Минобрнауки россии

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

“ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ”

(ФГБОУ ВО «ВГУ»)

Факультет компьютерных наук

Реализация структуры данных HashMap на языке Java и ее отображение в виде графа в формате Graphviz

Проектная работа

4 семестр 2024/2025 учебного года

09.03.02 Информационные системы и технологии

Обучающийся \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ст. 2 курса Е.А. Аверкиева

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ст. преп. Н.К. Самойлов

Воронеж 2025

ВВЕДЕНИЕ

Современные вычислительные задачи требуют эффективных структур данных, обеспечивающих быстрый доступ, вставку и удаление элементов. Одной из таких структур является HashMap. Она позволяет хранить пары "ключ-значение" и обеспечивает быстрый доступ к данным за счет хеширования. В языке Java HashMap реализован как часть стандартной библиотеки (в пакете java.util) и широко применяется в различных задачах.

Однако для глубокого понимания работы HashMap важно не только знать его интерфейс, но и представлять внутреннюю организацию, включая принцип хеширования, разрешение коллизий и расширение таблицы. Визуализация структуры данных в виде графа позволяет наглядно продемонстрировать эти механизмы, упрощая анализ и отладку. Инструмент Graphviz предоставляет удобный способ генерации таких графических представлений.

Основной целью проектной работы является реализация структуры данных HashMap на языке программирования Java и ее отображение в виде графа в формате Graphviz. Основные требования к создаваемой программе формулируются следующим образом:

* Изучение принципов работы HashMap.
* Реализация собственной версии HashMap на Java.
* Исследование работы HashMap при различных сценариях.
* Визуализация структуры данных HashMap в виде графа с помощью Graphviz.

1. Теоретические основы HashMap

**Хеш-таблицей (**HashMap) называется структура данных, реализующая интерфейс **ассоциативного массива** (абстрактная модель «ключ – значение» или entry), которая обеспечивает очень быструю вставку и поиск. По сути, это обычный массив, где местоположение элемента зависит от ключа самого элемента. **Связь между ключом элемента и его позицией в хеш-таблице задает хеш-функция.**

**Хеш-функция** получает входную часть данных, которую мы называем **ключом,** а на выходе она выдает целое число, известное как **хеш-значение (или хеш-код).** Затем, **хеш-значение** привязывает наш ключ к определенному индексу хеш-таблицы. Для основных операций: вставки, поиска и удаления мы используем одну и ту же хеш-функцию, поэтому эти операции осуществляются довольно быстро.

1. Реализация

2.1 Средства реализации

В качестве средств реализации для разработки программы для наглядной демонстрации структуры данных HashMap использованы:

* объектно-ориентированный язык Java,
* пакет java.util,
* модуль JUnit Jupiter,
* пакет Graphviz.

2.2 Логика приложения

Разработанная программа состоит из классов, представленных в виде диаграммы на рисунке 1:

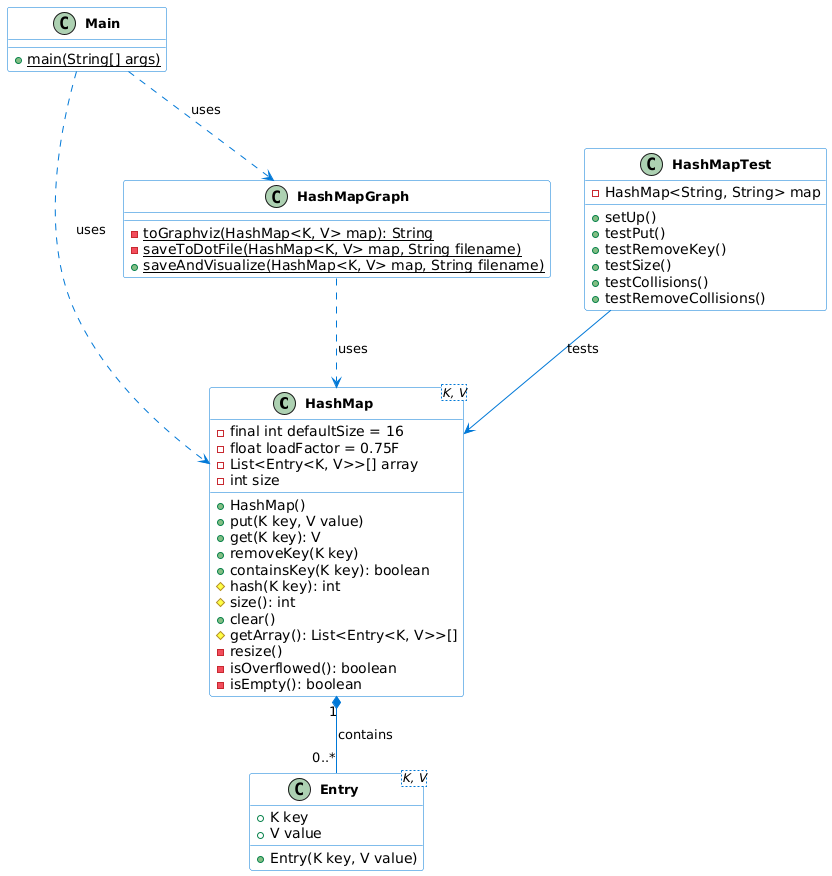


Рисунок 1 – Диаграмма классов

* 1. Класс Main

Класс Main является основным классом данной программы. В нем находится точка входа в программу. Данный класс демонстрирует работу класса HashMap и вызывает визуализацию HashMap через Graphviz.

2.4 Класс HashMap

Данный класс предоставляет стандартный интерфейс структуры данных HashMap. включая операции добавления, получения и удаления элементов. Внутренняя структура данных представляет собой массив связных списков, где каждый список соответствует определённой корзине (бакету) в хеш-таблице. При добавлении элементов вычисляется хеш-код ключа, который определяет индекс бакета. Если в одном бакете оказывается несколько элементов (коллизия), они хранятся в виде цепочки узлов.

Класс содержит основные методы работы с хеш-таблицей, такие как put, get, removeKey, containsKey, hash, и тд. Данные методы обеспечивают выполнение различных операций со структурой данных, таких как добавление, получение и удаление элементов.

* 1. Класс HashMapGraph

Класс HashMapGraph предназначен для визуализации структуры хеш-таблицы. Он генерирует описание графа в формате DOT, который затем может быть преобразован в графическое изображение с помощью утилиты Graphviz. Визуализация отображает все корзины таблицы, цепочки коллизий, а также ключи и значения элементов. Это позволяет наглядно анализировать распределение элементов. Например, на рисунке 2 представлен граф, который может получиться в результате работы программы.

