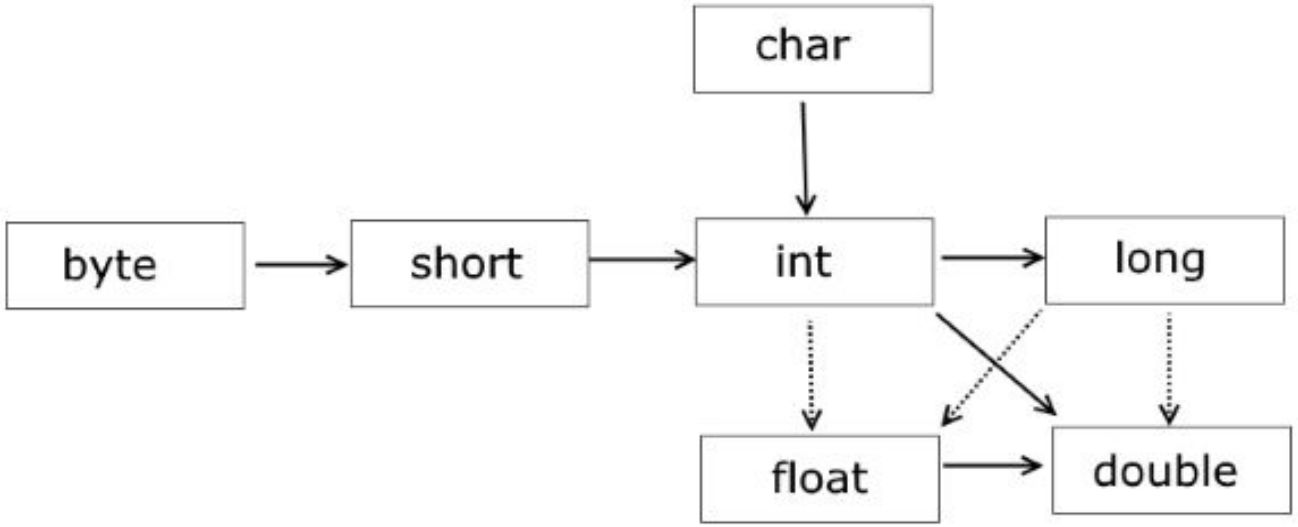
Преобразование типов



Стрелками на рисунке показано, какие преобразования типов могут выполняться автоматически. Пунктирными стрелками показаны автоматичекие преобразования с потерей точности.

```
double d = Double.parseDouble(str);
Double e = Double.valueOf(str);
```

Операторы. Операторы управления.

Арифметические операторы

- В большинстве языков программирования, команды языка называются операторами. Java имеет операторы для выполнения основных арифметических операций:
- + - Сложение - - Вычитание * - Умножение / - Деление При использовании арифметические операторы записываются аналогично обычным математическим выражениям:
- 2 + 2 3 * 15 3 - 5 * 4
- Отдельно следует упомянуть оператор % — остаток от целочисленного деления. В результате выполнения этого оператора получается остаток, например, 7%5 вернет остаток равный 2.

Операторы сравнения

- Кроме математических операторов, в Java имеются операторы сравнения, позволяющие сравнить два значения, и получить результат в виде логического значения т.е. "истина" или "ложь". Оператор == позволяет сравнить два значения на равенство.
- x == 10 x == y y * 2 == x - 10
- Иногда, требуется, сравнивать не на равенство, а на неравенство. В таких случаях используется оператор не равно !=
- x != 10
- Кроме сравнения на равенство, существуют также операторы меньше < и больше >. Результатом оператора больше будет истина, если значение слева больше значения справа, а у оператора меньше - наоборот.
- x > 10 и x < y

Логические операторы

- оператором "логическое и". Данный оператор записывается, как && и его результат является истинным, только если оба выражения слева и справа от него являются истинными. Мы получаем запись:
- 1 < x && x < 5
- Результат выражения будет истиной, если и первое и второе условие будут истинными. Если хотя бы одно из условий, является ложным, выражение будет ложным. Существует также оператор "логическое или", который записывается как две вертикальные черты ||. Результат этого оператора будет истинным, если хотя бы одно из выражений справа и слева от него будет истинным. Ложным он будет, только если оба выражения ложны. Например, нам надо проверить, что значение переменной не попадает в промежуток. Это можно записать следующим образом:
- 5 < x || x < 1
- То есть значение не попадает в промежуток, если оно меньше 1 или больше 5.

