Рубцова Татьяна Павловна.

1. Знакомство с Java.Простые типы данных и функции
2. Базовые типы в Java. Ветвление в Java. Quality Assurance
3. Ввод/вывод в консоль
4. Объектно-ориентированное программирование. Основные принципы. Чистый код
5. Структуры выбора, циклы, массивы
6. Input/Output. Исключения в Java
7. Стандартная библиотека Java
8. Инструменты проверки кода (Junit, Mockito, Cucumber)
9. Java Collections Framework
10. Ввод/вывод, исключения, дженерики
11. Spring
12. Java Database Connectivity
13. Функциональное программирование
14. HTML. CSS. JavaScript

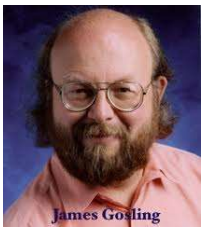
Git !!!

История языка

Java изначально была разработана для интерактивного телевидения, но в то время она была слишком продвинутой технологией для индустрии цифрового кабельного телевидения. История Java начинается с Зеленой команды.

Члены команды Java (также известные как Green Team) инициировали этот проект по разработке языка для цифровых устройств, таких как телевизионные приставки, телевизоры и т. Д. Тем не менее, он лучше всего подходил для интернет-программирования. Позже технология **Java** была включена в Netscape.

Принципы создания Программирования на **Java** были такими: *«Простой, надежный, портативный, платформонезависимый, защищенный, высокопроизводительный, многопоточный, архитектурно-нейтральный, объектно-ориентированный, интерпретируемый и динамический»*. **Java была разработана** Джеймсом Гослингом, который известен как отец **Java**, в 1995 году. Джеймс Гослинг и члены его команды начали проект в начале 90-х годов.



В настоящее время Java используется в интернет-программировании, мобильных устройствах, играх, решениях для электронного бизнеса и т. Д. Ниже приведены важные моменты, описывающие историю Java.

1) Джеймс Гослинг, Майк Шеридан и Патрик Нотон инициировали проект языка Java в июне 1991 года. Небольшая команда солнечных инженеров называется Green Team.

2) Первоначально он был разработан для небольших, встроенных систем в электронных приборах, таких как телевизионные приставки.

3) Во-первых, он назывался «Greentalk» Джеймсом Гослингом, а расширение файла было .gt.

4) После этого он получил название Oak и был разработан в рамках проекта Green.

Почему Ява была названа «Дубом»? История Java от дуба до Явы

5) Почему дуб? Дуб является символом силы и выбран в качестве национального дерева многих стран, таких как США, Франции, Германии, Румынии и т. Д.

6) В 1995 году Oak был переименован в «Java», потому что он уже был торговой маркой Oak Technologies.

Почему Java-программирование называется "Java"?

7) Почему они выбрали название Java для языка Java? Команда собралась, чтобы выбрать новое имя. Предложенными словами были «динамичный», «революционный», «шелковый», «толчок», «ДНК» и т.д. Они хотели что-то, что отражало бы суть технологии: революционное, динамичное, живое, крутое, уникальное, легко пишущееся и веселое.

По словам Джеймса Гослинга, «Java была одним из лучших вариантов наряду с Silk». Поскольку Java была настолько уникальной, большинство членов команды предпочли Java другим именам.

8) Ява - это остров в Индонезии, где был произведен первый кофе (называемый кофе Java). Это своего рода эспрессо. Название Java было выбрано Джеймсом Гослингом, когда он пил кофе рядом со своим офисом.

9) Обратите внимание, что Java - это просто имя, а не аббревиатура.

10) Первоначально разработан Джеймсом Гослингом в Sun Microsystems (которая в настоящее время является дочерней компанией Oracle Corporation) и выпущен в 1995 году.

11) В 1995 году журнал Time назвал Java одним из десяти лучших продуктов 1995 года.

