», «Подведомственные организации»
43	Система: «Депо» Компоненты: «Поезда», «Диспетчерская служба», «Ремонтная служба»
44	Система: «Док» Компоненты: «Судна», «Диспетчерская служба», «Ремонтный цех»
45	Система: «Музей» Компоненты: «Экспонаты», «Помещения», «Администрация»
46	Система: «Электростанция» Компоненты: «Генераторы», «Помещения», «Администрация»
47	Система: «Автомойка» Компоненты: «Места», «Моющие средства», «Оборудование»
48	Система: «Автосервис» Компоненты: «Места», «Ремонтное оборудование»
49	Система: «Чебуречная» Компоненты: «Кухня», «Столики», «Продукты»
50	Система: «Велопрокат» Компоненты: «Станции», «Велосипеды»