Оператор присваивания

- Может состоять из одного значения, переменной, или быть более сложным с использованием арифметических или логических операторов. Слева в таком операторе всегда должна быть переменная, в которую заносится результат. Т. е. всегда сначала вычисляется правая часть оператора, а затем результат заносится в переменную, находящуюся в левой части. Примеры операторов присваивания:
- x = y * 2 / z; x = 3; x = y;
- Также допустимо использовать переменную, стоящую слева в правой части выражения. Например, чтобы увеличить значение x в два раза следует записать:
- x = x * 2;
- В этом случае в правом выражении используется старое значение переменной, а потом в нее же заносится результат. Существует сокращенные формы записи подобных операторов. Они выглядят как += -= *= /= %=. Например две следующие строчки идентичны по выполняемым действиям:
- x += 2 ; x = x + 2;

Инкремент и декремент

- Операторы инкремента и декремента это унарные операторы, которые увеличивают или уменьшают на единицу значение операнда при использовании инкремента и декремента соответственно
- Различают префиксную и постфиксную форму для обоих операторов. Префиксная форма сначала изменяет значение операнда на 1 и использует новое значение для дальнейших вычислений. Постфиксная форма сначала использует старое значение операнда, а затем изменяет его на 1
- Префиксный оператор инкремента:
- int i = 1;
- int m = 2 * ++i; // результат m = 4
- Постфиксный оператор инкремента:
- int i = 1;
- int m = 2 * i++; // результат m = 2
- Префиксный оператор декремента:
- int i = 2;
- int m = 2 * --i; // результат m = 2
- Постфиксный оператор декремента:
- int i = 2;
- int m = 2 * i--; // результат m = 4

Приведение типов

- Следует помнить, что если в операторе присваивания результат справа относится к одному типу данных, а слева — к другому, компилятор может не позволить выполнить такое присваивание, и выдать ошибку. Например, если слева булевская переменная, а справа целое число. И это касается не только принципиально разных типов. В переменную int нельзя просто присвоить значение long.
- В случае, если переменные сходных типов, например хранят числа, это можно обойти с помощью т.н. приведения типов. Например:
- int x; double z =10.5 x =(int)z;
- Для приведения типов перед присваиваемым выражением надо поставить нужный нам тип в круглых скобках (часто остальное выражение тоже приходится брать в скобки из-за приоритетов). В этом случае присваивание выполнится, и в x попадет значение 10. При этом часть данных может теряться, как в нашем примере при преобразовании из дробного числа в целое всегда отбрасывается дробная часть.

Операторы работы со строками

- Оператор + может применяться не только к числам, но и к строкам. При сложении двух строк происходит их склеивание. Например:
- String str = "Hello "; String newStr = str + "world!";
- В результате в переменной newStr будет находиться значение "Hello world!". Следует помнить, что при сложении двух слов оператор не вставляет между ними пробела, поэтому пробел должен быть предусмотрен в одной из строк, либо прибавлен отдельной строкой:
- String str = "Hello"; String newStr = str + " " + "world!";
- При сложении строк один из аргументов может быть и не строкой. В этом случае он автоматически преобразуется в строку:
- int x = 15; String str = "Значение x = " + x;

Функции

- Для выполнения некоторых часто выполняемых действий в языках программирования используются функции. Функции могут создаваться самим программистом, а могут быть стандартными, созданными авторами языка. Каждая функция имеет свое собственное имя. Имена функций задаются по тем же правилам, что и имена переменных. Чтобы функция выполнила свои действия, ее надо вызвать. Для этого следует написать ее имя и круглые скобки. Круглые скобки, которые идут после имени означают, что мы имеем дело именно с функцией, а не с переменной. Чаще всего функция выполняет действия не просто так, а используя некоторые исходные данные. Например, функция синус вычисляет значение синуса угла, используя в качестве исходных данных значение угла. Исходные данные, их также называют входными параметрами, пишутся внутри круглых скобок. Например:
- Math
- В данном случае функция вычисляет синус угла равного пяти. Java позволяет не только использовать уже существующие функции, но и создавать пользователю собственные.

Уровни доступа

- Каждый объект не должен выставлять наружу все свои параметры для изменения просто так. Например, если у нашего робота есть координаты X и Y, то не вызывает сомнений факт, что их нельзя менять прямо. Робот должен поменять свои координаты в результате передвижения. Нельзя обратиться к переменной X внутри объекта Robot и сделать самое простое присваивание. Это будет как минимум нелогично. Лучше такого вообще не позволять. Т.е. мы таким образом должны создавать описание класса, чтобы нельзя было просто так получать доступ к его внутренним переменным.
- Всего в Java есть четыре модификатора доступа. Перечислим их в порядке от самых строгих до самых «мягких»: **private;**
- **protected;**
- **default (package visible);**
- **public.**

private

- Собственно, ограничение доступа к полям и реализация геттеров-сеттеров — самый распространенный пример использования private в реальной работе.
- То есть реализация инкапсуляции в программе — главное предназначение этого модификатора.
- public class Cat {
- private String name;
- public Cat(String name) {
- this.name = name;
- }
- public Cat() {
- }
- public String getName() {
- return name;
- }
- public void setName(String name) {
- this.name = name;
- }
- }

public

- Части кода, помеченные модификатором public, предназначаются для конечного пользователя.
- Если привести пример из жизни, private — это все процессы, происходящие внутри телевизора, когда он работает, а public — это кнопки на пульте телевизора, с помощью которых пользователь может им управлять. При этом ему не нужно знать как устроен телевизор и за счет чего он работает. Пульт — это набор public-методов: on(), off(), nextChannel(), previousChannel(), increaseVolume(), decreaseVolume() и т.д.

default

- Дальше у нас по списку идет модификатор default или, как его еще называют, package visible. Он не обозначается ключевым словом, поскольку установлен в Java по умолчанию для всех полей и методов.