12) JDK 1.0 был выпущен 23 января 1996 года. После первого выпуска Java в язык было добавлено много дополнительных функций. В настоящее время Java используется в приложениях Windows, веб-приложениях, корпоративных приложениях, мобильных приложениях, картах и т. Д. Каждая новая версия добавляет новые возможности в Java.

История версий Java

Многие версии Java были выпущены до сих пор. Текущая стабильная версия Java — Java SE 10.

JDK Альфа и Бета (1995)

JDK 1.0 (23 января 1996)

JDK 1.1 (19 февраля 1997)

J2SE 1.2 (8 декабря 1998)

J2SE 1.3 (8 мая 2000)

J2SE 1.4 (6 февраля 2002)

J2SE 5.0 (30 сентября 2004)

Java SE 6 (11 декабря 2006 г.)

Java SE 7 (28 июля 2011 г.)

Java SE 8 (18 марта 2014 г.)

Java SE 9 (21 сентября 2017 г.)

Java SE 10 (20 марта 2018 г.)

Java SE 11 (сентябрь 2018 г.)

Java SE 12 (март 2019 г.)

Java SE 13 (сентябрь 2019 г.)

Java SE 14 (март 2020 г.)

Java SE 15 (сентябрь 2020 г.)

Java SE 16 (март 2021)

Java SE 17 (сентябрь 2021 г.)

Java SE 18 (март 2022 года)

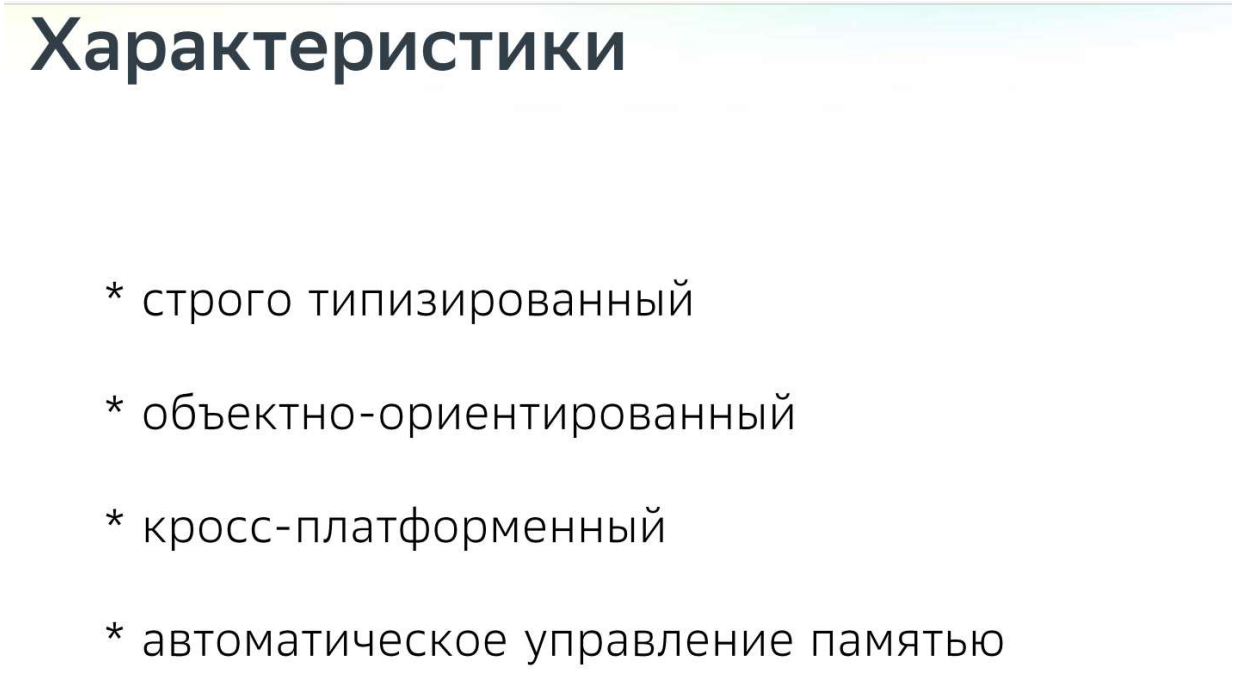
С момента выпуска Java SE 8 корпорация Oracle следует шаблону, в котором каждая четная версия выпускается в марте месяце, а нечетная версия выпускается в сентябре.

