Реферат

Пояснительная записка 20 листов, 1 рисунок, 2 приложения, 4 источника.

JAVA, ОБЪЕКТ, КЛАСС, СТАТИЧЕСКИЙ ИНИЦИАЛИЗАТОР, СЧЁТЧИК, ПРОГРАММА, ПЕРЕМЕННАЯ

Предметной областью разработки является задача объектно-ориентированного программирования — подсчёт количества созданных объектов класса.

Цель работы – разработать консольное приложение на языке Java, которое с помощью статического инициализатора будет отслеживать и выводить количество созданных объектов определённого класса.

В результате работы реализовано приложение на Java, использующее статический инициализатор для автоматического увеличения счётчика при каждом создании нового объекта. Также осуществлён контроль пользовательского ввода, проведено тестирование и подготовлена документация.

При разработке использовалась среда программирования IntelliJ IDEA.

Содержание

[Введение 5](#_Toc167966024)

[1 Общий раздел 7](#_Toc167966025)

[1.1 Анализ предметной области 7](#_Toc167966026)

[1.2 Формирование требований к программному продукту 8](#_Toc167966027)

[1.3 Обоснование выбора программно-аппаратных средств 9](#_Toc167966028)

[2 Специальный раздел 11](#_Toc167966029)

[2.1 Разработка структуры программного продукта 11](#_Toc167966030)

[2.2 Описание разработанного приложения 12](#_Toc167966033)

[2.3 Документирование программного продукта 13](#_Toc167966035)

[Заключение 14](#_Toc167966036)

[Список используемых источников 15](#_Toc167966037)

[Приложение А 16](#_Toc167966038)

[Приложение Б 18](#_Toc167966039)

Введение

Актуальность темы курсовой работы:

В условиях широкого распространения объектно-ориентированного программирования и роста масштабов программных систем особую значимость приобретает умение эффективно управлять созданием и учётом объектов классов. Одним из ключевых инструментов для реализации такого контроля являются статические члены и инициализаторы классов в языке Java.

В ряде задач важно отслеживать количество созданных объектов определённого класса — например, при контроле ресурсов, моделировании поведения системы, отладке, сборе статистики или построении ограничений в приложении. Подобный функционал широко используется как в учебных целях для демонстрации принципов ООП, так и в профессиональной практике при разработке сложных программных систем.

В связи с этим, разработка программного решения, позволяющего автоматически отслеживать количество созданных объектов с использованием статического инициализатора, является актуальной и практически значимой задачей.

Целью курсовой работы является создание консольного приложения на языке Java, в котором реализован механизм подсчёта количества созданных объектов класса с использованием статического инициализатора. Программа должна быть простой в использовании и наглядно демонстрировать принцип работы данного механизма.

Задачи курсовой работы:

— Изучить теоретические основы статических инициализаторов и статических переменных в языке Java.

— Разработать архитектуру консольного приложения на Java для демонстрации подсчёта объектов.

— Реализовать приложение, в котором создаются объекты класса и подсчитывается их количество.

— Провести тестирование программы и подготовить документацию по результатам работы.

Таким образом, выбранная тема курсовой работы обладает актуальностью. Работа демонстрирует практическое применение навыков программирования на языке Java, а также использование знаний из области математики и алгоритмизации для решения прикладной задачи.

.

1 Общий раздел

1.1 Анализ предметной области

В рамках данной задачи необходимо изучить особенности работы статических инициализаторов и статических переменных в объектно-ориентированном программировании на языке Java. Статические инициализаторы представляют собой специальные блоки кода, которые выполняются один раз при загрузке класса в память, и могут использоваться для начальной настройки статических переменных.

Цель — реализовать механизм, позволяющий автоматически отслеживать количество созданных объектов определённого класса. Для этого используется статическая переменная-счётчик, которая увеличивается каждый раз, когда создаётся новый экземпляр класса.

Такая концепция может применяться в различных сферах: от анализа нагрузки в многопоточных приложениях до отладки, профилирования, ограничения числа объектов или контроля использования ресурсов.

