Переменные и типы данных

№ урока: 3 **Курс:** Java Starter

Средства обучения: Компьютер с установленной IntelliJ IDEA

Обзор, цель и назначение урока

Рассмотрение понятия переменной, константы и типа данных.
Рассмотрение арифметических операторов и операторов сравнения.

Изучив материал данного занятия, учащийся сможет:

- Применять переменные и константы.
- Понимать, когда и какие типы использовать при создании переменной.
- Выполнять арифметические операции над значениями переменных.
- Сравнивать значения переменных.
- Выполнять форматирование строк.

Содержание урока

1. Рассмотрение примера: Константы.
2. Рассмотрение примера: Преобразование типов (Casting).
3. Рассмотрение примера: Арифметические операторы.
4. Рассмотрение примера: Математические функции.
5. Рассмотрение примера: Инкремент и Декремент.
6. Рассмотрение примера: Операции сравнения.
7. Рассмотрение примера: Присвоение с действием.
8. Рассмотрение примера: Локальные области видимости.
9. Рассмотрение примера: Ключевые слова в качестве идентификаторов.
10. Рассмотрение примера: Конкатенация
11. Рассмотрение примера: Форматирование строк
12. Рассмотрение примера: Сравнение значений разных типов.

Резюме

- **Переменная (`Variable`)** – это именованная область памяти, которая хранит в себе некоторое значение, которое можно изменить.
- **Инициализация переменной** – это первое присвоение ей значения. Все последующие присвоения новых значений этой переменной, не считаются инициализацией.
- Технически, имена переменных могут начинаться со знака «_» – нижнее подчеркивание, знака «$» и с любого алфавитного символа (имена не могут начинаться с цифр и других символов).
- Для именования локальных переменных в Java, рекомендуется использовать соглашение camel Casing. Чтобы выделить слова в идентификаторе, первые буквы каждого слова (кроме первого) сделайте заглавными. Например, `myAge`, `myName`.
- Язык Java чувствительный к регистру (casesensitivity). Например, `MyName` и `myName` – это разные имена.
- Не используйте символы подчеркивания, дефисы и любые другие неалфавитно-цифровые символы для разделения слов в идентификаторе.
- Не используйте венгерскую нотацию. Суть венгерской нотации сводится к тому, что имена идентификаторов предваряются заранее оговорёнными префиксами, состоящими из одного или нескольких символов. Например, `String sClientName; int iSize;`
- Имена переменных должны быть понятны и передавать смысл каждого элемента.
- В редких случаях, если у идентификатора нет точного семантического значения, используйте общие названия. Например, `value`, `item`.
- При создании переменной, используйте название-псевдоним, когда это возможно, а не полное имя типа.

CyberBionic Systematics ® 2016
19 Mariny Raskovoy Str., 5 floor
Kyiv, Ukraine

t. +380 (44) 361-8937
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

- **Константа (Constant)** – это область памяти, которая хранит в себе некоторое значение, которое нельзя изменить.
- Правила использования констант:
  1) Константам необходимо присваивать значение непосредственно в месте создания;
  2) Попытка присвоения константе нового значения приводит к ошибке уровня компиляции;
- Преобразование типа (Casting или Type conversion) – это преобразование значения переменной одного типа в значение другого типа. (Преобразование не следует путать с приведением типов – Cast) Выделяют явное (explicit) и неявное (implicit) преобразование типов.
- Неявное преобразование типа (безопасное) – преобразование меньшего типа в больший или целого типа в вещественный. Является безопасным, так как не происходит потеря точности.
- Явное преобразование типа (опасное) – преобразование большего типа в меньший или вещественного типа в целый. Является опасным, так как происходит потеря точности результата без округления.
- Возможно неявное преобразование значения константы большего типа в меньший, при инициализации переменной значением константы, если значение константы не превышает максимально допустимого значения переменной.
- Возможно явное преобразование значения константы вещественного типа в целый тип при инициализации переменной значением константы, если значение константы не превышает максимально допустимого значения переменной.
- Если значение константы превышает максимально допустимый диапазон значения переменной, такое преобразование возможно с потерей результата (все старшие биты будут отброшены).
- Оператор присвоения (=) сохраняет значение своего правого операнда в месте хранения (переменной) обозначенной в левом операнде. Операнды должны быть одного типа (или правый операнд должен допускать явное преобразование в тип левого операнда).
- Если **после знака присвоения** идет выражение с вычислением или передачей каких-либо значений, то данная операция **выполняется справа-налево**. Для повышения приоритета операции можно использовать круглые скобки ( ).
- **Только четыре операции гарантируют порядок вычислений слева направо: ,, ?:, && и ||**
- Язык Java предоставляет большой набор операторов, которые представляют собой символы, определяющие операции, которые необходимо выполнить с выражением. К операторам, которые выполняют арифметические операции можно отнести операторы:
  +(сложения),
  – (вычитания),
  *(умножения),
  / (деления),
  % (получения остатка от деления)
- Язык Java предоставляет большой набор математических функций для выполнения различных вычислений.
- `Math.sqrt()` – математическая функция которая извлекает квадратный корень. В аргументных скобках указываем значение числа, из которого хотим извлечь квадратный корень.
- `Math.pow()` – возведения числа в степень. В аргументных скобках через запятую указываем два аргумента (первый – число, которое хотим возвести в степень, второй – степень, в которую мы хотим возвести число).
- **Операции умножения, деления, получения остатка от деления имеют больший приоритет, чем сложения и вычитания**, поэтому выполнятся в первую очередь.
- При получении результата остатка от деления – знак результата не сокращается и соответствует значению первого операнда (делимого).
- Если в правой части выражения выполнялись операции деления между целыми числами, то результат будет приведен компилятором к целому типу, даже если результат записать в переменную вещественного типа или привести все выражение к вещественному типу.
- Оператор **инкремента** (++) увеличивает свой операнд на 1. Оператор инкремента может находиться как перед операндом, так и после него: ++variable или variable++.
- **Префиксная операция увеличения** – результатом выполнения этой операции является использование значения операнда после его увеличения.
- **Постфиксная операция увеличения** – результатом выполнения этой операции является использование значения операнда перед его увеличением.

