



Laboratory Work #02

Java Basic Syntax. Java Primitive Data Types. Variables and Literals



LEARN. GROW. SUCCEED.

© 2020-2021. Department: <Software of Information Systems and Technologies>
Faculty of Information Technology and Robotics
Belarusian National Technical University
by Viktor Ivanchenko / ivanvikvik@bntu.by / Minsk

ЛАБОРАТОРНАЯ РАБОТА #02

Базовый синтаксис языка Java.

Примитивные типы данных в Java.

Переменные и литералы

Цель работы

Ознакомиться с базовым синтаксисом языка Java; приобрести навыки объявления, инициализации и использования переменных и литерал при программировании вычислительных алгоритмов; закрепить всё вышеописанное на примере разработки простейших Java-приложений.

Требования

- 1) При разработке кода можно использовать любую интегрированную среду разработки. Однако, при запуске программы рекомендуется вручную задействовать основные компоненты Java (компилятор – ***javac***, утилиту для запуска JVM – ***java***).
- 2) Если логически не подразумевается или в задании иного не указано, то входными и выходными данными являются вещественные числа (числа с плавающей запятой).
- 3) При написании кода считать, что пользователь вводит всегда корректные данные.
- 4) При разработке программ придерживайтесь соглашений по написанию кода на JAVA (*Java Code-Convention*).

Общее задание

Для закрепления написания простейших программ с использованием языка программирования Java попробуйте создать простенькие программы-кон-

верторы для различных шкал температур (из градусов Цельсия в градусы Фаренгейта или Кельвина и наоборот) или для различных валют (к примеру, из бел. руб. в евро или наоборот). Можно использовать любую предметную область для создания однотипных приложений (к примеру, конвертор значений углов из градусы в радианы и наоборот).

Основное задание

- 1) Масса динозавра задаётся в граммах. Разработайте программу, которая вычисляет, сколько это килограммов, центнеров и т.д.
- 2) Дан общий размер файла в байтах (размер задаётся в виде целого числа). Разработайте программу, которая вычисляет, сколько это килобайтов, мегабайтов и т.д.
- 3) Значение расстояния между двумя городами задаётся в сантиметрах. Разработайте программу, которая вычисляет, сколько это километров и метров.
- 4) Попробуйте разработать программу, которая меняет местами содержимое двух переменных a и b , не используя для этого дополнительные переменные (это самый первый и простой алгоритм в программировании).
- 5) Разработать программу вычисления того, сколько стоит один кг шоколадных конфет и 1 один кг желатинок, а также во сколько раз шоколадные конфеты дороже (дешевле) желатинок, если известно, что X кг шоколадных конфет стоит A бел. руб., а Y кг желатинок стоит B бел. руб.

Best of LUCK with it, and remember to HAVE FUN while you're learning :)

Victor Ivanchenko



Что нужно запомнить (краткие тезисы)

Перед написанием простейшего Java-приложения, очень важно вспомнить (запомнить) следующие вещи о Java.

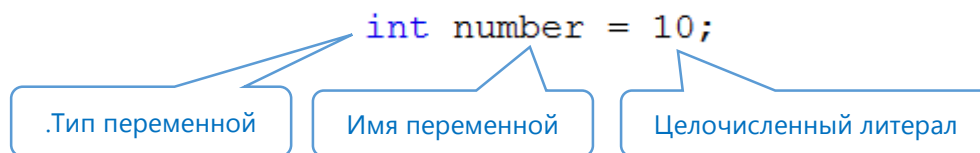
1. В Java весь код пишется в пользовательских типах данных, таких как класс, интерфейс или перечисление.
2. Имя класса всегда пишется с БОЛЬШОЙ буквы и, как правило, обозначает ответственное существительное в единственном числе!
3. Есть класс должен быть запускаемым, то в нём обязательно должен быть объявлен стартовый открытый (*public*) статический (*static*) метод *main(...)*, который на вход должен обязательно принимать массив строковых данных и ничего не возвращать, т.е. декларироваться с типом *void*. Пример общего вида стартового статического метода *main()* приведён ниже:

