Специализация: данные и функции

Иван Игоревич Овчинников

GeekBrains. Java Core.

2022

Перейдём к интересному: что можно хранить в джаве, как оно там хранится, и как этим манипулировать

В предыдущих сериях

- Краткая история (причины возникновения);
- инструментарий, выбор версии;
- CLI;
- структура проекта;
- документирование;
- некоторые интересные способы сборки проектов.

На прошлом уроке мы коротко поговорили об истории и причинах возникновения языка джава, вскользь посмотрели на инструментарий, который позволит нам писать на джава и получать результат, поверхностно изучилит интерфейс командной строки, научились стремительно создавать довольно симпатичную документацию к своему коду и посмотрели на то как можно автоматизировать ручную работу при компиляции своих проектов.

На этой лекции

Будет рассмотрен базовый функционал языка, то есть основная встроенная функциональность, такая как математические операторы, условия, циклы, бинарные операторы. Далее способы хранения и представления данных в Java, и в конце способы манипуляции данными, то есть функции (в терминах языка называющиеся методами).

Типы, преобразование типов

Хранение данных в Java осуществляется привычным для программиста образом: в переменных и константах, желательно именованных, но об этом позже, для начала поговорим о том, какие вообще бывают языки относительно типов и собственно типы.

Итак, языки программирования бывают типизированными и нетипизированными (бестиповыми). Про нетипизированные языки мы много говорить не будем, они не представляют интереса не только для джава программистов, но и в целом, в современном программировании.

Типы, преобразование типов

рисунок перфокарты

Отсутствие типизации в основном присуще чрезвычайно старым и низкоуровневым языкам программирования, например, Forth и некоторым ассемблерам. Все данные в таких языках считаются цепочками бит произвольной длины и, как следует из названия, не делятся на типы. Работа с ними часто труднее, и при чтении кода не всегда ясно, о каком типе переменной идет речь. При этом часто безтиповые языки работают быстрее типизированных, но описывать с их помощью большие проекты со сложными взаимосвязями довольно утомительно.

Типы, преобразование типов

Java является языком со **строгой** (также можно встретить термин «**сильной**») **явной статической** типизацией.

Что это значит?

- Статическая - у каждой переменной должен быть тип и мы этот тип поменять не можем. Этому свойству противопоставляется динамическая типизация;
- Явная - при создании переменной мы должны ей обязательно присвоить какой-то тип, явно написав это в коде. Бывают языки с неявной типизацией, например, Python;
- Строгая(сильная) - невозможно смешивать разнотипные данные. С другой стороны, существует JavaScript, в котором запись `2 + true` выдаст результат `3`.

Типы, преобразование типов

таблица из методички «Основные типы данных в языке Java»

Все данные в Java делятся на две основные категории: примитивные и ссылочные. Примитивных всего восемь и это, наверное, первое, что спрашивают на джуниорском собеседовании, это байт, шорт, инт, лонг, флоут, дабл, чар и булин. как вы можете заметить в этой таблице, каждый из типов имеет диапазон значений, а значит основное их отличие в объёме занимаемой памяти.

Данные: типы, преобразование типов, константы и переменные (примитивные, ссылочные), бинарное представление, массивы (ссылочная природа массивов, индексация, манипуляция данными);
Базовые функции языка: математические операторы, условия, циклы, бинарные операторы;
Функции: параметры, возвращаемые значения, перегрузка функций;

Антипаттерн «магические числа»

кусок Петренко, спасибо ему за идею

В прошлом примере мы использовали антипаттерн - плохой стиль для написания кода. Число 18 используется в коде коде без пояснений. Такой антипаттерн называется ”магическое число”. Рекомендуется помещать числа в константы, которые храняться в начале файла. ADULT = 18 age = float(input(’Ваш возраст: ’)) how_old = age - ADULT print(how_old, ”лет назад ты стал совершеннолетним”)

Плюсом такого подхода является возможность легко корректировать большие проекты. Представьте, что в вашем коде несколько тысяч строк, а число 18 использовалось несколько десятков раз. При развертывании проекта в стране, где совершеннолетием считается 21 год вы будете перечитывать весь код в поисках магических ”18”и править их на ”21”. В случае с константой изменить число нужно в одном месте. Дополнительный сложности могут возникнуть, если в коде будет 18 как возраст совершеннолетия и 18 как коэффициент для рассчёт чего-либо. Теперь править кода ещё сложнее, ведь возраст изменился, а коэффициент -нет. В случае с сохранением значений в константы мы снова меняем число в одном месте.