Первые шаги в Java
Пособие по Java-технологиям
(c) AntonSaburov

Введение

Идея данного пособия родилась в результате ведения курсов по началам Java. Не думаю, что это будет что-то выдающееся в огромном мире книг по Java. Скорее всего это будет просто очередная книга. Но она будет написана мной, а значит содержать мой опыт и мое видение процесса обучения. И в ней будет то, о чем я хотел бы рассказать начинающим свой путь в программировании. Буду тешить себя мыслью, что мне доступно тайное знание как именно надо обучать программированию 🙂
ВНИМАНИЕ !!! Я очень хочу, чтобы вы при чтении делали упражнения вместе с пособием. Это очень важный момент. Может поэтому и хорошо, что я делаю это в электронном виде — все-таки вы ближе к компьютеру.

Что значит программировать

Давайте сначала постараемся понять, что значит «программировать». Если не вдаваться в сложные теоретические изыскания, то на самом деле программист в процессе написания программы делает две вещи:

- Исключения
- Решения на основе классов
- Список контактов — начало
- Коллекции — базовые принципы
- Коллекции — продолжение
- Список контактов — GUI приложение
- Что такое JAR-файлы
- Многопоточность — первые шаги
- Многопоточность и синхронизация
- Работаем с XML
- Reflection — основы
- Установка СУБД PostgreSQL
- Базы данных на Java — первые шаги
- Возможности JDBC — второй этап
- JDBC — групповые операции
- Список контактов — работаем с БД
- Переезжаем на Maven
- Потоки ввода-вывода
- Сетевое взаимодействие
- С чего начинается Web

1. Описывает задачу на обычном (русском, английском, китайском и т.д.) языке
2. Переводит описание задачи на язык программирования (Java, C, C++, Pascal, C# и т.д.)

Описывать задачу на обычном языке — это на самом деле достаточно сложный процесс. Потому что задачи могут описываться очень сложно. Для многих задач вычислительного характера придуман язык математики — формулы. Интегралы, дифференциалы, тригонометрические формулы и прочая. Т.к. изначально именно математика требовала создания компьютеров, то многие считают, что для программирования необходимо очень хорошо знать математику. Причем не просто математику, а высшую — с полным набором дисциплин вроде математического анализа, аналитической геометрии и пр. Автор не относится к этим людям. В большом количестве проектов, в которых я принимал участие, знание математики сводилось к знанию арифметики. Другой вопрос — надо было уметь описывать задачу с большой степенью детализации. Вот этот навык надо укреплять и усиливать в процессе постоянной практики.
Что же касается второго пункта — тут надо просто научится переводить человеческий язык в машинный. Значит надо владеть человеческим языком (что на самом деле ОЧЕНЬ ВАЖНО) и языком программирования (что тоже важно, но для начинающего программиста не так сильно).
Необходимо учесть, что человеческий язык гораздо богаче, чем язык программирования. Это значит, что не все действия, которые легко описываются с помощью человеческого языка, можно также легко описать с помощью языка программирования.
Например, представим себе, что у нас есть робот на колесах, который умеет двигаться вперед и разворачиваться на месте. Если у него еще есть ролик с краской,, который поднимается/опускается по команде, то его можно запрограммировать рисовать определенные фигуры на плоской поверхности.
Но сразу возникает вопрос — а насколько язык робота может отличаться от нашего ? Может не сильно — например, ему можно сказать «вперед на 10 метров» и он это выполнит. Или можно будет сказать — «повернуть направо на 45 градусов».
Но может быть и гораздо сложнее — робот умеет только включать/выключать двигатель. И для разворота надо включить режим типа «колеса крутятся в разные стороны». Для движения вперед режим надо включить «колеса крутятся в одну сторону».
Если вы пошли дальше и поняли, что расстоянием придется управлять либо по времени работы двигателя, либо считать количество оборотов — при выборе профессии программиста вы явно на правильном пути. Если же вы придумали еще какой-либо способ — то мне явно стоит пообщаться с Вами лично.