https://www.javatpoint.com/java-versions (https://www.javatpoint.com/java-versions)

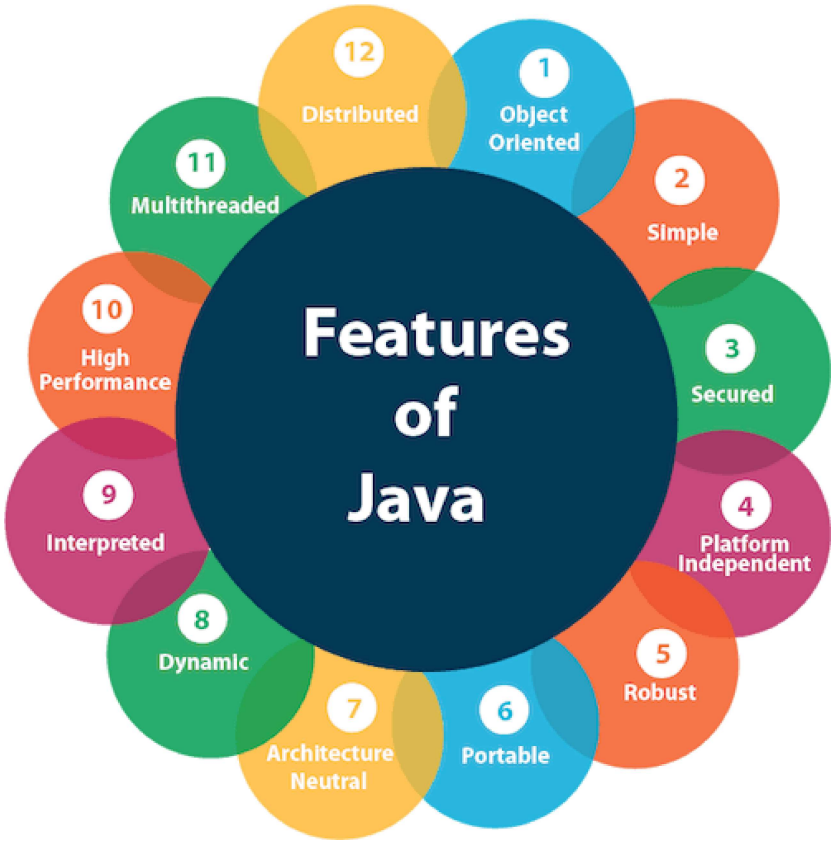
https://www.javatpoint.com/features-of-java (https://www.javatpoint.com/features-of-java)

Основная цель Программирование на Java

Создание языка должно было сделать его портативным, простым и безопасным языком программирования. Помимо этого, есть также некоторые отличные функции, которые играют важную роль в популярности этого языка. Особенности Java также известны как модные слова Java.



- Простой
- Объектно-ориентированный
- Портативный
- Независимость от платформы
- Обеспеченных
- Надежный
- Нейтральная архитектура
- Интерпретировать
- Высокая производительность
- Многопоточных
- Распределённый
- Динамический



Простой

Java очень прост в освоении, а его синтаксис прост, чист и понятен. Согласно Sun Microsystem, язык Java является простым языком программирования, потому что:

Синтаксис Java основан на C++ (поэтому программистам легче освоить его после C++). Java удалила многие сложные и редко используемые функции, например, явные указатели, перегрузку операторов и т.д.

Нет необходимости удалять объекты без ссылок, так как в Java существует автоматическая сборка мусора.