Важным моментом является понимание жизненного цикла объекта, отличия между статическими и нестатическими членами класса, а также синтаксиса и порядка инициализации в Java. Также следует рассмотреть альтернативные подходы к подсчёту объектов — например, через конструктор — и обосновать выбор в пользу использования статического инициализатора.

Применение статического инициализатора в языке Java позволяет обеспечить однократную инициализацию статических переменных при первом обращении к классу. Такой механизм удобно использовать для подсчёта количества созданных объектов класса, так как он исключает повторную инициализацию и сохраняет точность подсчёта.

* 1. Формирование требований к программному продукту

— Программа должна демонстрировать создание объектов некоторого класса и подсчёт их количества.

— Подсчёт должен осуществляться с использованием статического инициализатора и статической переменной-счётчика.

— Объекты создаются в рамках основной программы (в main) по заданному количеству, вводимому пользователем.

— Ввод должен осуществляться с клавиатуры через консоль. Ожидается ввод целого положительного числа n, задающего количество создаваемых объектов.

— После создания объектов программа должна вывести результат в понятной форме: "Количество созданных объектов: count", где count — фактическое значение счётчика.

— Программа должна быть кроссплатформенной и успешно компилироваться и запускаться в любой среде, поддерживающей JDK (Java Development Kit).

Сформулированные требования обеспечивают надёжную реализацию задачи подсчёта количества объектов с использованием статического инициализатора. Учитываются все аспекты: корректность ввода, точность учёта объектов и понятность вывода результата.

1.3 Обоснование выбора программно-аппаратных средств

В качестве языка программирования для разработки приложения был выбран Java. Данный язык является одним из самых популярных и широко используемых в мире благодаря своей объектно-ориентированной природе, высокой надёжности и кроссплатформенности. Java предоставляет разработчику мощные инструменты для реализации принципов ООП, таких как инкапсуляция, наследование и полиморфизм. Особенно важно то, что язык Java позволяет удобно работать со статическими членами класса и статическими инициализаторами, что критично для решения задачи текущей курсовой работы.

Одним из ключевых преимуществ Java является то, что данный язык работает на виртуальной машине Java (JVM), благодаря чему программы, могут быть запущены на любой платформе, где установлена JVM. Это делает Java идеальным выбором для реализации кроссплатформенных решений, включая учебные и демонстрационные приложения.

Для реализации проекта использовалась интегрированная среда разработки IntelliJ IDEA Community Edition. Указанная IDE предоставляет мощные возможности для написания, отладки и тестирования кода. Среди преимуществ — интеллектуальное автодополнение, удобная навигация по проекту и встроенный анализатор кода. Указанные инструменты особенно полезны при работе с объектно-ориентированными конструкциями, такими как классы, статические члены и инициализаторы, что делает среду максимально подходящей для целей работы.

Разработка и тестирование приложения осуществлялись на операционной системе Windows 10, которая предоставляет стабильную и производительную среду для запуска Java-приложений. Благодаря популярности, Windows 10 позволяет создавать и тестировать программы, которые будут востребованы широкой аудиторией. Тем не менее, благодаря кроссплатформенной природе Java, созданное приложение может успешно запускаться и на других операционных системах — таких как macOS или Linux — при наличии установленной версии JDK 8 или выше.

Для функционирования программы не требуется подключение к сети или использование сторонних библиотек, что делает приложение простым в установке и использовании. Это особенно важно для студентов и начинающих разработчиков, которые могут не обладать обширным техническим опытом.

Таким образом, выбор языка Java, использование IntelliJ IDEA и ориентация на кроссплатформенность обеспечивают надёжную техническую основу для реализации задачи по подсчёту количества созданных объектов класса с использованием статического инициализатора, соответствуя целям и задачам курсовой работы.

2 Специальный раздел

2.1 Разработка структуры программного продукта

Программное приложение состоит из трёх основных компонентов:

— Модуль ввода данных - реализует поочерёдное считывание имени и фамилии с консоли с помощью класса Scanner. Ввод продолжается в цикле до тех пор, пока пользователь не введёт символ n вместо имени или фамилии. Такая реализация позволяет гибко управлять завершением ввода и предотвращает создание лишних объектов.