Программа на человеческом языке

Давайте попробуем написать программу для нашего робота. Для этого сначала определим (абстрактно конечно), что наш робот понимает следующие команды:

1. Опустить ролик (с краской)
2. Поднять ролик (с краской)

3. Вперед (на какое-то количество метров)
4. Налево (на какое-то количество градусов)
5. Направо (на какое-то количество градусов)

Итак, теперь пишем программу для рисования квадрата со стороной в 2 метра. Допустим, что робот уже находится в том месте, где надо рисовать квадрат. Многие из вас наверно уже догадались. Будет нечто вроде:

1. Опустить ролик
2. Вперед (на 2 метра)
3. Налево (на 90 градусов)
4. Вперед (на 2 метра)
5. Налево (на 90 градусов)
6. Вперед (на 2 метра)
7. Налево (на 90 градусов)
8. Вперед (на 2 метра)
9. Налево (на 90 градусов)
10. Поднять ролик

Давайте внимательно посмотрим на эти замечательные команды. В них явно наблюдается повторение пары команд. Это «Вперед (на 2 метра)» и «Налево (на 90 градусов)». Модифицируем нашу программу:

1. Опустить ролик
2. Сделать 4 раза следующие действия:
    1. Вперед (на 2 метра)
    2. Налево (на 90 градусов)
3. Поднять ролик

Вы можете удивиться, но в нашей несложной программе мы использовали 3 из 4 типов команд. Вот они:

1. Команда без аргумента
2. Команда с аргументом
3. Выполнение команды по условию
4. Циклическое выполнение команды

В наше программе у нас есть:

1. Команда без аргумента — Опустит ролик, Поднять ролик
2. Команда с аргументом — Вперед (на 2 метра), Налево (на 90 градусов)
3. Циклическое выполнение команды — Сделать 4 раза следующие действия

Для демонстрации типа «Выполнение команды по условию» мы можем расширить возможности нашего робота для реализации более сложного алгоритма. Например, у робота можно спросить — хватит ли ему краски на рисование линии определенной длины. Робот вернет в этом случае ответ «ДА» или «НЕТ». А также добавим включение звукового сигнала. Тем самым робот привлечет к себе внимание. Мы его заправим и он продолжит программу дальше.
Теперь наш робот умеет делать следующее:

1. Опустить ролик (с краской)
2. Поднять ролик (с краской)
3. Вперед (на какое-то количество метров)
4. Налево (на какое-то количество градусов)
5. Направо (на какое-то количество градусов)
6. Хватит краски (на какое-то количество метров) — «ДА»/»НЕТ»
7. Подать звуковой сигнал

И наша программа превращается в следующий набор шагов:

1. Опустить ролик
2. Сделать 4 раза следующие действия:
    - Проверить краску на 2 метра
    - Если ответ «НЕТ»
        1. подать звуковой сигнал
    - Иначе
        1. Вперед (на 2 метра)
        2. Налево (на 90 градусов)
3. Поднять ролик

Программа на языке программирования

Я бы очень хотел прямо сейчас вам показать программу на языке программирования, но не смогу. Потому как языка для общения с роботом у нас с вами нет. При изучении языка программирования Java мы с вами попробуем создать такого робота. Правда он у нас получится тоже немного абстрактным. Но не будем отчаиваться.
Как я уже говорил выше, еще одной задачей программиста является перевод описания выполнения поставленной задачи с человеческого описания на язык программирования. Чем мы с вами и займемся. Но перед этим нам необходимо прояснить еще несколько вопросов.

Запуск программ

Если вы работали за компьютером, то вы скорее всего слышали о таком понятии как файл. Вся информация хранится в этих файлах. По сути файл — это некоторая область на жестком диске, которая является логическим целым. Именно логическим — ваш файл может быть разбросан по нескольким секциям жесткого диска (или иного носителя, например флешки или компакт-диска). Специальная программа — операционная система — управляет файлами на диске. А человек (и другие программы) видят файл как одно целое. И могут его копировать, переносить и даже удалять.
Так вот программа — это просто файл с набором команд, которые могут быть выполнены процессором. Запускает программу обычно операционная система — она считывает программы из файла, загружает в память и запускает.