Объектно-ориентированный

Java - это объектно-ориентированный язык программирования. Все в Java является объектом.

Объектно-ориентированный означает, что мы организуем наше программное обеспечение как комбинацию различных типов объектов, которые включают в себя как данные, так и поведение.

```
Объектно-ориентированное программирование (ООП) — это методология, которая
упрощает разработку и сопровождение программного обеспечения, предоставляя
некоторые правила.

Основными понятиями ООП являются:
```

- Объект
- Класс
- Наследство
- Полиморфизм
- Абстракция
- Инкапсуляция

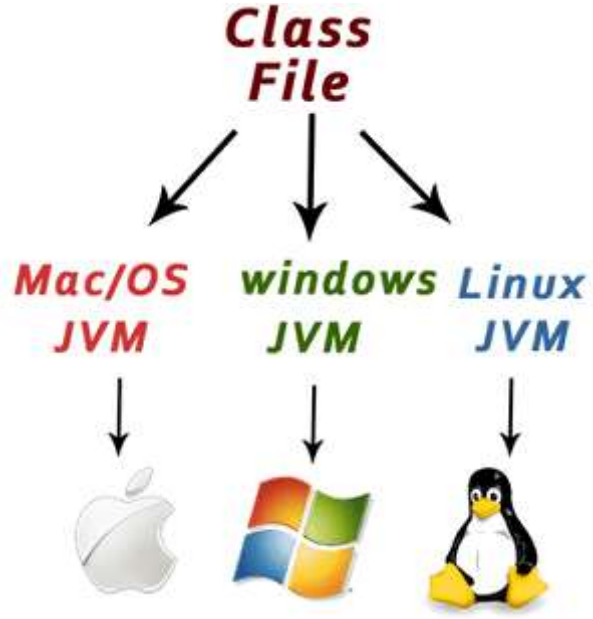
Базовые принципы ООП

1. инкапсуляция (сокрытие реализации, предоставление методов для работы с объектом)

2. наследование (создание объекта -потомка, который наследует структуру базового класса)

3. полиморфизм (один интерфейс — множество реализаций)

Кросс платформенность



Java не зависит от платформы, потому что она отличается от других языков, таких как C , C++ , и т. Д., Которые компилируются в конкретные платформы машины, в то время как Java является записью один раз, запустите любой язык. Платформа — это аппаратная или программная среда, в которой выполняется программа.

Существует два типа платформ на программной и аппаратной основе. Java предоставляет программную платформу.

Платформа Java отличается от большинства других платформ в том смысле, что это программная платформа, которая работает поверх других аппаратных платформ. Он состоит из двух компонентов:

Среда выполнения API (интерфейс прикладного программирования) Java-код может быть выполнен на нескольких платформах, например, Windows, Linux, Sun Solaris, Mac/OS и т.д. Код Java компилируется компилятором и преобразуется в байт-код. Этот байт-код является платформонезависимым кодом, поскольку он может быть запущен на нескольких платформах, т. E. Write Once и Run Anywhere (WORA).

Надежный и безопасный

Java обеспечивает надежность, поскольку может обрабатывать исключения, ошибки времени выполнения и т. д. Говорят, что это надежный язык из-за мощной функции управления памятью. И он считается безопасным языком программирования, поскольку не поддерживает концепцию указателей. Более того, JVM играет очень важную роль с точки зрения безопасности, поскольку гарантирует, что никакая небезопасная программа не будет запущена.

