Оглавление

[1. История создания Java 2](#_Toc127693430)

[2. Синтаксис 3](#_Toc127693431)

# 1. История создания Java

1 этап:

Название: дуб

Основа: С++ - синтаксис

Применение: софт бытовой техники

2 этап: (переориентировали)

Название: java

Основа: smallTalk – первый ООП язык

Применение:

- высоконагруженные системы (много данных в режиме реального времени)

- обработка сложного набора данных

- ИИ

- сетевые приложения

Из-за большого количества сфер использования существовали разные версии Джавы:

1) Java SE – Standart Edition

Тут все бесплатно. Сейчас существует только она

2) Java EE –

Она была платной. Использовалась для больших проектов крупных компаний. Например, если бы вы хотели написать elma с нуля, то использовали бы эту версию джаву.

Сейчас она не существует, все это можно сделать в SE

3) Java mobile edition

Использовалась для написания приложений для смартфона в те далекие времена, когда они назывались персональный компьютер и к ним шёл стилус. Необходимость этой версии джаве в том, что эти устройства имели ограниченные ресурсы.

4) java card

На картах есть небольшая серая пластинка, для нее тоже надо писать софт

Плюсы джавы:

1) кроссплатформенность

Она достигается за счет того, что запуск происходит на виртуальной машине.

2) надежность

3) безопасность

Если вы обращаетесь к не пользовательской области данных, то jvm рухнет а не ось.

В браузере можно запускать код, написанный на Джаве. Они были безопасны. Сейчас это динамический веб.

*Лирическое отступление:*

1. *Компании пишут ЯП, как инструмент и не зарабатывают на нём, а зарабатывает на продуктах, на которых они написаны.*
2. *Профилировщики - отслеживаешь параметры. Скорость выполнения строчки. Можно использовать для взлома. Рутекер?*
3. *Памяти требуется намного больше, так как нужно выделить память для Виртуальный машины и для параметров.*
4. *Llvm - это использование Виртуальной машины для других языков.*

Jre - это только запуск, то есть виртуальная машина

jdk - и компилчтор и виртуальная машина

Сейчас там добавили jit

Джава - объектное ориентированный язык. Сейчас на уровне оптимизации есть реализация работы

Jvm.

Если пишем код, который должен выполниться, то jvm нужно назвать точку входу. Public static void main (string [], args []

Если из идешки, то просто указываем?

Если из командной строки. То можно создать исполняемый файл и создать там манифест.

Jar архив. Прога на первом курсе, генератор заданий.

Обратная совместимость. Много кода, его выбросить сложновато. Старые версии не работает на новых.

Jdk 9, может запуститься jdk 8.

Jdk 15 , не всегда запустить на jdk 7.

Vector - diplicated - устаревший

ArrayList - не потоко безопасен.

Есть обёртка, которая может позволить потоко безопасных.

Джава - консервативный.

Джава активно развивалась сообществом энтузиастов. Котлин - на Джаве.

По Джаве гугловский документ, который говорит, как нужно оформлять код.

# 

# 2. Синтаксис

Режим доступа:

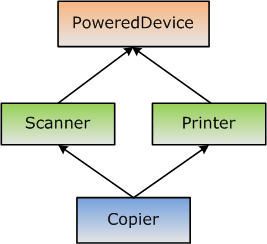
- private

- public

- protected – видимость внутри класса и его наследников

- --(без указателя) – внутри всего пакета

Проблема множественного наследования



Сканеры (Scanner) и принтеры (Printer) являются устройствами с питанием, поэтому они являются производными от PoweredDevice. Однако копировальный аппарат (Copier) включает в себя функции как сканеров, так и принтеров.

Проблемы:

1) кол-во копий PoweredDevice: по факту их 2 – от Scanner и от Printer. Но это же дублирование

2) разрешение определенных типов неоднозначных ссылок. То есть, нужно конкретно указывать к кому обращаемся

Как эту проблему решили в Java – убрали множественное наследование

Interface

Их создали с 2 целями:

1) полиморфизм – выполнение одного и того же действия разными способами.

Например, у нас есть класс Человек, у него есть метод info, который выводит информацию о нем (фамилия, имя, отчество, возраст). У класса Человек, есть наследник - класс Рабочий и у него тоже есть метод info. Но тут нужно переопределить этот метод, чтобы он выводил дополнительно еще и название компании.

