Java Basics.
Java Fundamentals.
Lesson 01.

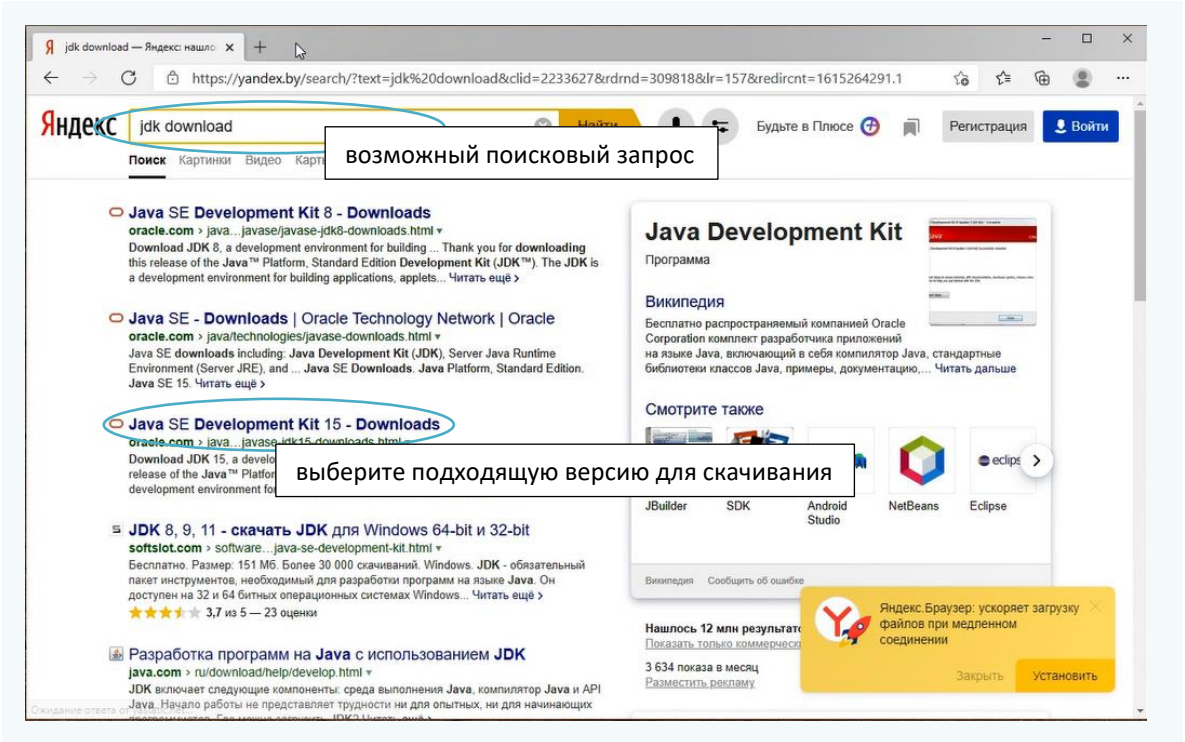
OLGA SMOLYAKOVA

План

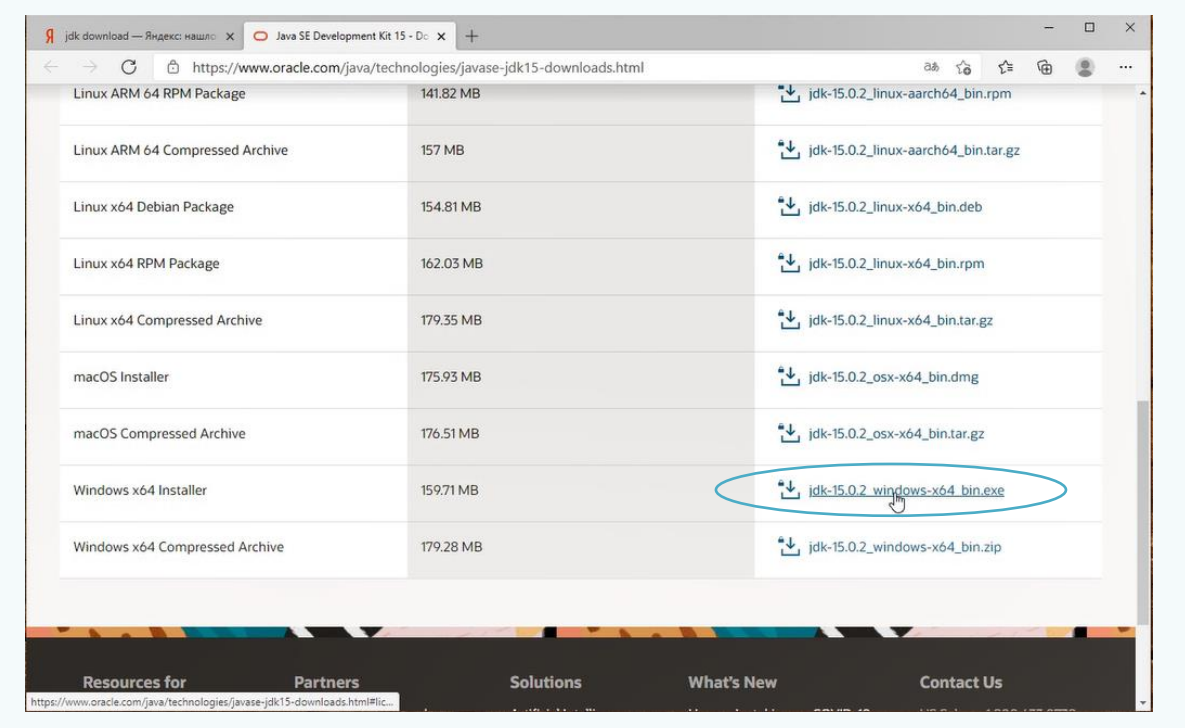
1. Установка JDK
2. Настройка переменных среды окружения (ОС Windows)
3. Пишем первую программу
4. Целочисленное деление или типы с плавающей точкой спешат на помощь
5. Литералы
6. Класс Math
7. Задачи для самостоятельного решения

1 Установка JDK

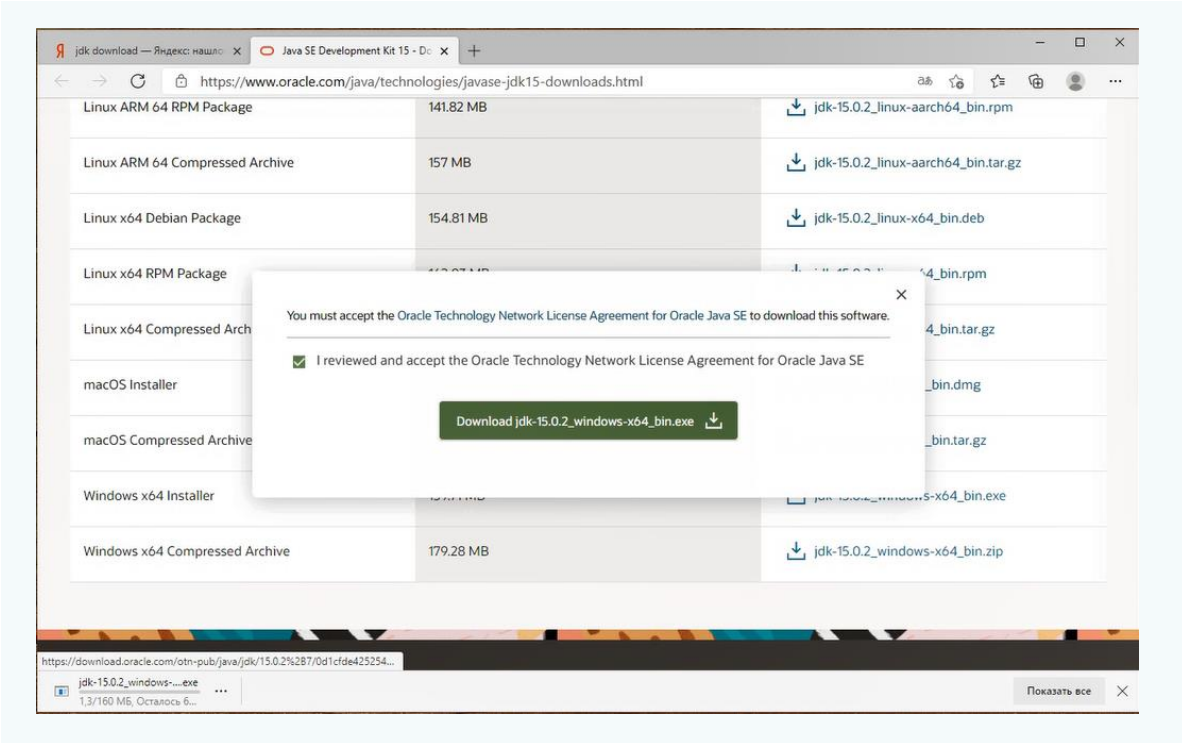
1.1 Откройте страницу для скачивания инсталяционного пакета JDK.



