Типы данных

Примитивные Типы данных

1. **boolean** - представляет логические значения true и false
2. **byte** - 8-разрядное целое число
3. **char** - символ
4. **double** - числовое значение с плавающей точкой двойной точности
5. **float** - числовое значение с плавающей точкой одинарной точности
6. **int** - целое число
7. **long** - длинное целое число
8. **short** - короткое число

**Значения целых типов являются целыми числами в следующих диапазонах:**

* **Byte** - от -128 до 127
* **Short** - от -32768 до 32767
* **int** - от -2147483648 до 2147483647
* **long** - от -9223372036854775808 до 9223372036854775807
* **char** - от '\u0000' до '\uffff', то есть от 0 до 65535

**Язык программирования Java предоставляет ряд операторов, которые воздействуют на целочисленные значения:**

**Операторы сравнения, результатом которых является значение типа boolean:**

* Операторы числового сравнения <, <=, > и >=
* Операторы числового равенства == и !=

**Числовые операторы, результатом которых является значение типа int или long:**

* Унарные операторы "плюс" и "минус" + и -
* Мультипликативные операторы \*, / и %
* Аддитивные операторы + и -
* Оператор увеличения ++, как префиксный, так и постфиксный
* Оператор уменьшения --, как префиксный, так и постфиксный
* Операторы сдвига со знаком и без знака <<, >> и >>>
* Оператор побитового дополнения ~
* Целочисленные побитовые операторы &, ^ и |

Условный оператор ? :

Оператор приведения, который может преобразовывать целочисленное значение в значение любого указанного числового типа.

Оператор конкатенации строк, который при задании String операнда и целого операнда преобразует целый операнд в String, представляющий его значение в десятичной форме, а затем создает вновь созданное String значение, представляющее собой объединение двух строк.

**Язык программирования Java предоставляет ряд операторов, которые воздействуют на значения с плавающей запятой:**

**Операторы сравнения, результатом которых является значение типа boolean:**

* Операторы числового сравнения <, <=, > и >=
* Операторы числового равенства == и !=

**Числовые операторы, результатом которых является значение типа float или double:**

* Унарные операторы "плюс" и "минус" + и -
* Мультипликативные операторы \*, / и %
* Аддитивные операторы + и -
* Оператор увеличения ++, как префиксный, так и постфиксный
* Оператор уменьшения --, как префиксный, так и постфиксный

Условный оператор ? :

Оператор приведения, который может преобразовывать значение с плавающей запятой в значение любого указанного числового типа

Оператор конкатенации строк, который при задании String операнда и операнда с плавающей запятой преобразует операнд с плавающей запятой в String, представляющий его значение в десятичной форме (без потери информации), а затем создает вновь созданное String значение путем объединения двух строк

**Логическими операторами являются:**

* Операторы отношения == и !=
* Оператор логического дополнения !
* Логические операторы &, ^ и |
* Операторы условного-и и условного-или && и ||

Условный оператор ? :

Оператор конкатенации строк, который при задании String операнда и boolean операнда преобразует boolean операнд в String (либо "true" или "false"), а затем создает вновь созданное String, представляющее собой объединение двух строк

**Логические выражения определяют поток управления в нескольких видах инструкций:**

1. if
2. while
3. do
4. for

Ссылочные типы данных

1. типы классов
2. типы интерфейсов
3. переменные типа
4. типы массивов

**Объект** — это экземпляр класса. Ссылочные значения являются указателями на эти объекты или специальной нулевой ссылкой, которая не ссылается ни на один объект.

**Массив** — это набор похожих типов данных в Java. Он может быть любого типа.

**Операторами для ссылок на объекты являются:**

* Доступ к полю с использованием либо точного имени (§6.6), либо выражения доступа к полю
* Вызов метода
* Оператор приведения
* Оператор конкатенации строк, который при задании String операнда и ссылки преобразует ссылку в String путем вызова toString метода ссылочного объекта (используя "null", если либо ссылка, либо результат toString является нулевой ссылкой), а затем создает вновь созданное String, представляющее собой объединение двух строк
* instanceof Оператор
* Операторы равенства ссылок == и !=
* Условный оператор ? :

**Класс Object** - является суперклассом всех других классов.

**Все типы классов и массивов наследуют методы класса Object, которые вкратце представлены следующим образом:**

* Метод clone используется для создания дубликата объекта.
* Метод equals определяет понятие равенства объектов, которое основано на сравнении значений, а не ссылок.
* Метод finalize запускается непосредственно перед уничтожением объекта.
* Метод getClass возвращает Class объект, который представляет класс объекта.
* Для каждого ссылочного типа существует Class объект. Его можно использовать, например, для обнаружения полного имени класса, его членов, его непосредственного суперкласса и любых интерфейсов, которые он реализует.
* Типом выражения вызова метода getClass является Class<? extends |T|>, где T - класс или интерфейс, по которому был произведен поиск getClass, а |T| обозначает удаление T.
* Метод класса, который объявлен synchronized, синхронизируется на мониторе, связанном с Class объектом класса.
* Этот метод hashCode очень полезен, вместе с методом equals, в хэш-таблицах, таких как java.util.HashMap.
* Методы wait, notify и notifyAll используются в параллельном программировании с использованием потоков.
* Метод toString возвращает String представление объекта.

**Экземпляры класса String** - представляют последовательности символов. String Объект имеет постоянное (неизменяемое) значение. Строковые литералы и текстовые блоки являются ссылками на экземпляры класса String. Оператор конкатенации строк неявно создает новый String объект, если результат не является постоянным выражением.

**Переменная типа** — это неквалифицированный идентификатор, используемый в качестве типа в телах классов, интерфейсов, методов и конструкторов. Переменная типа вводится путем объявления параметра типа универсального класса, интерфейса, метода или конструктора. Переменная типа

**Параметризованные типы — это** класс или тип интерфейса формы C<T1,...,Tn>, где C является именем универсального класса или интерфейса, и <T1,...,Tn> представляет собой список аргументов типа, которые обозначают определенный параметризация универсального класса или интерфейса.

**Параметризованный тип C<T1,...,Tn> является правильно сформированным, если, верно, все из следующего:**

* C — это имя универсального класса или интерфейса.
* Количество аргументов типа совпадает с количеством параметров типа в универсальном объявлении C.
* При преобразовании захвата, приводящем к типу C<X1,...,Xn>, каждый аргумент типа Xi является подтипом S[F1:=X1,...,Fn:=Xn] для каждого связанного типа S в Bi.

Аргументы типа могут быть либо ссылочными типами, либо подстановочными знаками. Подстановочные знаки полезны в ситуациях, когда требуется только частичное знание параметра типа.