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

4. В одном файле исходного кода (т.е. в модуле компиляции) на языке Java можно описать сколько угодно классов, но только один из них может быть открытым (публичным), т.е. использоваться с модификатором доступа *public*. Все остальные классы описываются с модификатором доступа по умолчанию, который вообще не имеет ключевого слова, а просто подразумевается.
5. Если в файле исходного кода на языке Java описан публичный (открытый) класс, то имя файла должно в точности совпадать с именем открытого класса! Если же в файле описаны классы и среди них нет общедоступного класса, то имя файла исходного кода может быть произвольным.
6. Чтобы в процессе выполнения программы можно было хранить начальные, промежуточные и результирующие значения (данные) в языках программирования используются переменные.
7. В общем случае, **переменная** (*variable*) – поименованная область памяти, которую можно использовать для хранения данных и осуществления доступа к ним. Данные, которые находятся в переменной (т.е. по данному адресу памяти), называют **значением** (*value*) данной переменной.
8. Переменные упрощают написание кода программы, делают этот код читабельным и легко поддерживаемым.
9. **Объявление переменной** – это определение её типа и имени.

10. Перед использование любой переменной её необходимо объявить, затем инициализировать, а лишь потом только использовать. Обычно нельзя использовать неинициализированную переменную – будет синтаксическая ошибка при компилировании или интерпретировании кода.
11. Для установления соответствующего значения переменной используется самый востребованный во всех языках программирования **оператор присваивания**. В языке Python он обозначается символом '=' («равно»).
12. С каждой переменной связаны две вещи: область видимости и время жизни.
13. **Область видимости переменной** – эта область, в которой можно обращаться к данной переменной, т.е. вычислительный блок кода, где она видна.
14. **Время жизни переменной** – промежуток времени от выделения памяти под переменную до освобождения памяти, занимаемой переменной.
15. В большинстве случаев выделяют объявление локальных и глобальных переменных.
16. **Локальная переменная** – переменная, которая объявляется в любом вычислительном блоке (функции, методе и т.д.). Локальная переменная имеет локальную область видимости (видимость предела вычислительного блока) и локальное время жизни (создаётся при вызове вычислительного блока (функции, метода, ...) и уничтожается при завершении выполнения вычислительного блока. Локальная переменная обычно создаётся в стеке (*stack*).
17. **Глобальная переменная** – переменная, которая объявлена вне вычислительного блока (функции, метода, ...). Глобальная переменная видна/доступна почти везде (т.е. в любом вычислительном блоке), если её не затемняет локальная переменная. Глобальная переменная создаётся сразу же при старте программы и уничтожается при завершении выполнения всей программы.
18. В языке Java напрямую можно объявить только локальные переменные. Они объявляются внутри вычислительных блоков кода (методов, конструкторов, блоков инициализации и т.д.).
19. Перед использование любой локальной переменной её необходимо объявить, затем инициализировать, а лишь потом только использовать. Нельзя использовать неинициализированную локальную переменную – ошибка синтаксиса языка (ошибка компиляции).
20. Объявление локальной переменной заключается в определении её типа и названии идентификатора.

21. Объявление типа локальной переменной необходимо для выделения соответствующего количества памяти, которая будет ассоциироваться с данной переменной и где будут храниться сами данные, а также для определения того, какие операции могут быть использованы с данной переменной.
22. В зависимости от того, на каком этапе определяется тип переменной, все языки программирования делятся на две группы: **языки со статической типизацией** (Java, C/C++, C#, Pascal, ...) и **языки с динамической типизацией** (JavaScript, Python, Ruby, ...).
23. В **языках со статической типизацией** программист сам должен явно указать тип соответствующей переменной в коде программы. В таких языках одной и той же переменной в разные моменты выполнения программы можно присвоить значения только одного типа, который был указан при объявлении данной переменной. Язык Java относится к языкам со статической типизацией.
24. Пример объявления целочисленной переменной в Java:



25. В **языках с динамической типизацией** тип переменной определяется самой системой во время выполнения программы в зависимости от значения, которое присваивается данной переменной. В таких языках одной и той же переменной в различное время выполнения программы можно присвоить значения различного типа.
26. **Литералом** в языках программирования называется фиксированное значение, которое непосредственно используется при написании программного кода. Выделяют целочисленные, вещественные, булевские, символьные и строковые литералы, а также *null*-литерал (пустой литерал).
27. Все **целочисленные литералы** в языке Java по умолчанию имеют тип *int* и могут отображать значения в четырёх системах счисления: в двоичной (начиная с JDK 7.0), в восьмеричной, в десятичной (по умолчанию) и в шестнадцатеричной. Пример представления значения литерала 36 в различных системах счисления:

```
int decimal = 36;
int octal = 044;
int hexadecimal = 0x24;
int binary = 0b100100;
```