В плюсах нужно в классе Человек, в методе info, написать virtual, а в классе Рабочий, в этом же методе, написать override. Внутри этого метода написать Человек::info и потом добавить вывод информации о компании

В джаве можно использовать интерфейсы

То есть создается интерфейс, в нем описывается сигнатура функции.

В классах, в которых эта функция будет реализована, к имени добавляется implements <имя интерфейса>

!! Тут важно отметить, что при использовании интерфейса в классе нужно описать все методы, которые содержатся в интерфейсе, иначе будет ошибка.

2) множественное наследование

!! В джаве запрещено множественное наследование классов, но не интерфейсов.

interface Add\_Sub {

    public void add(double x, double y);

    public void subtract(double x, double y);

}

interface Mul\_Div {

    public void multiply(double x, double y);

    public void divide(double x, double y);

}

interface Calculator extends Add\_Sub, Mul\_Div {

    public void printResult(double result);

}

public class MyCalculator implements Calculator {

    public void add(double x, double y)

    {

        double result = x + y;

        printResult(result);

    }

    public void subtract(double x, double y)

    {

        double result = x - y;

        printResult(result);

    }

    public void multiply(double x, double y)

    {

        double result = x \* y;

        printResult(result);

    }

Interface tasker

{ Public string getTask() }

Маркерный интерфейса.

Сервиризация.

Java 1.6. Далее усложняется.

Class Integer. Если передаем, то упаковке и пересылаем, получили, то распаковали.

Автоупаковка.

Неизменяемые классы.

Идеальны для хешей - можно использовать для ключей

Потоко безопасны - то есть не нужна синхронизация

Кодового слова нет, нужно нормально

Поля должны быть объявлены - final

Ссылки не куда не ведут.

String - неизменяемый класс

Если к текущей строке добавить символ, то по факту создаться дополнительный объект, в него запишется буква, а потом ещё один объект и туда запишется строка + буква.

Обратится к методу классу, но не хотим создавать экземпляр.

Static - можно обращаться не создавая объект

Напрмер

Class example

{

Static int logism;

}

В другом месте можно обращаться example. Logism

Логирование - применение.

Модификатор

Final - константы для типов

Final с методом не переопределить

Final класса - нельзя наследовать

Указатели :

Нельзя работать с памятью.

Класс scanner. Он работает очень долго

Флашить вывод - очищать буфер . Вывод сначала записывает в буфер, а потом выводится в файл или в консоль

Передача параметры

По значению - примитивные

Объекты - копия ссылки

Сравнить объекты

Str == str1 - сравнивает ссылки

То нужно сравнивать через метод

Коллекции

3 библиотеки

Сторонние позволяют работать с примитивными типами данных, а ещё расширяют базовые коллекции ( пример, set. Расширение - multi set)

[Изображение 201.jpg]

Низкоуровневые повторяющие конструкции - алгоритмы

Высокоуровневые - шаблоны. На уровне логики построении большой системы.

Шаблон iteruble - перебирать данные внутри структуры не опираясь на архитектуры структуры.

Потом всё это собирается в collection

Interface List

Подряд идущие элементы.

Array list - динамический массив

Interface set

Хранением уникальных объектов

Всегда нужно перееопредделить хеш код.

Hash set - хэш функция, считается алгоритмом. Класс можно создать экземпляр

Linked hash set - хэш таблица

Открытое и закрытое хеширование. Применяют сейчас закрытой.

Есть интервал, каждая значение хэш функции. .

0...М

Используется для отображение элементов большого множества в элементы маленького множества . Например, для сравнения строк можно прееобразвать их в числа и сравнивать их.

Принцип Дирихле - возникновение коллизии

Hash set хорош тогда, когда мало объектов

Linked hash set

Добавка: использует дублирующий Список чтобы знать в каком порядке они были добавлены

Int sorted set

Тут есть порядок :

Тут нужно вспомнить про структуру дерево, оно классное, так как у него есть порядок. Но сложность в том, что оно должно быть всегда сбалансированным.

Красно чёрное дерево

Вершинки покрашены в чёрные и красные

Внуки чёрного дедушки не могут быть красными. И тд. Это хорошо, таким как ограничения дают лёгкость в поддержке балансировки.

Tree set - красно чёрное дерево. В плюсах тоже есть

Квадро дерево используется для решения задач в пространств.