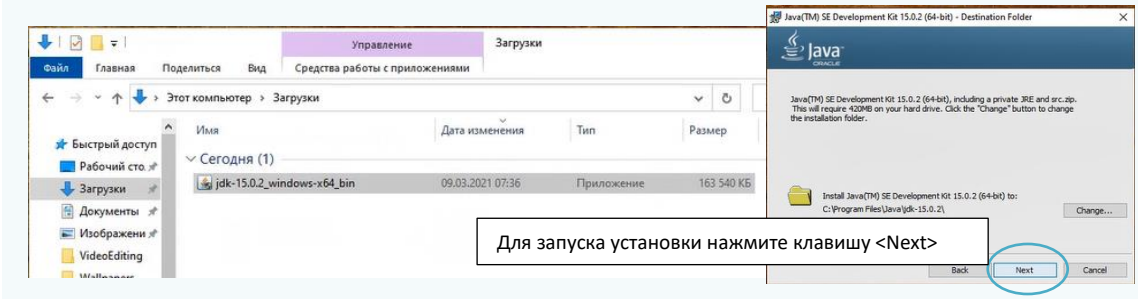
1.2 Выберите и скачайте инсталяционный пакет, подходящий для вашей ОС.



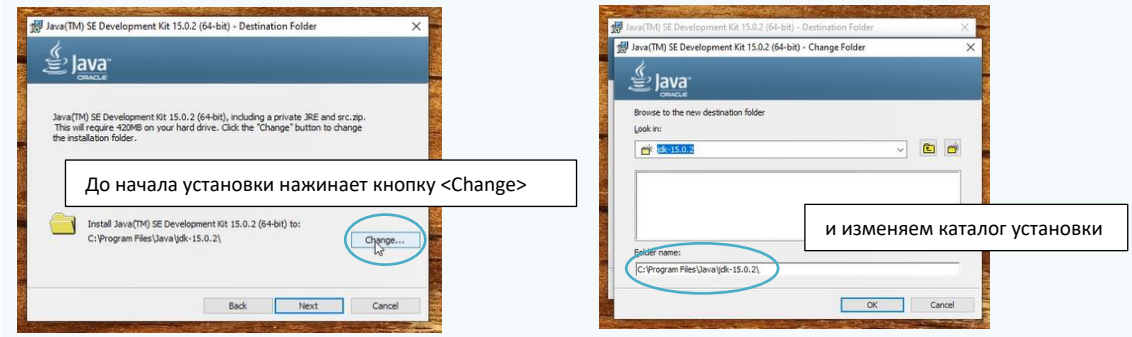
1.3 Перед скачиванием необходимо согласиться с лицензионными требованиями.



1.4 После скачивания запустите файл на выполнение.

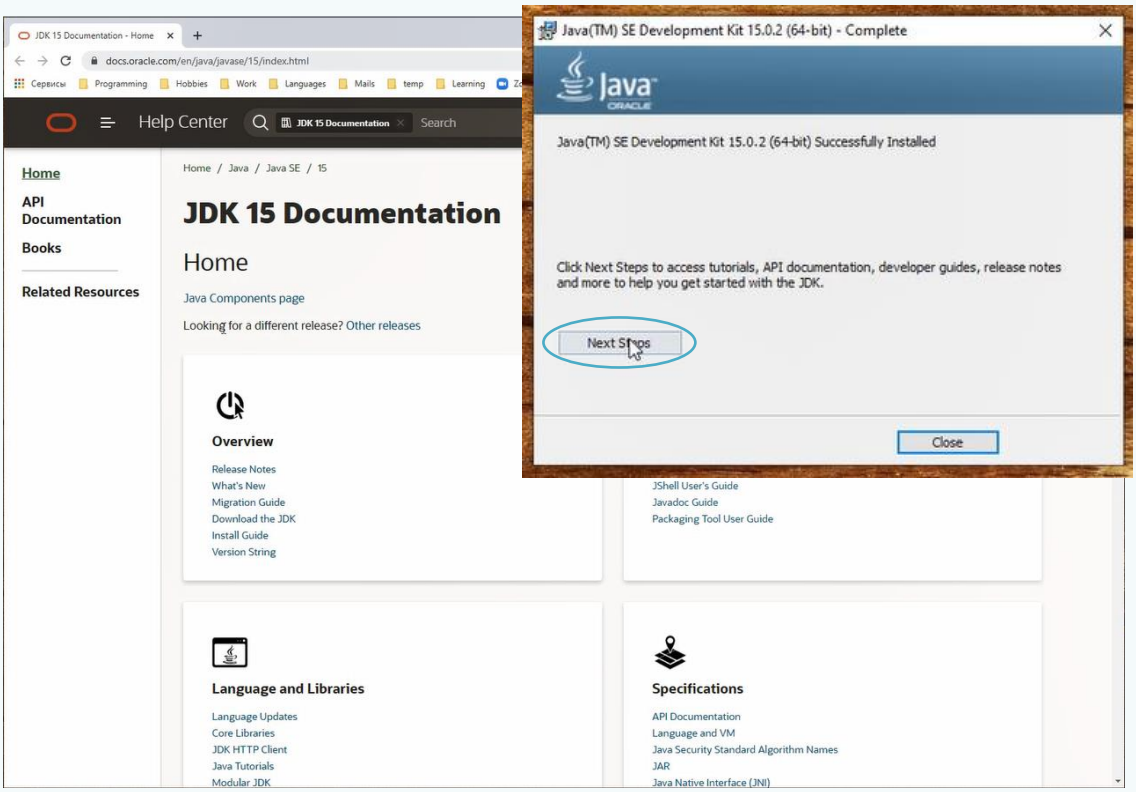


1.5 При необходимости вы можете поменять каталог установки по умолчанию.

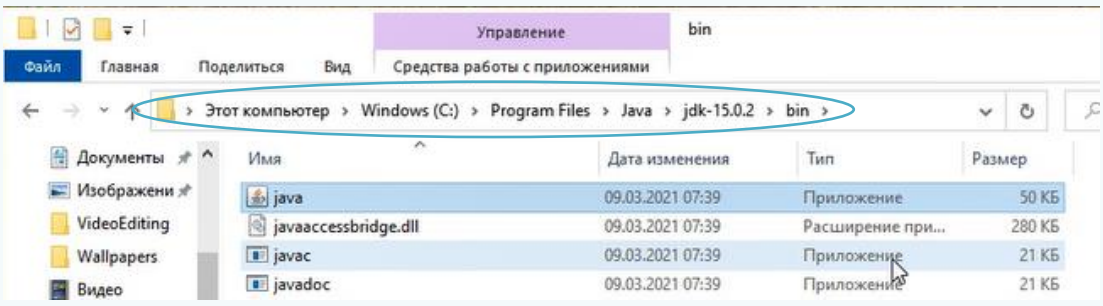


1.6 Дождитесь окончания установки и закройте окно.