- Если написать в твоем коде —
- int x = 10
- … у переменной x будет этот самый package visible доступ.

- Запомнить, что он делает, легко. По сути, default = protected-наследование :)

- Случаи его применения ограничены, как и у модификатора protected. Чаще всего default-доступ используется в пакете, где есть какие-то классы-утилиты, не реализующие функциональность всех остальных классов в этом пакете.

protected

- Поля и методы, обозначенные модификатором доступа protected, будут видны:
- в пределах всех классов, находящихся в том же пакете, что и наш;
- в пределах всех классов-наследников нашего класса.

часто используется в абстрактном классе.

Метод main

- Для того, чтобы программа успешно запустилась, один из классов в ней должен иметь функцию main. Заголовок метода должен быть именно таким, какой показан в данном примере. Этот метод запускается, когда вы запускаете вашу программу. Поэтому его часто называют "точка входа" Т.к. метод main является точкой входа, при запуске программы следует всегда указывать именно тот класс, в котором он объявлен, в нашем случае java HelloWorld. Как правило, в программе присутствует один класс с методом main. В простых программах иногда достаточно только одного класса и одного этого метода. Все необходимые операции в таком случае должны быть выполнены внутри main.

Операторы управления

- Общий вид оператора if Для того, чтобы выполнять в разных случаях разные действия в языке Java используется специальный оператор if. Он выполняет разные операторы в зависимости от указанного ему логического значения. Общий вид оператора if:
- if (Условие) Оператор1; else Оператор2;
- Если Условие истинно, то будет выполняться Оператор1, если Условие ложно, будет выполняться Оператор2. В качестве условия может использоваться любое выражение, результатом которого является логическое значение. Это может быть логическое выражение, например:
- if (x == 3)
- Также это может быть функция, возвращающая логическое значение:
- if(str.equal("test"))
- Условие может также состоять из одной переменной, если это переменная типа boolean, например
- boolean b = x == 3; if(b)
- Конструкция else Оператор2; может отсутствовать и тогда оператор if примет вид:
- if (Условие) Оператор1;

Пример

- Например, нам следует проверить, является ли значение в переменной x четным и сообщить об этом пользователю.
- if (x % 2 == 0) {
- System.out.println("Число "+ x + " является четным");
- } else {
- System.out.println("Число "+ x + " является нечетным");
- }

Составной оператор

- Составной оператор представляет собой несколько операторов, заключенных в фигурные скобки { }, например:
- {
  - f = a + b ;
  - c = f * a ;
  - m = f * c;
- }
- Составной оператор может включать в себя любые операторы и вызовы функций, в том числе и другие составные операторы:
- {
  - f = a + b;
  - {
    x = f * a;
    m = f * c;
  - }
  - c = f * x * m;
- }

Вложенные операторы IF

- В качестве оператора, выполняемого по условию, в операторе if может использоваться другой оператор if . В таком случае говорят о вложенных условных операторах. Такая конструкция имеет вид:
- if (Условие1)
  - if(Условие2)
    - Оператор1;
- else
  - Оператор2;

  else
  - Оператор3;

- Избежать возможных затруднений при анализе данной конструкции позволяет простое правило: else относится к ближайшему свободному if.

Пример

```
if(a > b) {
	if(a > c) {
		System.out.println("максимальное число:" + a);
	} else {
		System.out.println("максимальное число:" + c);
	}
} else {
	if(b > c) {
		System.out.println("максимальное число:" + b);
	} else {
		System.out.println("максимальное число:" + c);
	}
}
```

Конструкция if else if

Очень часто встречаются задачи, в которых выбор следует сделать между более чем двумя возможными вариантами, каждому из которых соответствует свое условие. Тогда применяется конструкция if else if. Она имеет вид:

```
if (условие1) {
     оператор1;
} else if (условие2) {
    оператор2;
} else if (условие3) {
    оператор3;
}
```

Оператор выбора switch

- Если необходимо делать выбор из конкретных значений, можно использовать не конструкцию if else if, а специальный оператор switch. Его общий вид:
- switch (выражение) {
  case значение1:
    операторы
    break;
  case значение2:
    операторы2
    break;
  default:
    Операторы
- }
- В скобках switch должно стоять выражение, результат которого далее будет сравниваться. После case ставится значение, и если результат выражения совпал с этим значением, выполняются операторы после двоеточия и до break. Если ни одно из предложенных значений не совпало с результатом выражения, выполняются операторы после default.