Окто дерево.

Int queue

Очередь всё просто. На следующий лекции рассказать, как реализовать очередь с приоритетом.

В предыдущей ветке хранились множества

Тут хранятся пары значений

Hash map

Ключ базируется на хэше

Weak

Алгоритм сборщика мусора - подсчёт ссылок. Слабая ссылка не учитывается сборщиком мусором. Организация кеширования.

Tree map

Хранение хеш таблицы при помощи красно чёрного дерева.

Аналоги:

Trove library - с примитивными типами

PCJ - мульти сеты

Guava - класс утилиты со статическими методами

Устаревшие коллекции

Enuneration - итератор

Stack

Vector

Dictionary ?

HashTable

Вроде бы, это потоки безопасно, но в Java 1.6. Добавили

Collections

Synchronized Map

Synchronized List

Пример:

Map m = collections. Synchronized Map ( new HashMap { integerInt, integer}())

Исключения

[Изображение 208.jpg]

Что-то поломалось - создаётся объект в нём

инкапсулируется ошибка.

Exception - ошибки обыкновенные

Error - ошибка с взаимодействием jvm .

- check ошибки

O - don't check ошибки

Рефлексия? Не хорошо использовать по мнению га

Когда возникает ошибка, обрабатывать try catch .

Если относится к check , то обрабатываем их обязательно. То есть jvm сама не обрабатывает этот код.

Uncatch не обязательно проверять. То есть jvm сама это проверяет. Сделано это с учётом того, чтобы ваш код не превратился в один обратки ошибок.

Деление на 0 может возникнуть часто.

Блок finally. Ну это хороший тон.

Throw new ioexeption (..) ;

Public int getSun ( int [] array) throws ioexception

{ };

Интерфейс  
Awt  
Swing   
JavaFX  
Awt - самый первый вариант. Но были проблемы. Использовались тяжёлые компаненты, относящиеся к ОС. И тут проблема в том, что это нарушает принцип кроссплатформенность.  
Параллельно создавалось swing - легковесные компоненты.   
Новый фреймворк javafx . Сейчс не перспективная ветвь . Попытка создать декларативный язык. Предполагалось, что будет и для компов, и для мобилок и всё будет красиво. Но не получилось   
   
Swing   
Первоначально нужно было всё писать ручками. Сейчас в идешках есть кидание на форму. Но это плохой тон, в лабах всё ручками.   
Концепция lookandfeel  
 Как видится компонент и как обрабатываются   
Изменение поведения на лету бывает полезно. Например, при разных правах.   
Концепция layout  
Как добиться кроссплатформенности. Менеджер компановки. Сам знает, как разместить компоненты, но прислушивается к нам.   
  
Элементы контейнеры - в них можно помещать другие объекты. ( конечные объекты - кнопка)   
J frame, j panel, j window, j dialog, japplet  
  
Borderlayout . Делит зону на 5 объектов.   
Flowlayout. Размещает слева - направо, сверху -вниз  
Grid layout - делит объект на табличку.   
GridBagLayout - можно регулировать размер строк и столбцов.   
Box layout - последовательность коробочек, друг на дружку  
Card layout - несколько менеджеров компановщиков. При переключении и быстрой реакции норм.   
SpringLayout - ?   
  
Концепция mvc  
Компоненты: модель, представление и контроллер.   
Модель - данные  
Представление - представление данных  
Обработка действий - контроллер  
Пунктирная чёрта - информационная связь   
Активная модель - информирует заинтересованная модель о том, что   
У вас есть датчики, снимающие данные. Когда получается новое значение, то всё заинтересованные должны быть проинформированы. Например, в графике появляется новая точка.   
Пассивная - через таймер происходит обработка новых значений, то есть обращение к буферу.   
Реализовано на шаблонах проектирования  
1. Наблюдатель   
Определяет связь один ко многим. Это про информированность других объектов.   
В swing - listner.   
2. Стратегия   
Поведенческая, алгоритм работы.   
Помогают избавиться от ифов, стратегия выбирается в runtime. Тут страдает принцип инкапсуляции, потому что пользователь должен знать о стратегиях.   
3. Компоновщик   
Строит дерево.   
Swing - панелька внутри панельки, которые позволяют задавать им свойства.   
  
Данные - модели. К ней подключаются наблюдатель.120:12