1.7 Также в окне завершения установки с помощью кнопки <Next Steps> можно открыть документацию.



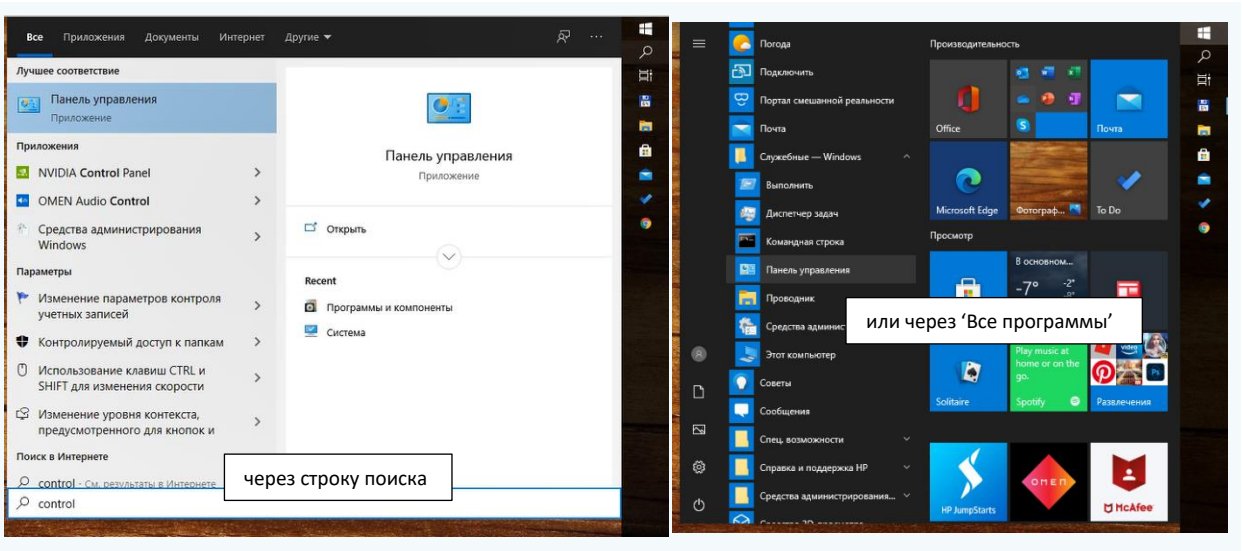
1.8 Найти JDK вы можете в каталоге установки.



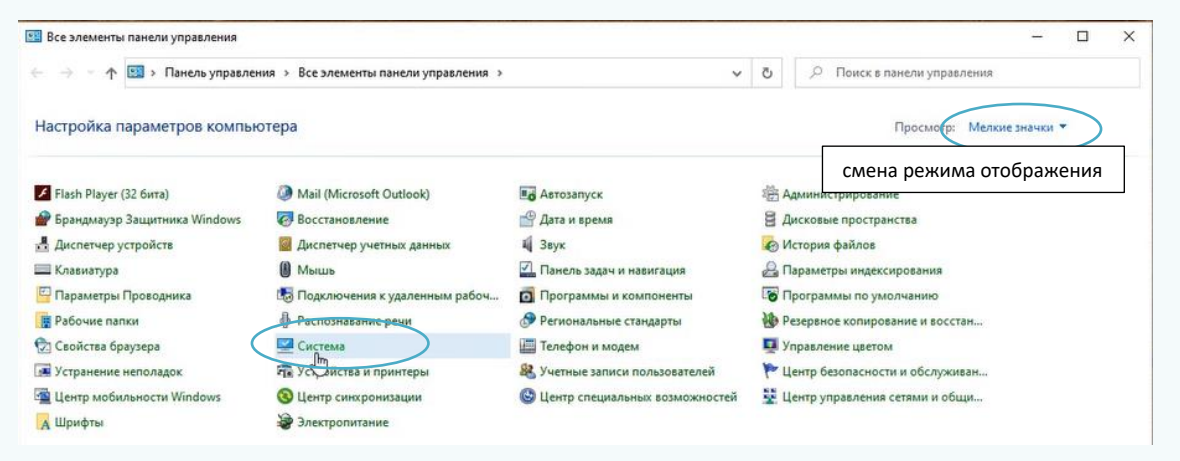
2 Настройка переменных среды окружения (ОС WIndows)

Переменные среды передают информацию и запускаемым процессам (программам), и самой операционной системе. ОС читает и использует значения переменных среды, поэтому возможно управлять некоторыми аспектами поведения операционной системы, изменяя эти переменные.
Переменная **Path** содержит список директорий, в которых операционная система пытается искать исполняемые файлы, если пользователь при запуске не указал явно путь к нужному исполняемому файлу.

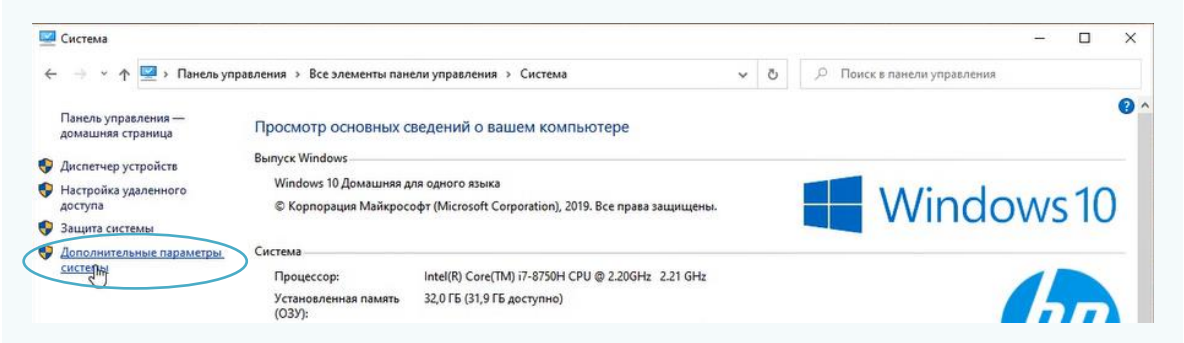
2.1 Для установки переменной среды окружения **Path** откройте *Панель управления* (*Control Panel*).



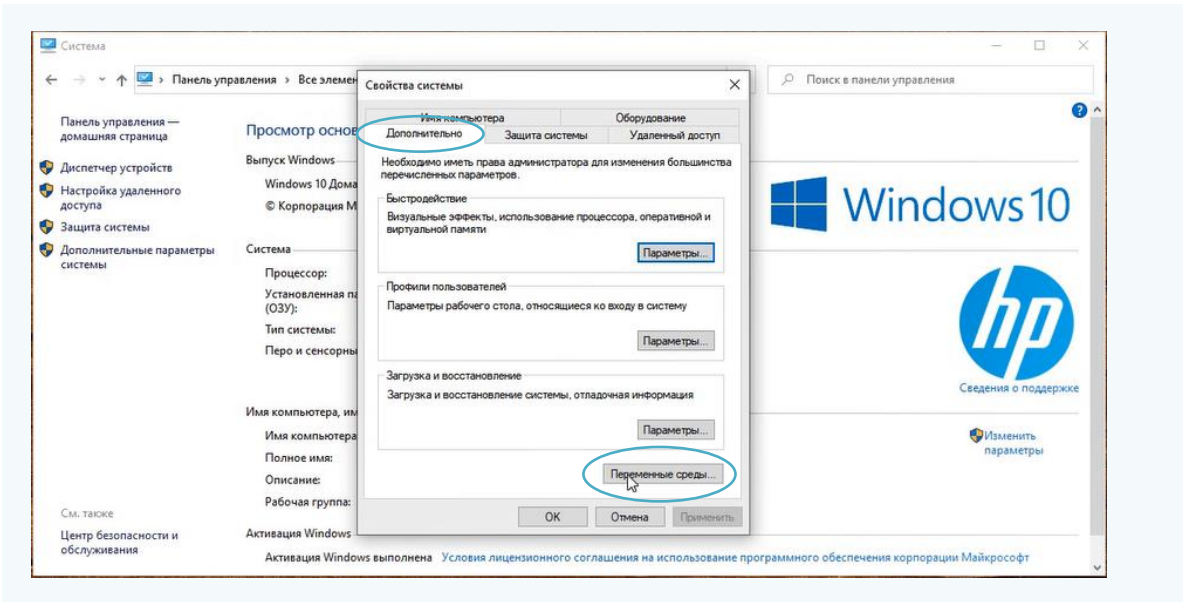
2.2 В открывшемся окне выберите пункт *Систета* (*System*). При необходимости смените режим отображения.