Программа: шаги по созданию и запуску

Каждая программа при своем создании проходит простой цикл — написание текста, компиляция, запуск и проверка работоспособности.
Большинство программ — это обычный текст, который можно набирать в том же приложении «Блокнот» в Windows (или еще каком-либо простом текстовом редакторе). В большие программы конечно же подключаются картинки, видео, звук и прочие ресурсы, но тем не менее основным элементом программ является обычный текст, который описывает порядок выполнения вашей программы. Шаг за шагом. Какие шаги вы можете делать — об этом мы поговорим несколько позже. А пока давайте остановимся на следующем шаге — это компиляция.
Компиляция — это перевод текста программы в набор кодов, которые понятны процессору. Вы наверно не будете удивлены тем, что программы, которые исполняются на CPU явно не текстовые. Они представляю из себя набор байтов (думаю, что о байтах вы что-то слышали) состоящие из нулей и единиц (и о двоичном коде возможно тоже что-то слышали). Каждая комбинация является командой или данными, которые обрабатывает CPU. Так вот много лет назад программы для компьютеров писались именно так — в память забивали набор байтов, состоящих из нулей и единиц и стартовали программу. Внешний вид такого чуда мысли программиста был мягко говоря слабо читаем. Хотя люди с опытом могли творить просто чудеса. Тем не менее такое кодирование имело еще одну проблему — надо было все делать самому. Где-то в памяти хранить данные и где-то хранить команды. Их надо было разделять, и ими надо было управлять И все это делалось вручную — т.е. программист был с одной стороны почти богом (мог делать все, что хочет), с другой стороны он был вынужден делать титаническую работу по разбиению программы на мелкие-мелкие детали. Как частенько говорят — «из-за деревьев леса не видно». Написать программу обработки текста — это была проблема. В итоге инженеры пришли к нормальной идее, что можно создать так называемые алгоритмические языки программирования. Идея достаточно простая — некоторые конструкции при написании алгоритмов являются стандартными и их на самом деле не так уж и много — мы это еще увидим. Конструкции стандартные — значит и перевод их с понятного представления в машинное тоже будет стандартным. Т.е. можно написать программу, которая будет «переводить» текст на алгоритмическом языке в машинный. И уже полученный результат можно исполнять на CPU. Правда тут возникла еще одна особенность — процесс компиляции может выдавать ошибки, если программа содержит некорректные с точки зрения компилятора конструкции. Но алгоритмические языки резко увеличили скорость программирования. Компилятор сам распределяет память для данных, команд — это огромное подспорье.
С этого и началась эра алгоритмических языков — люди писали программы в более-менее понятной форме, в виде текста. Компилятор их преобразовывал в машинный код, который можно было исполнять.
Стоит отметить. что также существуют программы-интерпретаторы. В отличии от компилируемых языков, которые на выходе получают сразу готовый машинный код, интерпретаторы получают на вход текст программы и пытаются ее выполнить «на лету». Т.е. каждая строка или команда сразу преобразуется в машинный код и выполняется. Это удобно — не надо компилировать — но с другой стороны это замедляет работу программы. Для каждой команды надо выполнить преобразование в машинный код, а потом выполнить этот машинный код. Долго, но зато удобно.