— Модуль создания объектов и подсчёта - выполняет основную задачу: при каждом вводе имени и фамилии создаётся новый объект класса Person. В этом классе реализован статический инициализатор, устанавливающий начальное значение счётчика, а также статическая переменная count, которая увеличивается каждый раз при создании объекта. Таким образом, обеспечивается подсчёт общего количества созданных объектов.

— Модуль вывода - по завершении ввода программа вызывает статический метод display() класса Person, который выводит на экран количество созданных объектов в понятной форме: Количество созданных объектов: count.

Логика программы реализована в классе Main, содержащем метод main, который служит точкой входа в приложение. Также используется вспомогательный класс Person, в котором сосредоточена логика инициализации и подсчёта объектов. Такое разбиение на два класса удобно для учебных целей, позволяет наглядно продемонстрировать принципы объектно-ориентированного программирования, работу со статическими членами класса и статическим инициализатором.

Предложенная структура программы демонстрирует последовательную реализацию задачи: ввод данных, создание объектов, инкремент счётчика в статическом контексте и вывод результата. Такой подход подчёркивает применимость статического инициализатора для управления общим состоянием класса.

2.2 Описание разработанного приложения

Программа написана на языке Java и предназначена для выполнения в консольной среде. При запуске пользователю предлагается поочерёдно вводить фамилию и имя. Ввод обрабатывается с помощью объекта Scanner. Цикл продолжается до тех пор, пока пользователь не введёт 'выход' вместо имени или фамилии, после чего программа завершает ввод и выводит результат.

При каждом вводе данных создаётся новый объект класса Person, в результате чего срабатывает конструктор, который увеличивает значение статической переменной count. Для инициализации этой переменной используется статический инициализатор, который гарантирует установку начального значения только один раз при первом обращении к классу.

После завершения цикла вызывается статический метод display(), который выводит в консоль количество созданных объектов. Вывод позволяет пользователю увидеть, сколько объектов было создано в рамках текущего запуска программы. Такой подход демонстрирует работу со статическими полями, статическими методами и статическим инициализатором в Java, а также применение конструктора для инкрементации счётчика.

Программа реализует механизм подсчёта экземпляров класса через статическую переменную, инициализированную в статическом блоке. Данный способ позволяет обеспечить точный и надёжный учёт без необходимости дополнительной логики в основной части программы.

2.3 Документирование программного продукта

Программа снабжена встроенными комментариями, поясняющими работу основных компонентов: статический инициализатор, конструктор, цикл ввода, метод для проверки условия завершения и вывод результата. Присутствие вспомогательных элементов упрощает понимание логики программы и делает пригодной для использования в учебных целях.

Как пользоваться программой:

1. Убедитесь, что на компьютере установлен JDK версии 8 или выше.
2. Скопируйте исходный код программы в текстовый файл с расширением .java, например Main.java.
3. Откройте командную строку или терминал.
4. Перейдите в директорию, где находится файл, и выполните команды:

javac Main.java

java Main

1. Следуйте инструкциям в консоли:

— Вводите имя и фамилию. Для завершения можно ввести символ n вместо любого из значений.

— После завершения программа выведет количество созданных объектов.

Программа не требует дополнительных библиотек, графического интерфейса или подключения к интернету. Демонстрируются основы объектно-ориентированного программирования: работа с классами, конструкторами, статическими членами, а также с консольным вводом и управляющими структурами языка Java.

Документация и структура кода позволяют легко разобраться в назначении каждой части программы, подчёркивая использование статического инициализатора как ключевого элемента, что делает продукт полезным как учебный пример по теме.

Заключение

В результате выполнения курсовой работы была достигнута поставленная цель — разработано программное приложение на языке Java, которое позволяет подсчитывать количество созданных объектов класса с помощью статического инициализатора.

Программа успешно реализует ключевые принципы объектно-ориентированного программирования, включая использование статических полей, методов и инициализаторов. В ходе выполнения пользователю предоставляется возможность создавать объекты через ввод данных, после чего программа выводит общее количество созданных экземпляров класса. Реализация получилась простой, наглядной и эффективной для учебных целей.

Таким образом, все задачи, поставленные в начале проекта, были выполнены. Цель работы достигнута.