28. Чтобы целочисленный литерал имел тип **long**, необходимо в конце данного литерала дописать символ **'l' ('L')**. Т.к. символ **'l'** похож на единицу, то рекомендуется в коде использовать только символ **'L'**.
29. Все **вещественные литералы** в языке Java по умолчанию имеют тип **double** и две нотации написания: простую и научную (*scientific notation*).
30. Чтобы вещественный литерал имел тип **float**, необходимо в конце данного литерала дописать символ **'f' ('F')**.
31. Пример представления значений вещественных литералов 0.0005 и 12345678.9 в различных нотациях:

```
double d1 = 0.0005;      double d1 = 5.0e-4;
double d2 = 12345678.9;  double d2 = 1.23456789e+7;
```

Стандартная форма вещественного литерала

Научная форма вещественного литерала

32. В языке Java есть только два **булевских литерала** – **true** и **false**.
33. **null-литерал** присваивается переменным в тех случаях, если отсутствует объект, на который должна ссылаться переменная.
34. Все **строковые литералы** берутся в двойные кавычки и имеют тип **String**.
35. Все **символьные литералы** берутся в одинарные кавычки и имеют тип **char**.
36. Символьные литералы имеют несколько нотаций для отображения символов. Пример, в котором в различных нотациях представляется символ доллара:

```
char c1 = '$';
char c2 = '\u0024';
char c3 = '\044';
char c4 = 36;
```

Стандартная форма символьного литерала

Unicode-форма символьного литерала, где символ задаётся четырёхзначным числом в 16-ой системе счисления

Задание символа с помощью его целочисленного эквивалента

Форма символьного литерала, где символ задаётся трёхзначным числом в 8-ой системе счисления

Пример выполнения основного задания

Задание

Разработайте две программы, одна из которых конвертирует заданную температуру в градусах (*degrees*) по шкале Цельсия (*Celsius*) в температуру по шкале Фаренгейта (*Fahrenheit*), а вторая наоборот.

Решение

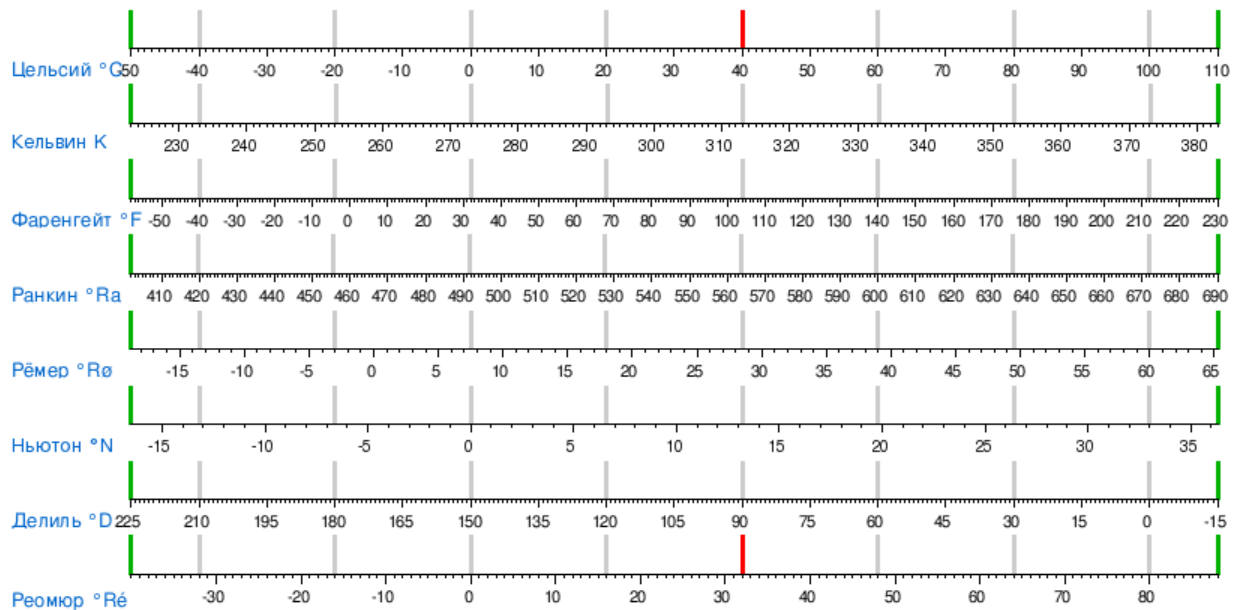
- Известно (источник: https://ru.wikipedia.org/wiki/Градус_Фаренгейта), что формула для преобразование температуры по шкале Фаренгейта в температуру по шкале Цельсия выглядит следующим образом:

$$t_C = \frac{5}{9} * (t_F - 32),$$

а формула для преобразования температуры по шкале Цельсия в температуру по шкале Фаренгейта:

$$t_F = \frac{9}{5} * t_C + 32.$$

Для общего развития приводим диаграмму перевода температур:



$$40\text{ }^{\circ}\text{C} = 313,15\text{ K} = 104\text{ }^{\circ}\text{F} = 563,67\text{ }^{\circ}\text{Ra} = 28,5\text{ }^{\circ}\text{Rø} = 13,2\text{ }^{\circ}\text{N} = 90\text{ }^{\circ}\text{D} = 32\text{ }^{\circ}\text{R}$$

- Перед написание первого приложения разработаем вербальное (словесное) описание последовательности действий для достижения результата:
 - объявление целочисленной переменной для хранения первоначального значения градусов по шкале Цельсия;

- b) объявление вещественной переменной для сохранения результата конвертации градусов по шкале Цельсия в градусы по шкале Фаренгейта;
 - c) инициация целочисленной переменной соответствующим значением;
 - d) выполнение выражения по конвертации градусов согласно указанной формулы и присвоение вещественной переменной
 - e) вывод полученного результата на консоль.
- 3) Исходный класс первой программы представлен ниже:

```

1  public class FirstApplication {
2      public static void main(String[] args) {
3          int celsiusDegree;
4          double fahrenheitDegree;
5
6          celsiusDegree = 22;
7
8          fahrenheitDegree = 9 / 5.0 * celsiusDegree + 32;
9
10         System.out.printf(
11             "\n%d Celsius degrees is %.2f Fahrenheit degrees\n",
12             celsiusDegree, fahrenheitDegree);
13     }
14 }

```

- 4) Результат работы первой программы:

```

Output - Sample (run)
run:
22 Celsius degrees is 71.60 Fahrenheit degrees
BUILD SUCCESSFUL (total time: 0 seconds)

```

- 5) Давайте рассмотрим вышеописанную программу более подробно:

5.1) объявление класса в исходном файле *FirstApplication.java*:

Модификатор доступа `public` – говорит о том, что класс будет доступен везде

Имя открытого (`public`) класса должно всегда совпадать с именем исходного файла и быть всегда существительным в единственном числе