Java защищена, потому что:

- Нет явного указателя
- Программы Java, выполняемые в изолированной программной среде виртуальной машины

- Загрузчик классов: Загрузчик классов в Java является частью среды выполнения Java (JRE), которая используется для динамической загрузки классов Java в виртуальную машину Java. Он добавляет безопасность, отделяя пакет для классов локальной файловой системы от тех, которые импортируются из сетевых источников.
- Верификатор байт-кода: Он проверяет фрагменты кода на наличие нелегального кода, который может нарушать права доступа к объектам.
- Менеджер по безопасности: Он определяет, к каким ресурсам класс может получить доступ, например к чтению и записи на локальный диск.
- Язык Java предоставляет эти ценные бумаги по умолчанию. Некоторая безопасность также может быть обеспечена разработчиком приложения явно через SSL, JAAS, криптографию и т. д.
- Загрузчик классов: Загрузчик классов в Java является частью среды выполнения Java (JRE), которая используется для динамической загрузки классов Java в виртуальную машину Java. Он добавляет безопасность, отделяя пакет для классов локальной файловой системы от тех, которые импортируются из сетевых источников.
- Верификатор байт-кода: Он проверяет фрагменты кода на наличие нелегального кода, который может нарушать права доступа к объектам.
- Менеджер по безопасности: Он определяет, к каким ресурсам класс может получить доступ, например к чтению и записи на локальный диск

Garbage collector

В Java реализовано автоматическое управление памятью.

Специальный процесс, называемый сборщиком мусора, периодически освобождает память, удаляя объекты, которые не будут востребованы приложением.

Независимый от платформы и портативный

Java-программа не зависит от платформы, что означает, что после компиляции Java-программа может работать в любой операционной системе, такой как Linux, Windows или macOS. И, следовательно, это увеличивает переносимость Java.

Независимая от платформы функция связана с компилируемой и интерпретируемой функцией. Java реализует функцию независимости от платформы благодаря байтовому коду. Байт-код интерпретируется с помощью виртуальной машины Java (JVM), после чего он может работать на любой машине, такой как Windows, Linux и т. д.

Архитектурно-нейтральным

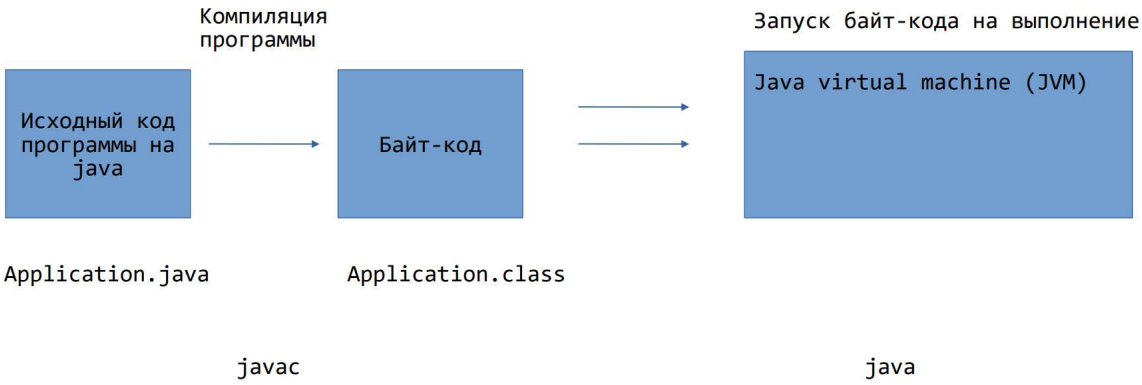
Компилятор генерирует архитектурно-нейтральные объекты формата файла, что делает скомпилированный код исполняемым на многих процессорах, с наличием системе Java Runtime.

Java не зависит от архитектуры, потому что нет зависимых от реализации функций, например, размер примитивных типов фиксирован.

В программировании на C тип данных int занимает 2 байта памяти для 32-битной архитектуры и 4 байта памяти для 64-битной архитектуры. Тем не менее, он занимает 4 байта памяти как для 32,так и для 64-разрядных архитектур в Java.

Скомпилировано и интерпретировано

Для преобразования кода с языка высокого уровня на язык низкого уровня большинство языков программирования использовали либо компилятор, либо интерпретатор, в то время как Java использовал как компилятор, так и интерпретатор. Приведенная ниже диаграмма продемонстрирует лучшее понимание этой концепции:



Java-программа выполняется в два этапа. На первом этапе компилятор компилирует код Java и генерирует байт-код, а на втором этапе интерпретатор преобразует байт-код в машинный код с помощью JVM. Таким образом, Java использовал силу и устойчивость как компилируемых, так и интерпретируемых языков.