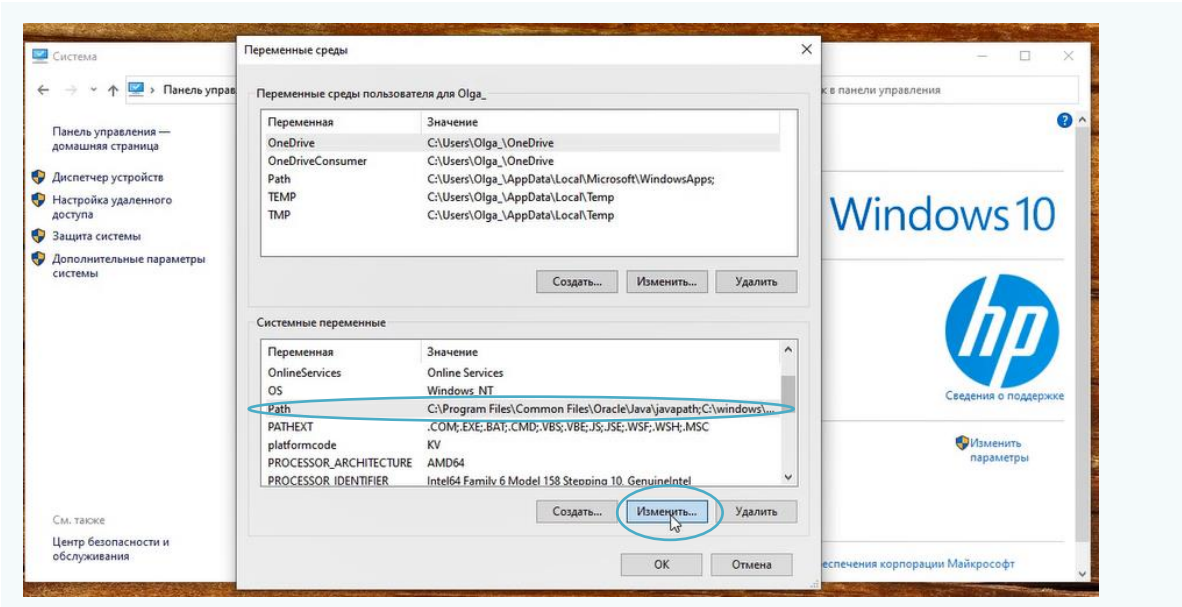
2.3 Далее выберите *Дополнительные параметры* системы.



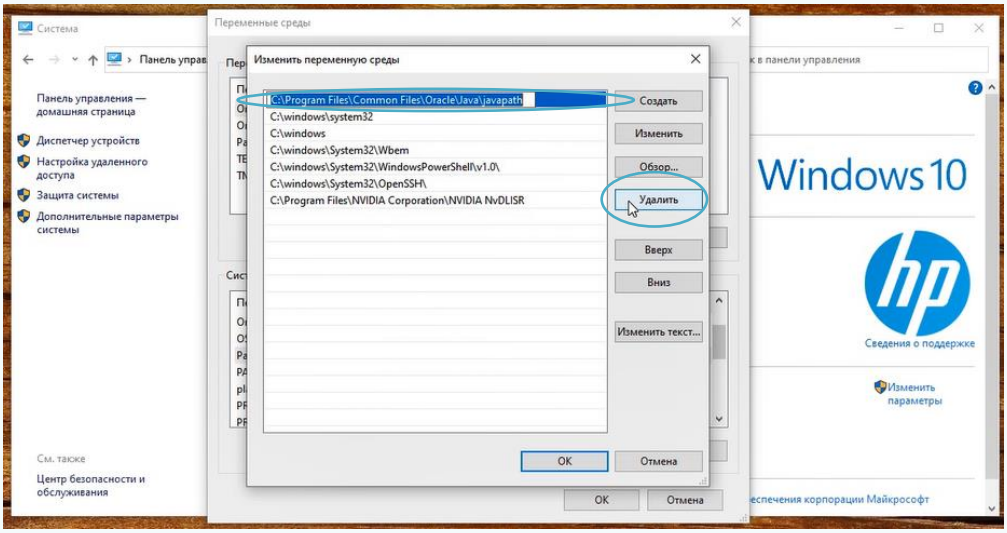
2.4 В окне *Свойства системы* перейдите на вкладку *Дополнительно* и нажмите на кнопку *Переменные среды*.



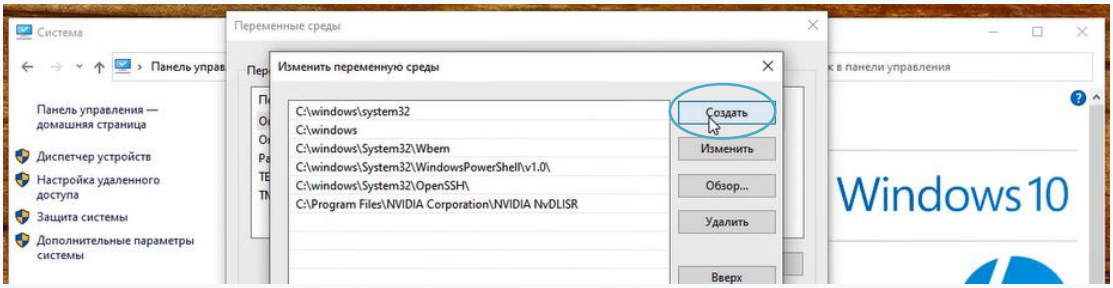
2.5 Среди *Системных переменных* найдите **Path** и нажмите кнопку *Изменить*.



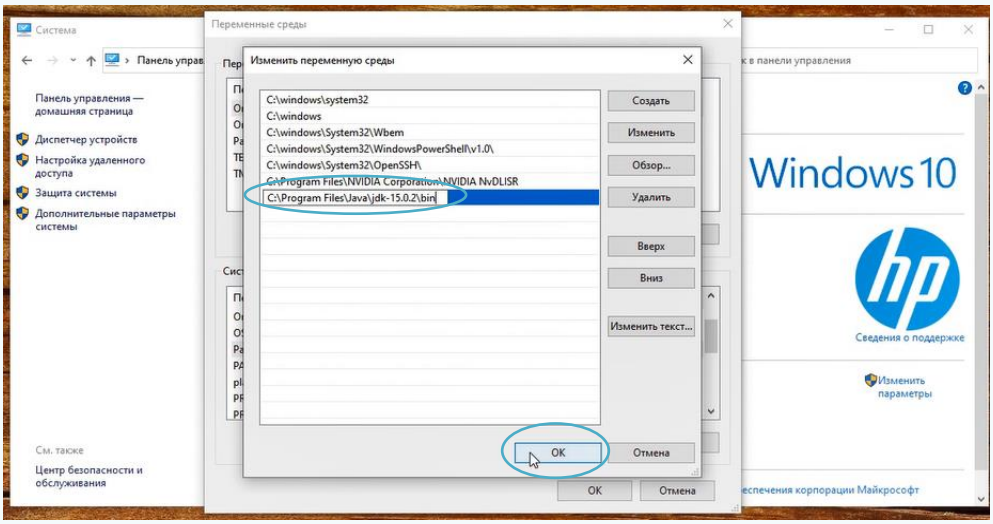
2.6 Если среди значений переменной **Path** есть '…\Program Files\Common Files\Oracle\Java\javapath' – удалите это значение.



