Примитивные типы данных

Java Syntax
2 уровень, 2 лекция

ОТКРЫТА

— Привет, бесплатный работник.

— Т.е. я хотел сказать «Здорова, Амиго». Хочу рассказать тебе о внутреннем устройстве переменных. Ты уже знаешь, что у каждой переменной есть область памяти, привязанная к ней, где эта переменная хранит своё значение.

— Ага. Ты рассказывал это в прошлый раз.

— Отлично. Хорошо, что ты это помнишь. Тогда продолжу.

— Все сложные типы состоят из более простых. Те, в свою очередь, из ещё более простых. Пока, наконец, дело не доходит до самых **примитивных**, неделимых типов. Их так и называют – **примитивные типы**. Например, **int** – это один из примитивных типов, а **String** – это уже сложный тип, хранящий свои данные в виде таблицы символов (где каждый символ — это примитивный тип — **char**).

— Очень интересно. Продолжай.

— Сложные типы образуются из простых путём группировки. Такие типы мы называем классами. Когда мы описываем в программе новый класс – это значит, что мы объявляем новый сложный составной тип, данные которого будут или другими сложными типами, или примитивными типами.

Код на Java	Описание
	Объявили новый сложный тип – `Person`. Его данные – это переменная `name` типа

`public class Person`
`{`
`  String name;`
`  int age;`
`}`
	`String` (сложный тип) и переменная `age` типа `int` (примитивный тип)

`public class Rectangle`
`{`
`  int x, y, width, height;`
`}`
	Объявили новый сложный тип – `Rectangle`.
	Он состоит из четырёх переменных примитивного типа – `int`.

`public class Cat`
`{`
`  Person owner;`
`  Rectangle territory;`
`  int age;`
`  String name;`
`}`
	Объявили новый сложный тип – `Cat`. У него есть переменные:
	— `owner`, сложный тип `Person`
	— `territory`, сложный тип `Rectangle`
	— `age`, примитивный тип `int`
	— `name`, сложный тип `String`

— Всё пока ещё понятно, как ни странно.

— Т.к. большие (сложные) типы содержат в себе много маленьких (примитивных), то их объекты занимают много памяти. Больше, чем обычные переменные примитивных типов. Иногда намного больше. Присваивание таких переменных выполнялось очень долго и требовало копирования больших объёмов памяти. Поэтому переменные сложных типов хранят в себе не сам объект, а всего лишь ссылку на него! Т.е. четырёхбайтовый адрес. Этого хватает, чтобы можно было обращаться к данным этих объектов. Всю сложность, связанную с этим, берет на себя Java-машина.

— Ничего не понял.

— Мы уже говорили, что переменная – это как коробка. Если ты хочешь сохранить в ней число 13, то ты можешь написать его на листе и положить в коробку.

— Но представь, что тебе надо сохранить в коробку (переменную) что-нибудь побольше. Например, собаку, машину или твоего соседа Васю. Чтобы не пихать в коробку невпихиваемое, можно поступить проще: вместо собаки взять ее фото, вместо машины – ее номер, вместо Васи – его номер телефона.

— Вот мы берем лист бумаги и пишем на нем телефонный номер Васи. Это и будет аналогом ссылки на объект. Если мы достанем из коробки лист с номером Васи, отксерим его и положим в

несколько коробок, то количество ссылок на Васю увеличится, но Вася как был один, так и остался. Что, в общем-то, логично.

— Особенность такого хранения данных в том, что ссылок может быть много, а объект – один.

— Очень интересно. Почти понял, кстати. Ответь только еще раз: что будет, если я одной переменной сложного типа присвою другую переменную сложного типа?

— Тогда эти две переменные будут содержать одинаковые адреса. И, значит, изменение данных, хранящихся в одной переменой сложного типа, приведёт к изменению данных, хранящихся в другой. **Объект-то**, на который они хранят ссылки, реально **всего один**. А переменных, хранящих на него ссылки, может быть очень много.

— А что хранится в переменных сложных (ссылочных/классовых) типов, пока там ещё нет ссылки на объект? Такое вообще может быть?