Высокая производительность

Java быстрее, чем другие традиционные интерпретируемые языки программирования, потому что байт-код Java «близок» к машинному коду. Он все еще немного медленнее, чем скомпилированный язык (например, C ++). Java является интерпретируемым языком, поэтому он медленнее, чем компилируемые языки, например, C, C ++ и т. д.

Распределённый

Java распространяется, потому что она облегчает пользователям создание распределенных приложений на Java. RMI и EJB используются для создания распределенных приложений. Эта функция Java позволяет нам получать доступ к файлам, вызывая методы с любой машины в Интернете.

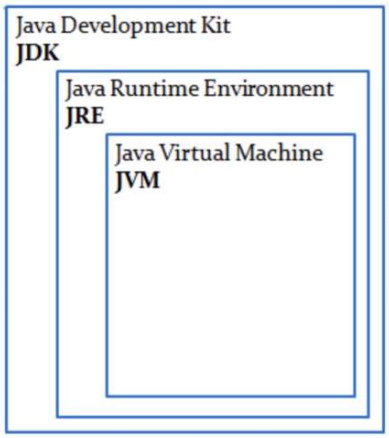
Многопоточный

Поток подобен отдельной программе, выполняемой одновременно. Мы можем писать Java-программы, которые справляются со многими задачами одновременно, определяя несколько потоков. Основным преимуществом многопоточности является то, что она не занимает память для каждого потока. Он имеет общую область памяти. Потоки важны для мультимедиа, веб-приложений и т.д .

Динамический

Java является динамичным языком. Поддерживает динамическую загрузку классов. Это означает, что классы загружаются по требованию. Он также поддерживает функции из своих родных языков, то есть C и C ++.

JVM. JRE. JDK



- JVM (Java Virtual Machine) - виртуальная машина Java - основная часть исполняющей системы Java, так называемой Java Runtime Environment (JRE). Виртуальная машина Java исполняет байт-код Java, предварительно созданный из исходного текста компилятором Java (javac). JVM обеспечивает платформо-независимый способ выполнения кода. Программисты могут писать код не задумываясь, как и где он будет выполняться.
- JRE (Java Runtime Environment) - минимальная реализация виртуальной машины, необходимая для исполнения Java приложений, без компилятора и других средств разработки. Состоит из виртуальной машины и библиотек Java классов.
- JDK (Java Development Kit) - комплект разработчика приложений на языке Java, включающий в себя компилятор, стандартные библиотеки классов Java, примеры, документацию, различные утилиты и исполнительную систему JRE.

JVM – Java Virtual Machine

- Виртуальная машина Java — это программа, целью которой является выполнение других программ.

Две основные функции

- позволяет запускать Java-программы на любом устройстве.
- управлять и оптимизировать программную память.

Базовые типы в Java

Базовые примитивные типы

Размеры примитивных типов

Приведение типов

ООП обертки над примитивными типами

Ограничения примитивных числовых типов, BigDecimal, BigInteger

String - как особый тип в Java

Типы данных

- Java - типизированный язык, в котором каждая переменная уже известна во время компиляции.
- Как только переменная объявлена определенным типом данных, она не может содержать значения других типов данных.