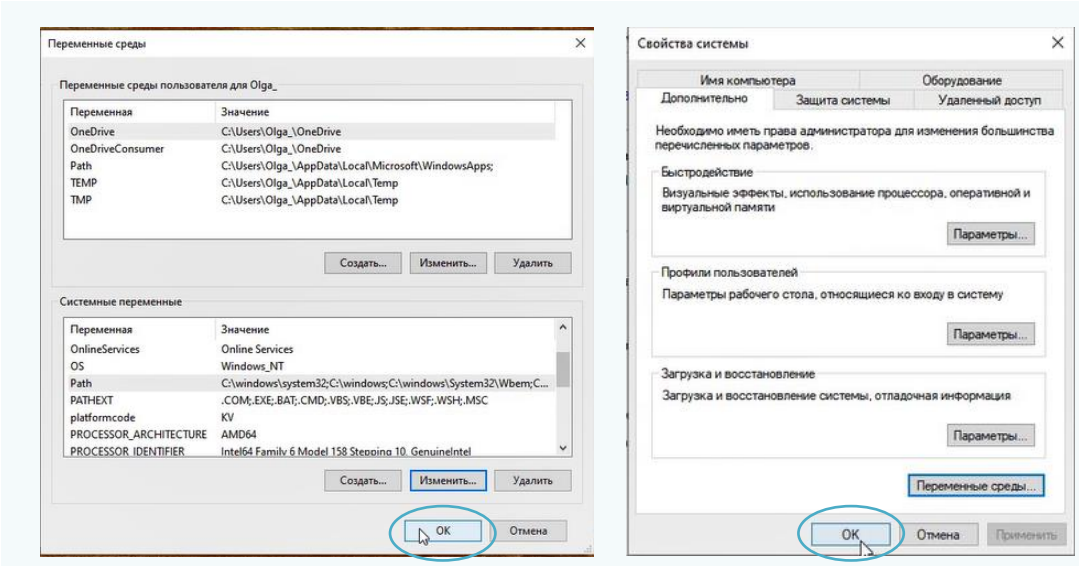
2.7 Далее создайте новое значение переменной **Path**.



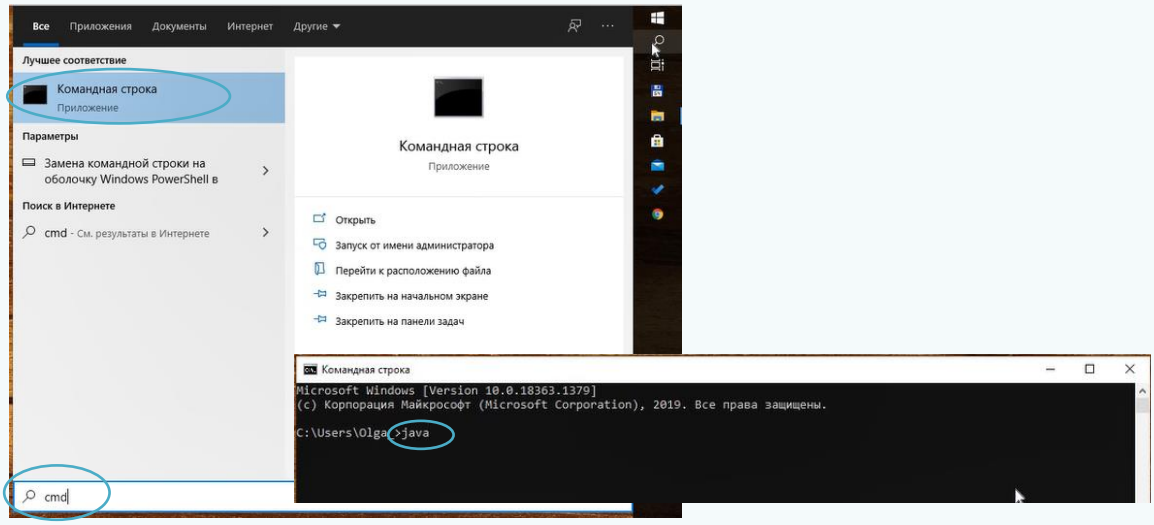
2.8 Укажите новым значением путь к каталогу bin установленного вами JDK.



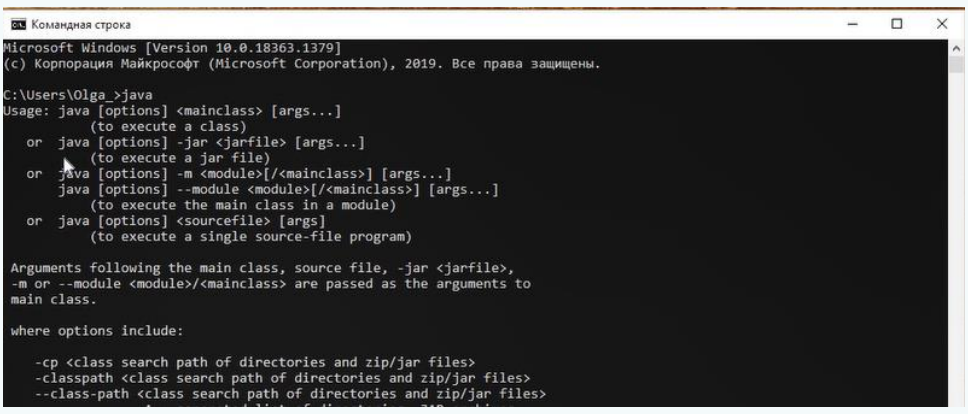
2.9 Сохраните все изменения.



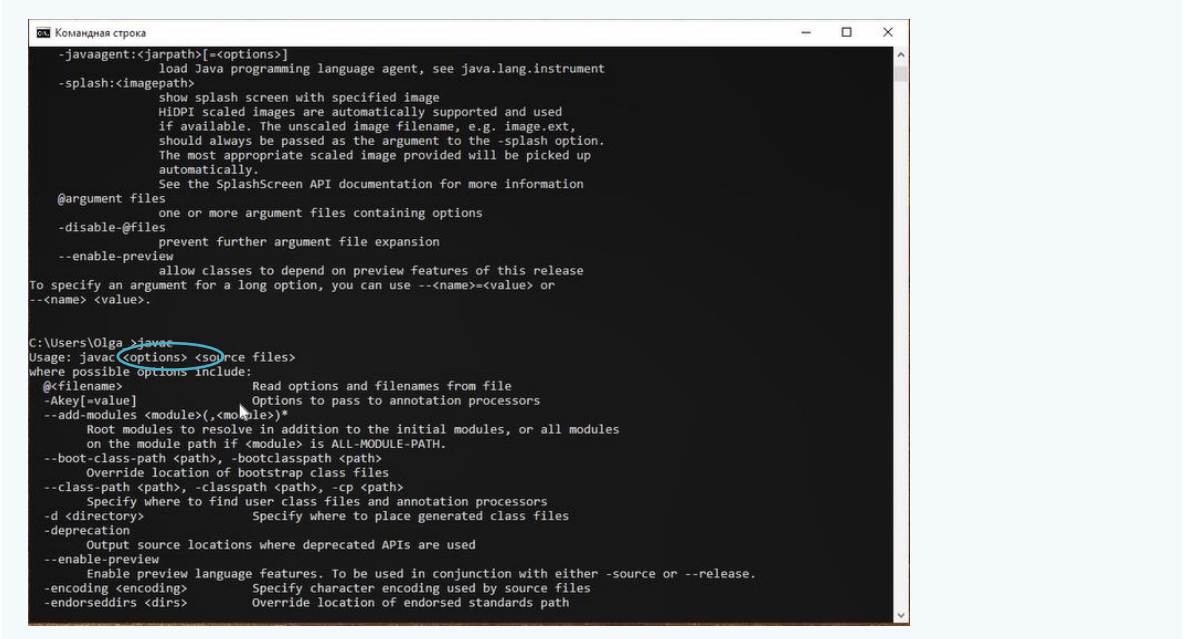
2.10 Для проверки корректности установки переменных среды окружения запустите командную строку (command line) и введите команду java.