CyberBionic Systematics ® 2016
19 Mariny Raskovoy Str., 5 floor
Kyiv, Ukraine
t. +380 (44) 361-8937
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

- Оператор **декремента** (--) уменьшает свой операнд на 1. Оператор декремента может находиться как перед операндом, так и после него: --variable или variable--.
- Префиксная операция декремента – результатом выполнения этой операции является использования значения операнда после его декремента.
- Постфиксная операция декремента – результатом этой операции является использование значения операнда до его декремента.
- К операциям сравнения можно отнести операции:
  > больше,
  >= больше или равно,
  < меньше,
  <= меньше или равно.
- К операциям проверки на равенство можно отнести операции:
  == равно,
  != не равно.
- Результатом выполнения операций сравнения и проверки на равенство неравенство всегда будет либо `false` или `true`.
- Для предопределенных типов значений оператор равенства **(==)** возвращает значение `true`, если значения его операндов совпадают, в противном случае – значение `false`. Для типа `string` оператор **==** сравнивает значения строк.
- Оператор неравенства **(!=)** возвращает значение `false`, если его операнды равны, в противном случае – значение `true`.
- Оператор сравнения "меньше или равно" **(<=)** возвращает значение `true`, если первый операнд меньше или равен второму, в противном случае возвращается значение `false`.
- Оператор сравнения "меньше" **(<)** возвращает значение `true`, если первый операнд меньше второго, в противном случае возвращается значение `false`.
- Оператор сравнения "больше" **(>)** возвращает значение `true`, если первый операнд больше второго, в противном случае возвращается значение `false`.
- Оператор сравнения "больше или равно" **(>=)** возвращает значение `true`, если первый операнд больше или равен второму, в противном случае возвращается значение `false`.
- Все арифметические операции, производимые над двумя значениями типа (`byte, short`) в качестве результата, возвращают значение типа `int`.
- Для типов `int, long` не происходит преобразования типа результата арифметических операций.
- **Локальная область** – участок кода, внутри класса или блок, который ограничен фигурными скобками.
- **Область видимости переменной** – часть текста программы, в которой имя можно явно использовать. Чаще всего область видимости совпадает с областью действия.
- Переменная созданная внутри локальной области называется **локальной переменной**, область ее действия - от открывающей скобки локальной области до ее окончания (закрывающей скобки) блока, включая все вложенные локальные области.
- Переменная уровня класса называется **глобальной переменной или полем.**
- В коде можно создавать локальные области и в двух разных локальных областях хранить одноименные переменные.
- Если в коде имеются локальные области, то запрещается хранить одноименные переменные за пределами локальных областей. И наоборот, если за пределами локальных областей уже созданы переменные с каким-то именем, то в локальных областях этого уровня запрещается создавать одноименные переменные.
- **Конкатенация – сцепление строк** или значений переменных типа string, для получения строк большего размера с помощью операции +.
- Для форматирования числовых результатов и вывода их на экран можно использовать метод `System.out.print()` или `System.out.println()`, который вызывает метод `String.format()`. Также, можно воспользоваться методом `System.out.printf()`, который выводит строку с заданным форматированием.
- Инструкция форматирования выглядит так:
  %[argument_index$][flags][width][.precision]conversion, где
  - **%** – специальный символ, обозначающий начало конструкции форматирования.