— Да, Амиго. Ты опередил меня своим вопросом. Такое может быть. Если в переменной ссылочного (сложного) типа ещё нет ссылки на какой-то объект, то она хранит null – специальную «пустую ссылку». На самом деле, она просто хранит адрес объекта равный 0. Но Java-машина никогда не создаёт объекты с таким адресом, и поэтому всегда знает, что если переменная-ссылка содержит 0, то никакого объекта там нет.

Код на Java	Описание
`String s;` `String s = null;`	Эквивалентные записи.

`Person person;`
`person = new Person();`
`person = null;`
	Создали переменную person, её значение null.
	Занесли в неё адрес новосозданного объекта.
	Присвоили переменной ссылку null.

`Cat cat = new Cat();`
`cat.owner = new Person();`
`cat.owner.name = "God";`
	Создали объект Cat, занесли его ссылку в переменную cat. cat.owner равен null.
	Занесли в cat.owner ссылку на новосозданный объект Person.
	cat.owner.name пока ещё null.
	cat.owner.name присвоили имя – God.

— Я правильно понял? Переменные делятся на два типа: примитивные и ссылочные. Примитивные типы у себя внутри хранят значение, а ссылочные – ссылку на объект.

Примитивные типы – это int, char, boolean и ещё немного, а ссылочные типы – это все остальные, и образуются они с помощью классов.

— Абсолютно верно, мальчик мой.

— Раз ты все понял, вот тебе задачи на закрепление материала.


×3

Задача Java Syntax, 2 уровень, 2 лекция

РЕШЕНА

Прибавка к зарплате

Хорошо быть программистами: у них быстро растут зарплаты. Ну а если этого не происходит, можно повлиять на ситуацию с маленькой помощью друзей-хакеров. Представьте, что вы получили доступ к автоматизированной системе выплаты зарплаты. Вам нужно написать метод-перехватчик, который каждый раз будет прибавлять к зарплате 100 долларов.

Открыть


×1

Задача Java Syntax, 2 уровень, 2 лекция

РЕШЕНА

Считаем длину окружности

Что ж, у нас есть план, давайте его реализовывать. Первым делом реализуем метод, который посчитает нам длину окружности. Для этого в методе нужно прописать формулу, по которой он это

должен делать, и задать параметры. А что же делать с числом Пи, которое, как мы знаем, равно 3.141592… и так далее в бесконечность? Упростим: пускай Пи = 3.14.

Открыть

×1

Задача Java Syntax, 2 уровень, 2 лекция

РЕШЕНА

★★★★½

Кусочек калькулятора

Давайте заставим компьютер считать за нас! В конце-концов, они для того и были созданы. В этой задачке попросим его вычислить сумму и произведение двух чисел. Всё предельно просто: объявляем целочисленные переменные, присваиваем им значение, складываем и перемножаем, а затем — выводим результат на экран.

Открыть

×3

Задача Java Syntax, 2 уровень, 2 лекция

РЕШЕНА

★★★½☆

Откуда берутся Person?

В Java люди берутся оттуда же, откуда и остальные классы: из головы программиста. Важно, чтобы их создатель продумал, что важно для класса, а что нет. В таком случае он будет иметь смысл и сослужит хорошую службу. Итак, начнем. Давайте создадим класс Person, да так, чтобы у этого Person было имя, возраст, вес и… деньги. А потом создадим объект.

Открыть

×7

Задача Java Syntax, 2 уровень, 2 лекция

РЕШЕНА

★★½☆☆

Наш первый конвертер!

Вы наверняка не однократно пользовались электронными конвертерами или программами, которые переводят что-то в одних единицах в нечто в других единицах. Например, доллары в фунты, или километры в мили. Настала пора и нам что-то такое написать. А именно — «переводчик» из градусов Цельсия в градусы Фаренгейта.

Открыть

×3

Задача Java Syntax, 2 уровень, 2 лекция

РЕШЕНА

★★½☆☆

О семейных отношениях

Программист может создать мужчину и женщину парой-тройкой ловких движений своих пальцев. Делов-то: пишем соответствующие классы, создаем объекты. Поработаем над семейной парой: сформируем объекты Man и Woman, затем сохраняем ссылку на Woman в man.wife, а на Man — в woman.husband. Видите, даже загс не нужен.

Открыть