2.11 В результате в окне консоли должна появиться справочная информация по настройке виртуальной машины.



2.12 Далее в консоли выполните команду javac (в результате в консольном окне также должны отобразиться доступные флаги компилятора).

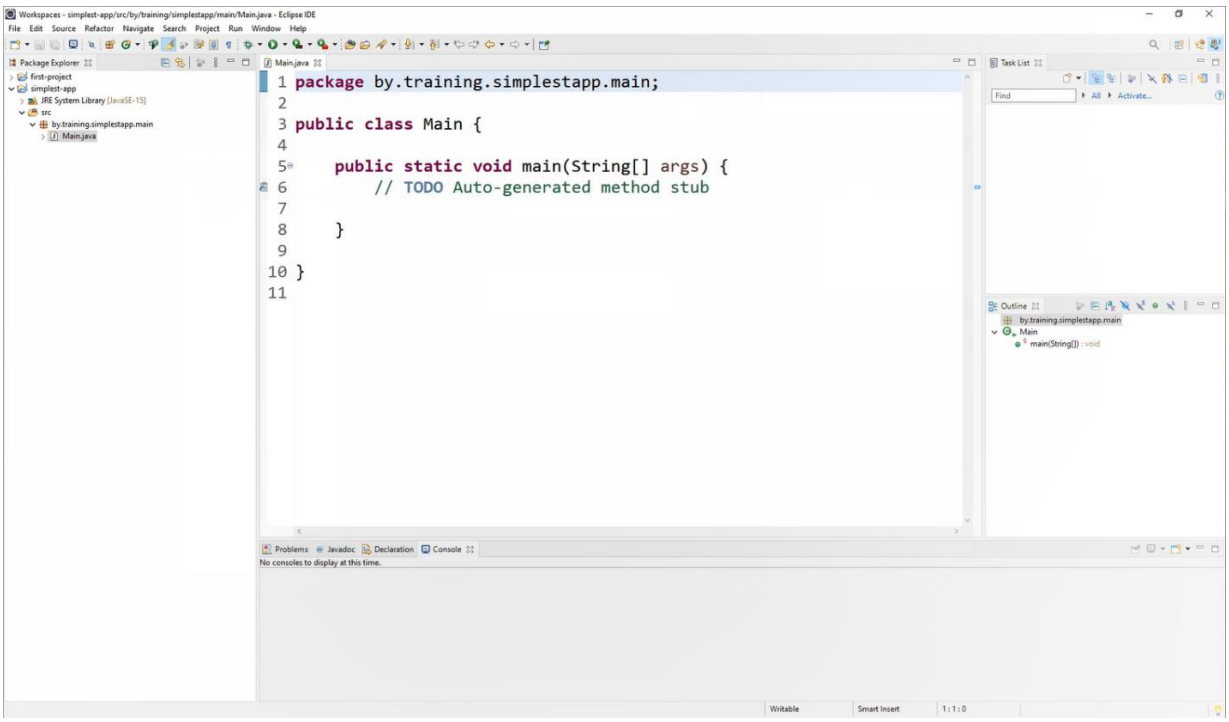


3 Пишем первую программу

3.1 Напишем программу, позволяющую складывать и вычитать числа.

Для этого:

- создайте в Eclipse консольный проект (**Java Project**).
- создайте класс **Main**, который положите в пакет **by.training.simplestapp.main**.
- В созданном классе **Main** определите запускной метод **main**.



3.2 Наберите в методе main следующий код

```
public static void main(String[] args) {
	int x;
	int y;

	int sum;

	x = 9;
	y = 11;

	sum = x + y;

	System.out.println("x="+x);

}
```

3.3 Сохраните и запустите приложение. В результате на консоли вы увидите следующий вывод.

Для того, чтобы в программе хранить какие-нибудь данные необходимы переменные. Именно в переменных хранятся требуемые нам значения.

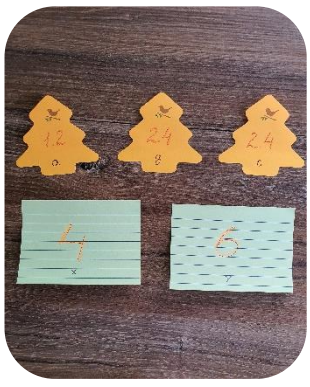
У каждой переменной есть тип (тип данных, data type), который определяет, какие именно данные можно сохранять в этой переменной.

Например, мы условимся записывать на желтые стикеры только целые числа, а на оранжевые – только дробные.

Таким образом категория желтых или оранжевых стикеров будет выступать в роли типа данных.

Сами стикеры будут переменными, в которых мы сможем сохранить информацию.

И чтобы знать, к какой переменной-стикеру мы обращаемся, ему можно придумать имя.



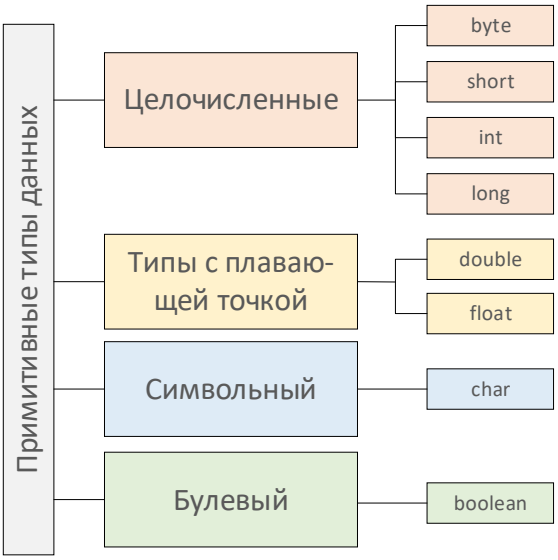
Чтобы в программе на Java объявить переменную, нужно написать

```
Тип_данных имя_переменной;
```

В Java есть две категории типов данных:

примитивные и ссылочные.

Примитивных типов данных всего восемь, ссылочных же очень много (все классы, интерфейсы, перечисления – это ссылочные типы данных).



Имена переменных (как примитивных, так и ссылочных типов) не могут начинаться с цифры. Из специальных символов допустимо применение ‘$’ и ‘_’ допустимо, в том числе и в первой позиции имени.

```
int itemsSold;
double salary;
float itemCost;
int i, k$;
double _interestRate;
```

Объявить переменную можно как без присвоения ей начального значения, так и проинициализировав при объявлении.

```
int itemsSold = 04;
double salary = 1.234e3;
float itemCost = 11.0f;
int i = 0xFd45, k$;
long simpleVar = 1_000_000_000_000L;
byte byteVar2 = 123;
```

В Java есть ряд слов, которые называют **ключевыми и зарезервированными.** Эти слова нельзя использовать в качестве имен идентификаторов (в том числе переменные этими словами также именовать нельзя).

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

Кроме ключевых слов, в Java существуют три литерала: **null**, **true**, **false**,не относящиеся к ключевым и зарезервированным словам. А также дополнительные зарезервированные слова: **const**, **goto**.