CyberBionic Systematics ® 2016
19 Mariny Raskovoy Str., 5 floor
Kyiv, Ukraine
t. +380 (44) 361-8937
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

- **[argument_index$] –** целое десятичное число, указывающее позицию аргумента в списке аргументов (1$ – первый аргумент из списка, 4$ – четвёртый). Не является обязательной частью конструкции; если позиция не задана, то аргументы будут браться в порядке очередности.
- **[flags] –** специальные флаги форматирования. Не является обязательной частью конструкции.
- **[width] –** положительное целое десятичное число, определяет минимальное количество символов, которые будут выведены. Не является обязательной частью конструкции.
- **[.precision] –** положительное целое десятичное число с точкой перед ним. Используется для ограничения количества символов. Не является обязательной частью конструкции.
- **Conversion –** символ, указывающий как аргумент должен быть отформатирован. Не является обязательной частью конструкции.

Закрепление материала

- Что такое переменная?
- Где и для чего используются переменные?
- Назовите основные типы данных.
- Какие типы данных подходят для хранения значений чисел с плавающей запятой?
- В каком формате должны задаваться значения для строковых переменных?
- Что такое константа?
- В каких случаях используются константы?
- Что такое преобразование значений типов (Casting)?
- Какие существуют правила использования преобразования значений при работе с константами?
- В чем разница явного и неявного преобразования значения типа?
- Что такое конкатенация?
- Что такое инкремент и декремент?
- Какие ограничения применяются к неинициализированным локальным переменным?
- Можно ли использовать в операциях сравнения, два значения разных типов данных?

Дополнительное задание

Задание
Используя IntelliJ IDEA, создайте проект с классом main.
Создайте две целочисленные переменные и выведите на экран результаты всех арифметических операций над этими двумя переменными.

Самостоятельная деятельность учащегося

Задание 1
Имеется 3 переменные типа `int x = 10, y = 12, и z = 3;`
Выполните и рассчитайте результат следующих операций для этих переменных:

```
x += y - x++ * z;
z = --x – y * 5;
y /= x + 5 % z;
z = x++ + y * 5;
x = y - x++ * z;
```

Задание 2
Используя IntelliJ IDEA, создайте класс **ArithmeticAverage** .
Вычислите среднее арифметическое трех целочисленных значений и выведите его на экран.
С какой проблемой вы столкнулись? Какой тип переменных лучше использовать для корректного отображения результата?

CyberBionic Systematics ® 2016
19 Mariny Raskovoy Str., 5 floor
Kyiv, Ukraine
t. +380 (44) 361-8937
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

Задание 3
Используя IntelliJ IDEA, создайте класс **Circle**.
Создайте константу с именем PI (число π «пи»), создайте переменную радиус с именем – r. Используя формулу πR^2, вычислите площадь круга и выведите результат на экран.

Задание 4
Используя IntelliJ IDEA, создайте класс **Volume**.
Напишите программу расчета объема - V и площади поверхности -S цилиндра.
Объем V цилиндра радиусом – R и высотой – h, вычисляется по формуле: $V = \pi R^2 h$
Площадь S поверхности цилиндра вычисляется по формуле: $S = 2\pi R^2 + 2\pi R^2 = 2\pi R(R+h)$
Результаты расчетов выведите на экран.

Задание 5
Используя IntelliJ IDEA, создайте класс **Main**.
Проверьте, можно ли создать переменные со следующими именами:
`uberflu? , _Identifier ,  \u006fIdentifier , &myVar , myVariab1le`

Рекомендуемые ресурсы

Final методы и классы в Java
https://docs.oracle.com/javase/tutorial/java/IandI/final.html

Арифметические операции
https://docs.oracle.com/javase/tutorial/java/nutsandbolts/op1.html

Приведение типов
https://docs.oracle.com/javase/tutorial/java/IandI/subclasses.html

Класс Math
http://docs.oracle.com/javase/7/docs/api/java/lang/Math.html

Операторы сравнения
https://docs.oracle.com/javase/tutorial/java/nutsandbolts/op2.html

Форматирование строк (класс Formatter)
http://docs.oracle.com/javase/7/docs/api/java/util/Formatter.html

Ввод данных с консоли. Класс Scanner
http://docs.oracle.com/javase/7/docs/api/java/util/Scanner.html

CyberBionic systematics
CyberBionic Systematics ® 2016
19 Mariny Raskovoy Str., 5 floor
Kyiv, Ukraine
t. +380 (44) 361-8937
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com