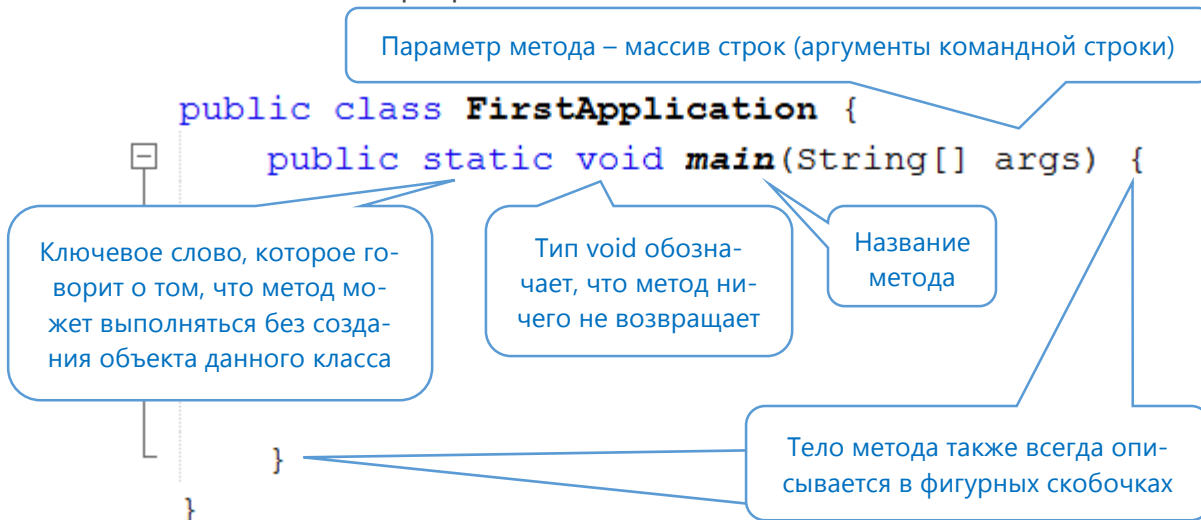
```
public class FirstApplication {
```

Ключевое слово для обозначения объявления класса

Тело класса всегда описывается в фигурных скобках

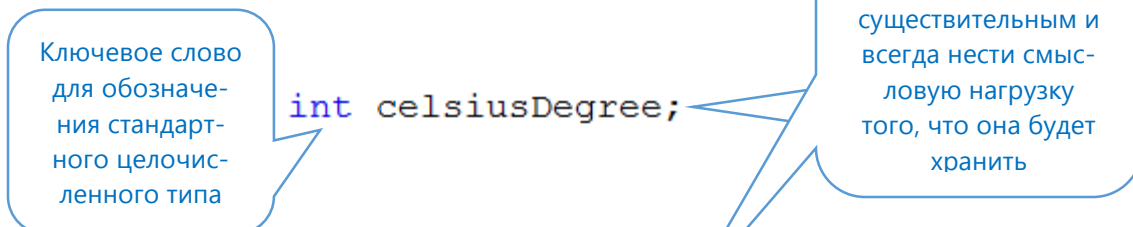
```
}
```

5.2) объявление метода *main(...)* – основного метода, с которого начинается выполнение всей программы:

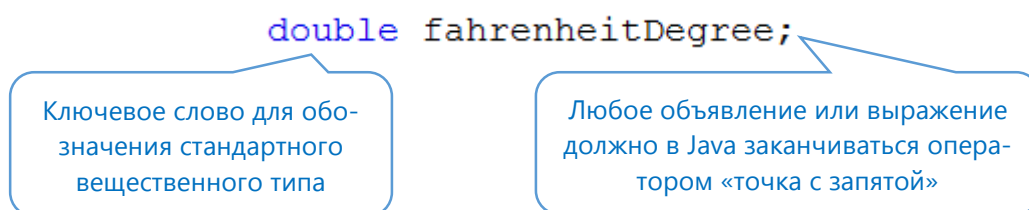


Отметим, что синтаксис главного стартового метода *main()* почти всегда остаётся неизменным.

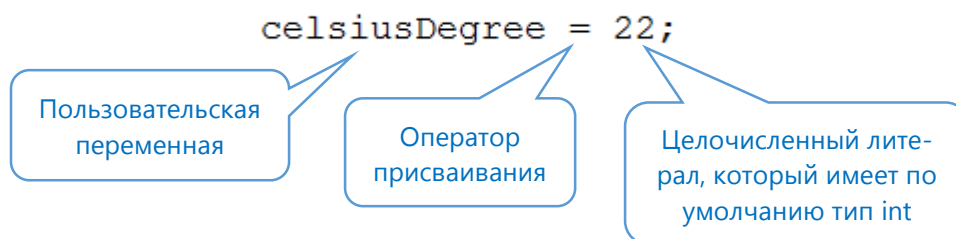
5.3) объявление целочисленной переменной:



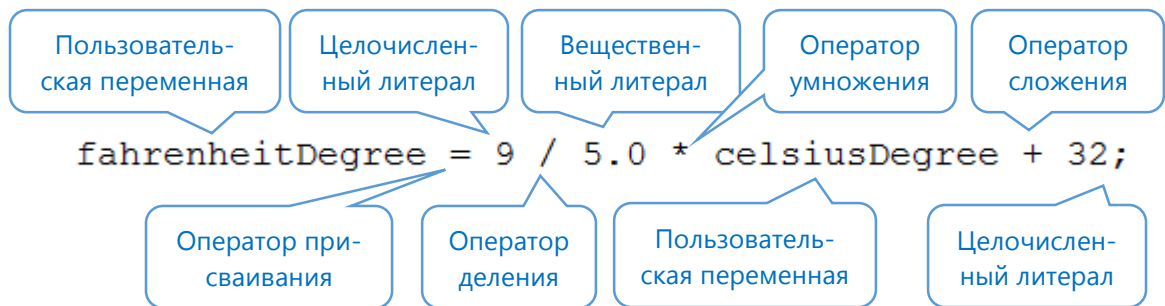
5.4) объявление вещественной переменной:



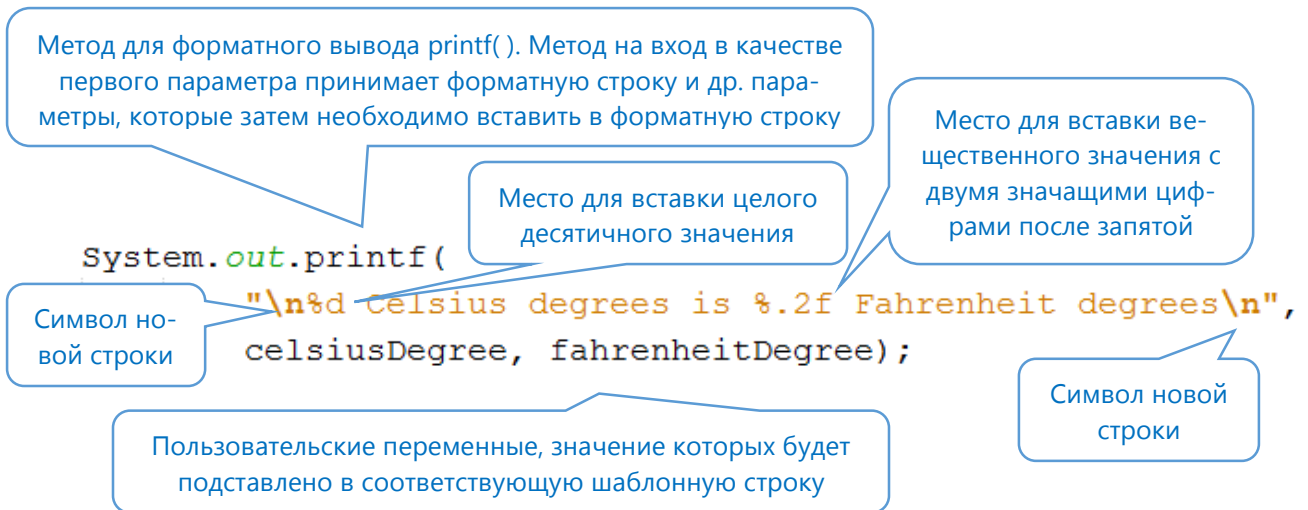
5.5) инициализация целочисленной переменной с помощью оператора присваивания:



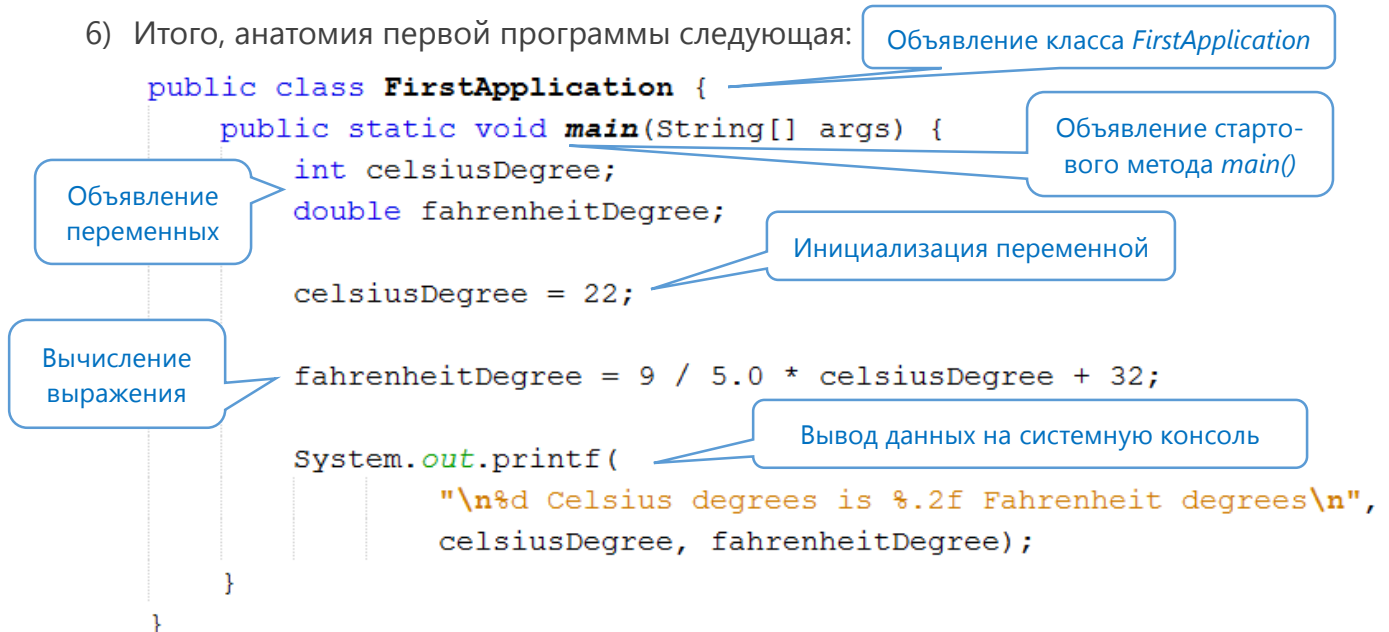
5.6) вычисление выражения (логика программы), согласно формуле конвертирования и присвоение результата выражения вещественной переменной:



5.7) вывод результата на системную консоль с помощью форматного вывода с использованием инструкции `System.out.printf(...)`:



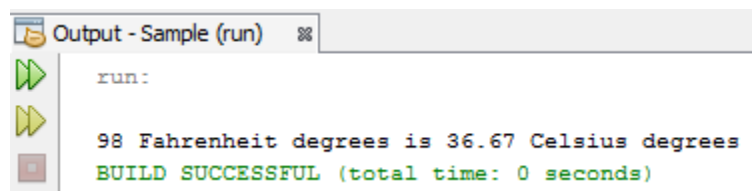
6) Итого, анатомия первой программы следующая:



- 7) Перед написание второго приложения также разработаем вербальное (словесное) описание последовательности действий для достижения результата:
- a) объявление целочисленной переменной для хранения первоначального значения градусов по шкале Фаренгейта;
 - b) объявление вещественной переменной для сохранения результата конвертации градусов по шкале Цельсия в градусы по шкале Фаренгейта;
 - c) инициация целочисленной переменной соответствующим значением;
 - d) выполнение выражения по конвертации градусов согласно указанной формулы и присвоение вещественной переменной
 - e) вывод полученного результата на консоль.
- 8) Исходный класс второй программы представлен ниже:

```
1 public class SecondApplication {  
2  
3     public static void main(String[] args) {  
4         int fahrenheitDegree;  
5         double celsiusDegree;  
6  
7         fahrenheitDegree = 98;  
8  
9         celsiusDegree = 5 / 9.0 * (fahrenheitDegree - 32);  
10  
11        System.out.printf(  
12            "\n%d Fahrenheit degrees is %.2f Celsius degrees\n",  
13            fahrenheitDegree, celsiusDegree);  
14    }  
15 }
```

- 9) Результат работы второй программы:



```
Output - Sample (run) %  
run:  
98 Fahrenheit degrees is 36.67 Celsius degrees  
BUILD SUCCESSFUL (total time: 0 seconds)
```

- 10) Попробуйте самостоятельно разобрать анатомию данной программы. Что можно улучшить в архитектурах приведённых программ?

Контрольные вопросы



1. Какие пользовательские типы данных можно использовать в Java?
2. Сколько классов и с какими модификаторами доступа можно описать в одном исходном файле в Java?
3. Как связано имя исходного файла с именами классов, которые в нём описаны?
4. Что необходимо добавить в класс, чтобы он был запускаемым?
5. Каков синтаксис стартового метода *main(...)*?
6. Что такое литерал (*literal*) и зачем он нужен?
7. Что такое переменная (*variable*) и зачем она нужна?
8. Что такое время жизни и область действия (видимости) переменной?
9. Что такое локальная переменная?
10. Что такое глобальная переменная?
11. К какому типу языка относится Java с точки зрения типизации?
12. Как в Java объявить локальную (глобальную) переменную?
13. Что гласит правило объявления идентификаторов в Java?
14. Что нужно помнить при выборе имени пользовательского идентификатора?
15. Какие системы счисления поддерживаются в Java при работе с числовыми данными?
16. Как JVM работает с вещественными данными?
17. Опишите типы литералов (*целочисленные, вещественные, булевские, символьные и строковые*), доступных в языке Java. Какими способами каждый из них можно отобразить?
18. Какой тип по умолчанию имеют целочисленные и вещественные литералы?
19. Зачем нужны спецификаторы типов и сколько их?
20. Какое новшество было добавлено с JDK 7.0 для удобства чтения чисел? Как правильно его нужно использовать?