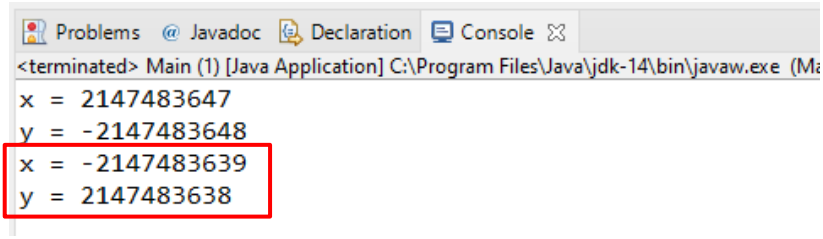
4 Диапазоны целых чисел

Добавим в программу возможность вычитать числа.

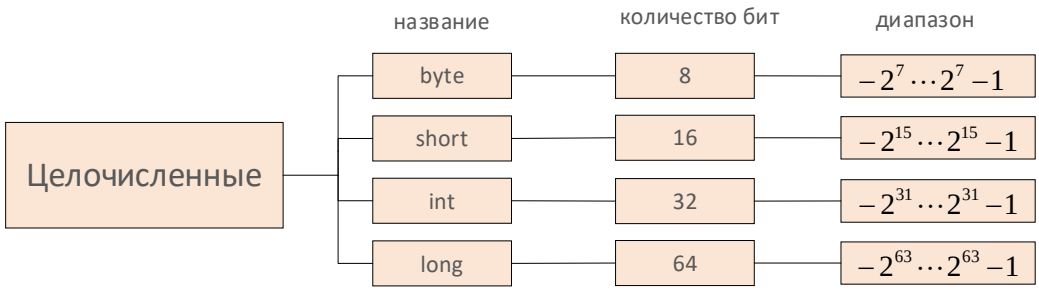
4.1 Напишем и запустим следующий код.

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

        int x = 2_147_483_647;
        int y = -2_147_483_648;

        System.out.println("x = " + x);
        System.out.println("y = " + y);

        x = x + 10;
        y = y - 10;

        System.out.println("x = " + x);
        System.out.println("y = " + y);

}
```

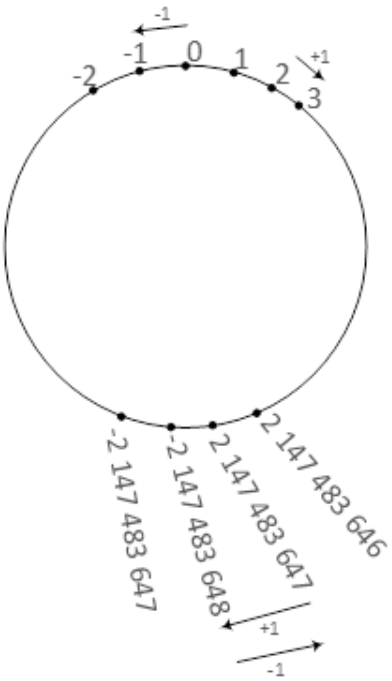
4.2 Результат должен быть следующий:



Значения переменных x и y неверны для нас и корректны для программы. Так получается потому, что целые числа имеют **диапазон**; и хранить значение, не входящее в диапазон, не могут.



Из-за представления целых чисел в памяти в двоичном виде (положительных в прямом коде, отрицательных – в дополнительном) и ограниченя размера ячейки все числа диапазона можно представить размещенными на круге. Тогда наглядно видно что происходит, когда к самому большому числу диапазона добавляют единицу (и наоборот).

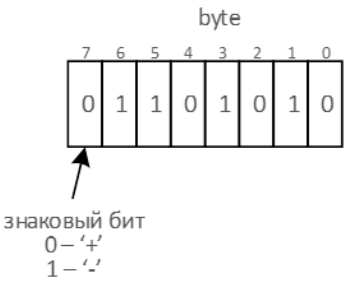


4.3 Самостоятельно исследуйте диапазоны целочисленных типов byte, short, long.

Дополнение: JVM не контролирует переполнение, т.е. исключения при такой операции вы не получите. Но можно использовать следующий код, чтобы попытаться словить переполнение.

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

		//x = 2_147_483_647;
		x = 200;

		int temp;
		temp = x + 10;

		if (temp < x) {
			System.out.println("Произошло переполнение.");
		}else {
			System.out.println("Сложение корректно.");
			x = temp;
			System.out.println("Сумма равна " + x);
		}

}
```

При хранении целочисленной переменной в памяти старший бит используется для указания знака числа, остальные хранят само число (в двоичной форме).

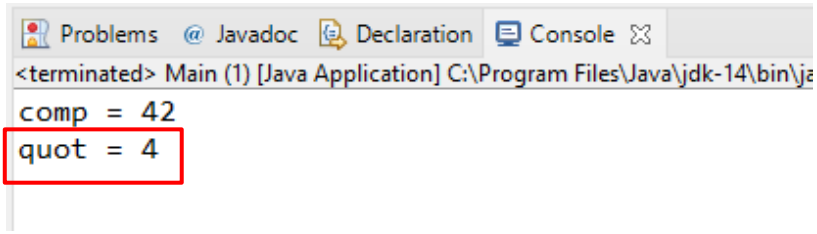


5 Целочисленное деление или типы с плавающей точкой спешат на помощь

5.1 Напишем программу которая позволяет умножать и делить целые числа.

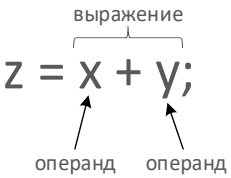
```
public static void main(String[] args) {
            int x;
            int y;

            int comp;
            int quot;

            x = 14;
            y = 3;

            comp = x * y;
            quot = x / y;

            System.out.println("comp = " + comp);
            System.out.println("quot = " + quot);
}
```

5.2 Результат должен быть следующий:



Итог умножения верен, переполнение при таких значениях мы не ожидаем сложения верен, переполнения не произошло. А вот результат деления – нет. Чтобы понять, что же случилось в коде, нужно знать о целочисленном делении.

В Java результат выражения определяется типом операндов.

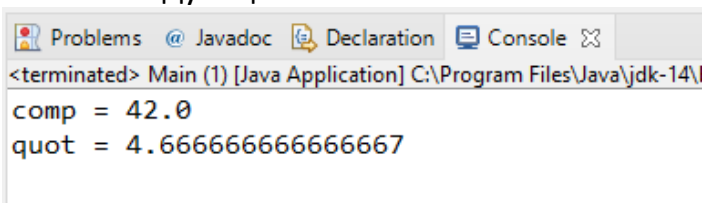


Если при делении и делимое, и делитель являются переменными (или константами) целочисленного типа, то и результат будет целым числом, т.е. вы получите частно от деления.
Чтобы при делении получить вещественное число, нужно использовать типы данных с плавающей точкой.

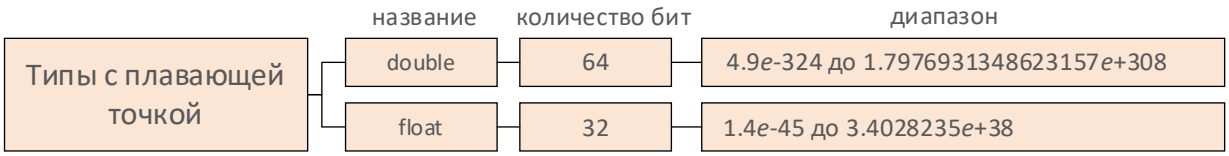
5.3 Измените тип переменных в программе и изучите результат:

```
public static void main(String[] args) {
            double x;
            double y;

            double comp;
            double quot;

            x = 14;
            y = 3;

            comp = x * y;
            quot = x / y;

            System.out.println("comp = " + comp);
            System.out.println("quot = " + quot);
}
```

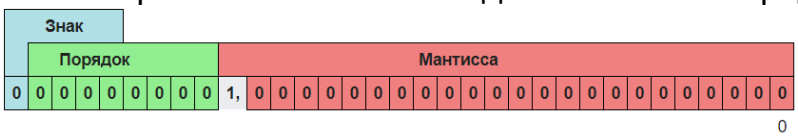
5.4 Результат должен быть следующий:



Вещественных типов в Java всего два.



