Глава 2. Краткий обзор языка Java.

Объектно-ориентированное программирование. Две парадигмы

Все компьютерные программы состоят из двух элементов: **кода и данных**. Кроме того, программа может быть **концептуально организована вокруг своего кода или своих данных**. Иными словами, одни программы пишутся исходя из того, "что происходит”; а другие - исходя из того, "что затронуто”.

Существуют две парадигмы, определяющие то, как строится программа:

1. Моделью, ориентированной на процессы – последовательность линейных шагов (т.е. кода), код, воздействующий на данные. При использовании этого подхода начинают возникать проблемы.
2. Объектно-ориентированное программирование. Позволяет организовать программу вокруг ее данных (т.е. объектов) и набора четко определенных интерфейсов к таким данным. Это данные, управляющие доступом к коду

Абстракция

Абстракция – принцип, который позволяет скрыть детали реализации объекта и выделить только самую важную информацию или функциональность.

Скажем, в состав аудиосистемы может входить радиоприемник, проигрыватель компакт-дисков и/или проигрыватель МРЗ. Суть в том, что вы управляете сложностью автомобиля (или любой другой сложной системы) за счет использования **иерархических абстракций.**

**Суть ООП**: Последовательность шагов процесса может стать совокупностью сообщений, передаваемых между этими объектами. Таким образом, каждый объект описывает свое уникальное поведение. Вы можете воспринимать такие объекты как конкретные сущности, которые реагируют на сообщения, указывающие им о необходимости делать что-то.

Три принципа ООП

1. Инкапсуляция

Инкапсуляция представляет собой механизм, который связывает вместе код и обрабатываемые им данные, а также защищает их от внешнего вмешательства и неправильного использования. Инкапсуляцию можно считать защитной оболочкой, которая предотвращает произвольный доступ к коду и данным из другого кода, определенного вне оболочки. Доступ к коду и данным, находящимся внутри оболочки, строго контролируется через четко определенный интерфейс.

**Сила инкапсулированного кода** в том, что каждый знает, как пол учить к нему доступ, и потому может использовать его независимо от деталей реализации и без каких-либо опасений столкнуться с неожиданными побочными эффектами.

**Основой инкапсуляции** в Java является класс.

Класс определяет структуру и поведение (данные и код), которые будут общими для набора объектов. Каждый объект заданного класса содержит структуру и поведение, определенные классом, как если бы он был "отлит" в форме класса. Поэтому объект – экземпляр класса.

Таким образом, класс представляет собой *логическую конструкцию*, а объект имеет *физическую реальность*.

В правильно написанных программах на Java методы определяют способ использования переменных-членов, т.е. поведение и интерфейс класса определяются методами, которые работают с данными его экземпляра.

**Целью класса** является инкапсуляция сложности.

1. Наследование

Наследование представляет собой процесс, посредством которого один объект приобретает свойства другого объекта. Оно важно, т.к. поддерживает концепцию иерархической классификации.

Суперкласс — это класс, который предоставляет общие свойства и методы, которые могут быть унаследованы подклассами. Суперкласс определяет базовую структуру и поведение, которые могут быть общими для нескольких классов. Это позволяет избегать дублирования кода и обеспечивает более легкое управление и сопровождение кода.

Наследование также взаимодействует с инкапсуляцией. Когда подкласс наследует суперкласс, он наследует не только его методы и свойства, но также и инкапсулированную структуру и данные.

1. Полиморфизм (от греческого "много форм")

Представляет собой средство, которое позволяет использовать один интерфейс для общего класса действий.

В языке, не являющемся объектно-ориентированным, вам придется создать три разных набора стековых процедур с отличающимися именами. Но благодаря полиморфизму в Java вы можете указать общий набор стековых процедур с одинаковыми имен.

Ввод кода программы

Файл с исходным кодом в Java официально называется *единицей компиляции.*

По соглашению имя главного класса должно совпадать с именем файла, содержащего программу.

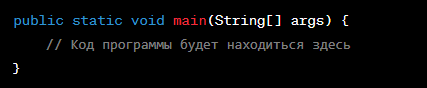
Компиляция программы

В Java исходный код программы обычно разбивается на несколько классов, каждый из которых может быть размещен в отдельном файле. Каждый файл с исходным кодом Java обычно имеет расширение .java. Однако при компиляции этого исходного кода каждый класс превращается в отдельный файл байт-кода с расширением .class. Этот байт-код является исполняемым для Java виртуальной машины (JVM).

Для того чтобы JVM могла найти и выполнить нужный класс при запуске программы, следует придерживаться следующей конвенции:

1. Имя файла с исходным кодом (.java) должно совпадать с именем класса, который он содержит.
2. Когда вы запускаете программу Java, вы указываете имя класса, который должен быть запущен. JVM автоматически ищет соответствующий файл с расширением .class.

Подробный анализ первого примера программы



Данная строка начинает метод main(). Как объяснялось в предыдущем комментарии, с этой строки программа начнет выполняться. Обычно программа на Java начинает выполнение с вызова main(). Полностью осознать смысл каждой части строки пока невозможно, т.к. для этого нужно хорошо понимать подход Java к инкапсуляции.

Ключевое слово puЫic представляет собой модификатор доступа, который позволяет программисту управлять видимостью чл енов класса. Когда член класса предварен ключевым словом puЫic, доступ к нему может быть получен из кода за пределами класса, где он объявлен. В данном случае метод main() должен быть объявлен как puЫic, потому что при запуске программы его потребуется вызывать в коде за пределами класса.

Ключевое слово static позволяет вызывать main() без создания конкретного экземпляра класса. Причина в том, что main() вызывается машиной JVM до создания каких-либо объектов, т.к. метод main принадлежит не конкретному экземпляру класса, а самому классу.

Ключевое слово void просто сообщает компилятору, что main() не возвращает значение.

Параметры метода – args, который получает любые аргументы командной строки, присутствующие при выполнении программы.



System — это класс в стандартной библиотеке Java, который предоставляет доступ к системным ресурсам и переменным окружения.

out — это объект типа PrintStream, который представляет стандартный вывод (обычно консоль). Он предоставляет методы для вывода данных.

println — это метод объекта PrintStream, который используется для вывода текста или значения и добавляет перевод строки (новую строку) в конце вывода. Это означает, что следующий вывод будет на новой строке.

Если за место текста указать переменную, то на самом деле ее значение сначала преобразуется из целочисленного (или другой тип данных) в эквивалентное строковое и затем объединяется с предшествующей ему строкой.

Доп.

Операции инкремента и декремента — это арифметические операции, которые увеличивают (инкремент) или уменьшают (декремент) значение переменной на 1.

Блок кода создан, то становится логической единицей, которую можно применять в любом месте, где разрешено использовать одиночный оператор. Главная причина их существования - создание логически неразрывных единиц кода.

В Java литералы — это константные значения, которые представляют данные определенных типов.