Лекция 2

Примитивные типы данных

Преобразование типов

Неявное преобразование типов

- Преобразование целочисленных типов в более емкие
  (byte □ short □ int □ long)
- Преобразование char в int и long
- Преобразование целочисленных типов в типы с плавающей точкой (возможна потеря точности)

Явное приведение типов

- Оператор приведения типа: (typename)
- При приведении более емкого целого типа к менее емкому старшие биты просто отбрасываются
- При приведении типа с плавающей точкой к целому типу дробная часть отбрасывается (никакого округления)
- Слишком большое дробное число при приведении к целому типу превращается в MAX_VALUE или MIN_VALUE
- Слишком большой double при приведении к float превращается в Float.POSITIVE_INFINITY или Float.NEGATIVE_INFINITY

Автоматическое расширение

- При вычислении выражения (a@b) аргументы a и b преобразовываются в числа, имеющие одинаковый тип:
  - если одно из чисел double, то в double;
  - если одно из чисел float, то в float;
  - если одно из чисел long, то в long;
  - иначе оба числа преобразуются в int

- Следствие: Арифметическое выражение над byte, short или char имеет тип int, поэтому для присвоения результата обратно в byte, short или char понадобится явное приведение типа

Неявное приведение с потерей данных

- Сокращенная запись var @= exval

  раскрывается в var = (typename) (var @ (exval))

  int i += 4 на самом деле i = (int) (i + (4))

- Неявно срабатывает приведение типа, в том числе с потерей данных

- java.lang.Math

- java.math.BigInteger
  java.math.BigDecimal

java.math.BigInteger

- Введен для того, чтобы было можно производить целочисленные вычисления с любой разрядностью

- Чтобы использовать класс **BigInteger** в своей программе его необходимо импортировать командой

  import java.math.BigInteger;

- Метод **compareTo()** сравнивает два объекта **BigInteger** и возвращает результат сравнения в виде чисел −1, 0 или 1

Конструкторы

- **BigInteger(String value)** — объект будет хранить большое целое число, заданное строкой цифр, перед которыми может стоять знак минус;

- **BigInteger(String value, int radix)** — задается строка цифр со знаком **value**, записанная в системе счисления с основанием **radix**;

  и т.д. (всего их 6)

- Статический метод **valueOf()** с целочисленным литералом или целочисленной переменной, переданной методу как аргумент, преобразовывает обычное число в число с произвольной точностью.

- Три константы — **ZERO, ONE** и **TEN** — моделируют нуль, единицу и число десять в операциях с объектами класса **BigInteger**.

- **toByteArray()** преобразует объект в массив байтов.

Основные операции и функции

- abs() — возвращает объект, содержащий абсолютное значение числа, хранящегося в данном объекте this;
- add(x) — операция сложения this + x;
- divide(x) — операция деления this / x;
- gcd(x) — наибольший общий делитель абсолютных значений объекта this и аргумента x;
- max(x) — наибольшее из значений объекта this и аргумента x;
- min(x) — наименьшее из значений объекта this и аргумента x;
- mod(x) — остаток от деления объекта this на аргумент x;
- subtract(x) — операция вычитания this — x;
- multiply(x) — операция умножения this * x;

и др.

java.math.BigDecimal

- Каждый объект этого класса хранит два целочисленных значения: мантиссу вещественного числа в виде объекта класса **BigInteger** и неотрицательный десятичный порядок числа типа **int**. Мантисса может содержать любое количество цифр, а порядок ограничен значением константы **Integer.MAX_VALUE.**
- При работе со значениями **BigDecimal** можно явно указать нужную точность (то есть количество десятичных разрядов). Кроме того, при выполнении операции, отбрасывающей разряды (например, при делении), требуется указать тип округления, которому подвергается первый разряд слева от отбрасываемых разрядов.
- Три константы — **ZERO, ONE** и **TEN** — моделируют вещественные нуль, единицу и вещественное число десять

Округление

- Результат операции над объектами класса **BigDecimal** округляется по одному из восьми правил, определяемых следующими статическими целыми константами:
- ROUND_CEILING — округление в большую сторону;
- ROUND_DOWN — округление в меньшую сторону по модулю;
- ROUND_UP — округление в большую сторону по модулю
- ROUND_FLOOR — округление в меньшую сторону;
- ROUND_HALF_DOWN — Округление вниз, если число после запятой > .5;
- ROUND_HALF_EVEN — Округление половины по чётности ;
- ROUND_HALF_UP — Округление вверх, если число после запятой >= .5;
- ROUND_UNNECESSARY — предполагается, что результат будет целым, и округление не понадобится;

Основные операции и функции

- abs() — абсолютное значение объекта this;
- add(x) — операция сложения this + x;
- divide(x, r) — операция деления this / x с округлением по способу r;
- divide(x, n, r) — операция деления this / x с изменением порядка и округлением по способу r;
- max(x) — наибольшее из this и x;
- min(x) — наименьшее из this и x;
- multiply(x) — операция умножения this * x;
- subtract(x) — операция вычитания this — x;

и др.