Однако хранение вещественных чисел в памяти отличается от хранения целых. **double** и **float** хранятся в памяти в виде мантиссы и порядка.



Тип	Количество бит	Знак (бит)	Мантисса (бит)	Порядок (бит)
float	32	1	23	8
double	64	1	52	11

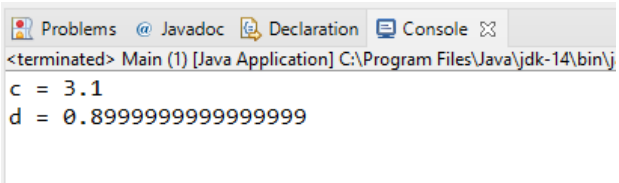
Точность **float** составляет примерно 24 бита, то есть около 7 знаков после запятой. А у **double** — точность примерно 53 бита, то есть примерно 16 знаков после запятой.

Представление вещественных чисел в памяти и особенности проведения вычислений с плавающей точкой обуславливают особенности работы с вещественными числами.

5.5 Запустите следующий пример:

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

            double a = 2.0;
            double b = 1.1;

            double c;
            double d;

            c = a + b;
            d = a - b;

            System.out.println("c = " + c);
            System.out.println("d = " + d);
}
```

5.6 Результат должен быть таким:



Итог сложения в переменной `c` ожидаем, а вот в `d` можно было ждать 0.9, а не 0.8999999.

Такое округление (возникающее как раз из-за представления вещественных чисел в ЭВМ) может доставить немало неприятностей при программировании математических алгоритмов. Однако дисциплины Численные методы и Вычислительная математика помогут с такими проблемами разобраться.

При реализации же небольших неитерационных формул погрешность, которую в вычисления может внести double или float, невелика и не переходит в значащие разряды числа – ей можно принебречь.

6 Класс Math

6.1 Вычислить значение выражения по формуле (все переменные принимают действительные значения):

$$x \ln x + \frac{y}{\cos x - \frac{x}{3}}$$

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

            double x = 0;
            double y = 0;

            double result;

            x = 4.0;
            y = 2.0;

            double temp;
            temp = Math.cos(x) - x / 3;

            if (temp != 0) {
                  result = x * Math.log(x) + y / temp;
            } else {
                  System.out.println("Знаменатель равен нулю.");
                  result = Double.NaN;
            }

            System.out.println("result=" + result);
}
```

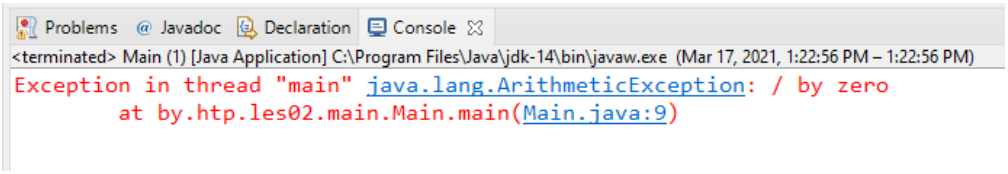
При решении задачи мы проверили знаменать, чтобы не выполнять операцию деления на ноль. Однако, что произойдет, если на ноль поделить? Оказывается есть разница, делить на целочисленный или вещественный ноль.

6.2 Посмотрим, что будет при делении на целочисленный ноль.

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

```
        int a = 4, b = 0, c;

        c = a / b;
        System.out.println("c = " + c);
}
```

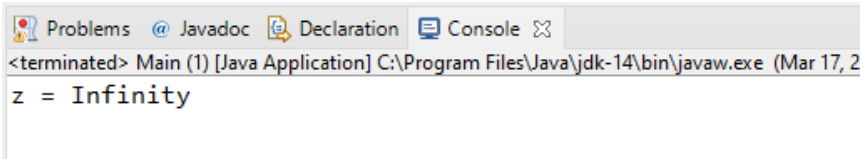
Результат:



6.3 А сейчас на вещественный.

```
public static void main(String[] args) {
        double x = 4.0, y = 0.0, z;
        z = x / y;
        System.out.println("z = " + z);
}
```

Результат:



Все вычисления, которые проводятся над числами с плавающей точкой следуют стандарту **IEEE 754**. Согласно стандарту **IEEE 754** в языке были добавлены элементы бесконечности:

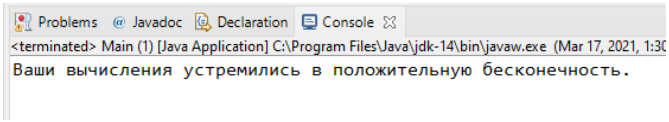
+Infinity **–Infinity** **NaN** (Not a Number, не число).

Java вводит эти значения в классах **Double** и **Float** (это ссылочные типы данных, которые изучаются в других темах)**.**

6.4 Небольшой пример применения.

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

        double x = 4.0, y = 0.0, z;

        z = x / y;

        if(z == Double.POSITIVE_INFINITY) {
              System.out.println("Ваши вычисления устремились в
положительную бесконечность.");
```

```
            }else {
                  System.out.println("z = " + z);
            }
}
```



Также при решении задачи мы использовали класс **Math**. Для организации математических вычислений в Java существует класс Math, содержащий методы, выполняющие основные алгебраические функции.

Обратите внимание на использование методов класса Math. Все методы статические и вызываются как

ИмяКласса.имяМетода([параметры]);

```
double temp;
temp = Math.cos(x) - x / 3;

if (temp != 0) {
    result = x * Math.log(x) + y / temp;
} else {
```

Класс Math является частью Java API и поставляется вместе с реализацией платформы (JDK). Класс находится в пакете **java.lang**. Однако импортировать этот пакет не нужно, т.к.

пакет java.lang импортируется в любой java-файл по умолчанию.

Приведем константы и некоторые методы класса Math.

E	2.72 (Math.E)
PI	3.14 (Math.PI)
sin(arg)	синус угла arg в радианах
cos(arg)	косинус угла arg в радианах
tan(arg)	тангенс угла arg в радианах
exp(arg)	экспонент arg
log(arg)	натуральный алгоритм arg
log10(arg)	логарифм по основанию 10 от arg
pow(x,y)	x в степени y
sqrt(arg)	квадратный корень из arg
abs(arg)	абсолютной значение arg
ceil(arg)	наименьшее целое, которое больше arg
floor(arg)	наибольшее целое, которое меньше или равно arg

max(x,y)	максимальное из двух чисел x и y
min(x,y)	минимальное из двух чисел x и y
round(arg)	возвращает arg, округленное вверх до ближайшего int
random()	возвращает псевдослучайное число между 0 и 1
toDegrees(angle)	преобразует радианы в градусы
toRadians(angle)	преобразует градусы в радианы

7 Задачи для самостоятельного решения

1	Скачайте и установите JDK и IDE. Создайте java-проект и запустите приложение Hello, world!
2	Выведите на консоль ваши ФИО, адрес и телефон.
3	Выведите на экран следующий текст. Пройдет много лет, и полковник Аурелиано Буэндиа, стоя у стены в ожидании расстрела, вспомнит тот далекий вечер, когда отец взял его с собой посмотреть на лед. Макондо было тогда небольшим селением с двумя десятками хижин, выстроенных из глины и бамбука на берегу реки, которая мчала свои прозрачные воды …
4	Напишите программу нахождения гипотенузы и площади прямоугольного треугольника по двум катетам.
5	Напишите программу вычисления суммы четырёх слагаемых.
6	Вычислить значение выражения a^2-$(b-c)^2$.
7	Вычислить значение выражения по формуле (все переменные принимают действительные значения): a) $\frac{b+\sqrt{b^2+4ac}}{2a}-a^3c+b$ b) $\frac{a}{c}*\frac{b}{d}-\frac{ab-c}{cd}$