- Примитивный тип данных:
  например -> boolean, char, int, short, byte, long, float и double

- Непримитивный тип данных или объектный тип данных:
  например -> строка, массив, Объект

Базовые примитивные типы

Восемь примитивных типов данных:
byte, short, int, long, char, float, double и boolean

- Целые числа: byte, short, int, long
- Числа с плавающей точкой: float, double
- Символы: char
- Логические значения: boolean

Логические выражения

- Литералы: false, true
- Результат любого сравнения — boolean: >, <, ==
- Нет преобразования между boolean и другими примитивными типами

Символьные значения

- char — 16 бит, беззнаковый (0 .. 2^16 – 1)
- Представляет номер символа в кодировке Unicode
- Литералы:
  - символ в одинарных кавычках: 'a'
  - шестнадцатеричный код символа: '\u78bc'
  - спецпоследовательности: '\t', '\n', '\r', '\"', '\\'

Свободно конвертируется в числовые типы и обратно

Целые числа

Имя	Размер (bit)	Диапазон значений
byte	8	от -128 до 127
short	16	от -32 768 до 32 767
int	32	от -2 147 483 648 до 2 147 483 647
long	64	от -9223372036854775808 до 9223372036854775807

- Размер фиксирован, одинаков для всех платформ
- Все типы знаковые, беззнаковых вариантов нет

Целые числа: литералы

- Десятичное число: 123
- Восьмеричное число: 0123
- Шестнадцатеричное число: 0x123
- Двоичное число: 0b101 (с Java 7)
- С подчеркиванием: 123_456_789 (с Java 7)
- С суффиксом L для long: 123L

Целые числа: литералы

- Десятичное число: 123
- Восьмеричное число: 0123
- Шестнадцатеричное число: 0x123
- Двоичное число: 0b101 (с Java 7)
- С подчеркиванием: 123_456_789 (с Java 7)
- С суффиксом L для long: 123L

Численные операции

- сложение +, +=
- вычитание -, -=
- умножение *, *=
- деление /, /=
- остаток %, %=
- инкремент ++
- декремент --

Вещественные числа

Имя	Размер (bit)	Диапазон значений
float	32	от -1.4e-45 до 3.4e+38
double	64	от -4.9e-324 до 1.7e+308

Вещественные числа: литералы

- Обычная запись: -1.234
- Экспоненциальная запись: -123.4e-2 (−123.4 ⋅ 10−2)
- Шестнадцатеричная запись: 0xFFFF (FFFF)
- С суффиксом типа:

  38f

  3e19d

  123.4e-2f

Приведение типов: неявное

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

Приведение типов: явное

- 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 превращается в MAX_VALUE или MIN_VALUE
- Слишком большой double при приведении к float превращается в Float.POSITIVE_INFINITY или Float.NEGATIVE_INFINITY

Приведение типов: явное

При вычислении выражения (a @ b) аргументы a и b преобразовываются в числа, имеющие одинаковый тип:

- если одно из чисел double, то в double;
- иначе, если одно из чисел float, то в float;
- иначе, если одно из чисел long, то в long;
- иначе оба числа преобразуются в int.

```
byte a = 1;
byte b = 1;
byte c = a + b; // compilation error
byte d = b + 1; // compilation error
```

ООП обертки над примитивными типами

Тип	Класс - обёртка
byte	Byte
short	Short
int	Integer
long	Long
char	Character
float	Float
double	Double
boolean	Boolean

- Autoboxing: примитивное значение → объект-обертка
- Autounboxing: объект-обертка → примитивное значение

Ограничения примитивных числовых типов: BigInteger, BigDecimal

Если мы говорим о целых числах, наиболее вместительным типом данных является long, а если речь идет о числах с плавающей точкой – double.

Но что если нужное нам число настолько велико, что не влезает даже в long?

Диапазон возможных значений Long довольно велик, но всё-таки ограничен определенным размером - 64 бита.

Что нам придумать, если наше ОЧЕНЬ БОЛЬШОЕ ЧИСЛО весит 100 бит?

К счастью, ничего придумывать не нужно. В Java для таких случаев были созданы два специальных класса – BigInteger (для целых чисел) и BigDecimal (для чисел с плавающей точкой).

Теоретически у них нет ограниченииия.

String – как особый тип в Java

- Создание экземпляра
- Конкатенация
- Неизменяемость
- String Pool
- StringBuilder / StringBuffer

Функции (методы) в java