Java Virtual Machine — виртуальная машина Java

Java — это тоже компилируемый язык программирования. И как у других подобных языков, у него тоже есть стадии написания текста, компиляции и запуска. Но в отличии от языков типа C, Pascal и прочих, программа на Java не сразу компилируется в машинный код. После компиляции создается так называемый байт-код. И этот байт-код выполняется специальной программой — Java Virtual Machine, JVM (виртуальная машина Java). Зачем это надо ? Идея очень проста и логична. Каждый тип процессора имеет свой набор команд, что означает, что один и тот же текст на алгоритмическом языке будет преобразовываться в разные машинные коды в зависимости от того, под какой процессор вы компилируете вашу программу. Да и сам компилятор тоже должен существовать под конкретный процессор. И даже если вы сами написали нужный компилятор — надо иметь под рукой исходный текст. А его вам могут и не дать. Он денег стоит.
Из всего этого следует, что компилировать программу под каждый процессор, и даже под каждую операционную систему отдельно требует больших усилий. В компании Sun Microsystems пошли по следующему пути: решили, что программы на Java будут выполняться под управлением специальной программы, по сути на виртуальном процессоре, который имеет совершенно определенный набор команд. И каждая программа на Java может быть выполнена по сути на любом процессоре, на любой операционной системе под которую написана JVM. Таким образом один раз скомпилированная программа на Java будет по идее выполняться где угодно. Девиз Sun был именно таким: Write once run anywhere — написано однажды, запускается везде.
В итоге программа на Java тоже проходит стадию написания текста, компиляции и запуска. Но запускается она под управлением JVM, которая интерпретирует байт-код. вы вполне законно можете подумать, что раз «интерпретирует» — значит работает медленнее. Да, в некоторых случаях это так. Но и тут разработчики Sun смогли найти решение. Был изобретен JIT-компилятор (JIT — Just In Time). Этот механизм позволяет преобразовать байт-код в машинный код при первом проходе программы и в дальнейшем выполнять просто готовый машинный код. Т.е. появилась еще одна стадия — компиляция байт-кода в машинный код. Это позволяет существенно ускорить работу программы. Скорость работы программы на Java вполне сопоставима со скоростью работы программы на том же C или Pascal. Так что страшные истории про жуткие тормоза программ на Java — это только истории и больше ничего. Хотя относительно графических программ можно сказать, что Java уступает в быстродействии.

Программа на Java — шаги по запуску программы

Кроме того, что программу надо написать, программу надо еще суметь запустить. Так что программисты постоянно делают один и тот же цикл из четырех шагов:

1. Редактировать код программы
2. Собрать/скомпилировать программу из текста в код, понятный JVM — байт-код. Опять же — надо исправлять ошибки компилятора. Тогда возвращаемся к первому шагу.
3. Если все скомпилировалось — запустить программу под управлением JVM
4. Посмотреть корректность работы программы. Если даже программа запустилась — не факт, что вы совершенно верно реализовали алгоритм. Нередко надо опять идти к первому шагу.

Если на каком-то шаге произошла ошибка, то повторяем все с самого начала. Если все хорошо, то цикл можно завершать.

Установка Java Development Kit (JDK)

JDK — это набор программ для разработки и запуска программ на Java. Для нас самыми главными будут две программы:

1. javac — компилятор файлов с программами, написанными на java. У таких файлов обычно расширение .java. Например First.java. После успешной компиляции появляется файл с расширением .class. Т.е. если мы скомпилируем First.java, то должен появиться файл First.class. По большому секрету файлов .class может быть несколько, но мы этот вопрос разберем, .когда придет время.
2. java — по сути это программа, которая и является JVM. Именно она исполняет байт-код. который содержится в файле .class

В нашем курсе мы будем рассматривать работу под операционной системой Windows. Если вы предпочтете использовать иную систему, то Вам придется потрудиться — мы не рассматриваем как работать на Java под другими ОС. Установка JDK не является сложной задачей — если у вас будет несколько иная версия, чем мы рассматриваем, то вряд ли вы увидите очень большую разницу.
Давайте посмотрим как это делается — Установка JDK.

Leave a reply

Comment

You may use these HTML tags and attributes: `<a href="" title=""> <abbr title=""> <acronym title=""> <b> <blockquote cite=""> <cite> <code class="" title="" data-url=""> <del datetime=""> <em> <i> <q cite=""> <s> <strike> <strong> <pre class="" title="" data-url=""> <span class="" title="" data-url="">`

Имя *

E-mail *

Сайт

− четыре = один

Add comment