Список используемых источников

1. Эккель, Б. Философия Java. — 4-е изд. — СПб.: Питер, 2022. — 1168 с. — ISBN 978-5-469-01867-8.
2. Документация Java Platform Standard Edition 8. — Текст: электронный // Oracle : [сайт]. — URL: https://docs.oracle.com/javase/8/docs/ (дата обращения: 02.05.2025).
3. Metanit. Java — Основы программирования (Статические поля и инициализаторы) — [сайт]. — URL: https://metanit.com/java/tutorial/3.6.php (дата обращения: 02.05.2025).
4. JavaRush. Статические блоки и статические поля. — Текст: электронный // JavaRush : [сайт]. — URL: https://javarush.com/groups/posts/staticheskie-bloki-i-polya (дата обращения: 02.05.2025).

Приложение А

(обязательное)

Блок схема

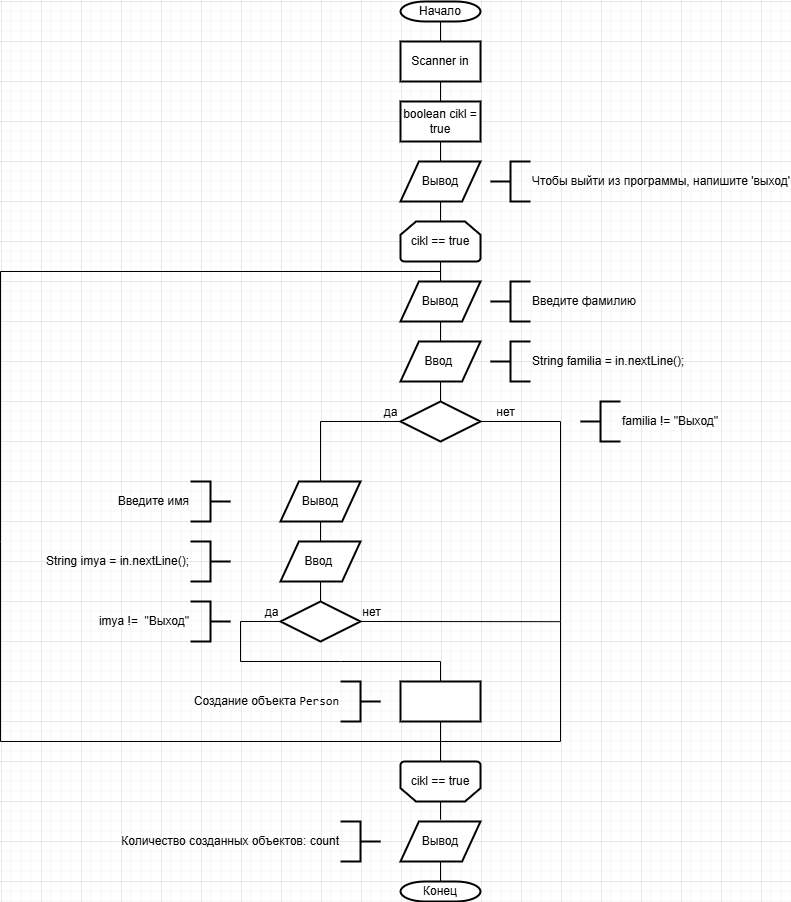


Рисунок 1 – Блок-схема программы

Приложение Б

(обязательное)

Листинг кода

import java.util.Scanner;

class Person{

String Fn;

String Sn;

static int count;

static{

count = 0;

}

Person(String Fn, String Sn){

this.Fn = Fn;

this.Sn = Sn;

count++;

}

static void display(){

System.out.println("Количество созданных объектов: " + count);

}

}

public class Main

{

public static boolean kila(String text){

if(!text.equals("выход")){

return true;

}

return false;

}

public static void main(String[] args) {

System.out.println("Чтобы выйти из программы, напишите 'выход'");

boolean cikl = true;

Person p;

Scanner in = new Scanner(System.in);

while(cikl){

System.out.println("Введите фамилию");

String familia = in.nextLine();

if(!kila(familia)){

break;

}

System.out.println("Введите имя");

String imya = in.nextLine();

if(!kila(imya)){

break;

}

p = new Person(familia, imya);

}

Person.display();

}

}