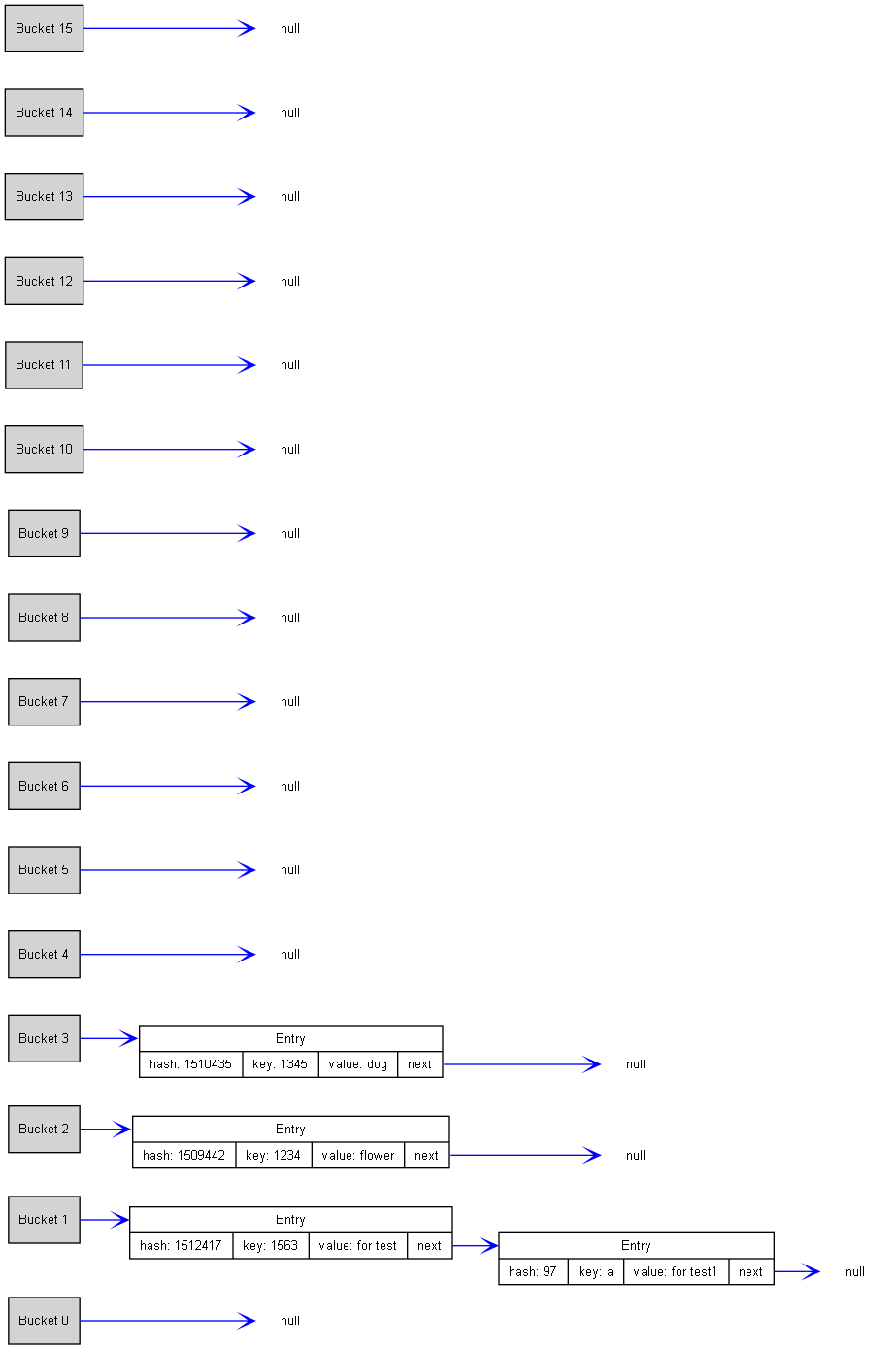


Рисунок 2 – Визуализация HashMap

* 1. Класс HashMapTest

Тестирование функциональности осуществляется с помощью класса HashMapTest, содержащего набор юнит-тестов. Тесты проверяют основные сценарии работы: добавление и обновление элементов, удаление, обработку коллизий, корректность подсчёта размера таблицы.

* 1. Пример работы

Приведем пример работы данной программы. Допустим, что нам необходимо в пустую хэш-таблицу добавить значение «apple» с ключом «1563».

При создании объекта HashMap инициализируется массив фиксированного размера (16 элементов), где каждая ячейка представляет собой связный список. Это позволяет эффективно хранить данные и обрабатывать коллизии. Начальный размер структуры устанавливается в ноль, а коэффициент загрузки, определяющий момент увеличения вместимости, по умолчанию равен 0.75.

Для добавления элемента применяется метод put. При вызове данного метода сначала проверяется загруженность массива (метод isOverflowed), если он переполнен, то вызывается метод resize, с помощью которого массив расширяется в два раза, и все элементы перераспределяются по новым индексам. После вычисляется хэш-код ключа. Для этого используется стандартный метод hashCode, который преобразует ключ в целое число. Чтобы гарантировать, что индекс попадает в границы массива, применяется операция взятия модуля от абсолютного значения хэш-кода по длине массива. В нашем случае хэш-код ключа «1563» равен 1512417, индекс будет вычислен как 123456 % 16, что, в результате, даёт 1.

После определения индекса проверяется, есть ли в соответствующем бакете элементы с таким же ключом. Если ключ уже существует, его значение обновляется. В противном случае в связный список добавляется новая запись (Entry), содержащая ключ и значение. В нашем случае, все бакеты были свободны, так как хэш-таблица по условию пустая, поэтому значение «apple» с ключом «1563» добавляется в 1 бакет.

Теперь представим, что нам необходимо в данную хэш-таблицу добавить новое значение «apple1» с ключом «a».

Его хэш-код равен 97, что при вычислении индекса 97 % 16 даёт 1. Из-за того, что в данном бакете уже хранится ранее добавленное нами значение, возникает коллизия. Ее разрешение происходит методом цепочек. То есть элемент добавится в виде связного списка, и при поиске или удалении будет происходить последовательный перебор записей в этом списке. Таким образом, добавленные нами хэш-коды «1563» и «a» окажутся в одном бакете, образуя цепочку.

В конце работы программы, после добавления всех необходимых нам элементов вызывается метод saveAndVisualize, который визуализирует получившуюся хэш-таблицу и позволяет наглядно увидеть распределение элементов по бакетам. Пустые бакеты отображаются как null, а заполненные — как узлы с указанием ключа, значения и ссылки на следующий элемент в случае коллизии. Окончательный результат работы программы при добавлении значения «apple» с ключом «1563» и значения «apple1» с ключом «a» приведен на рисунке 3.

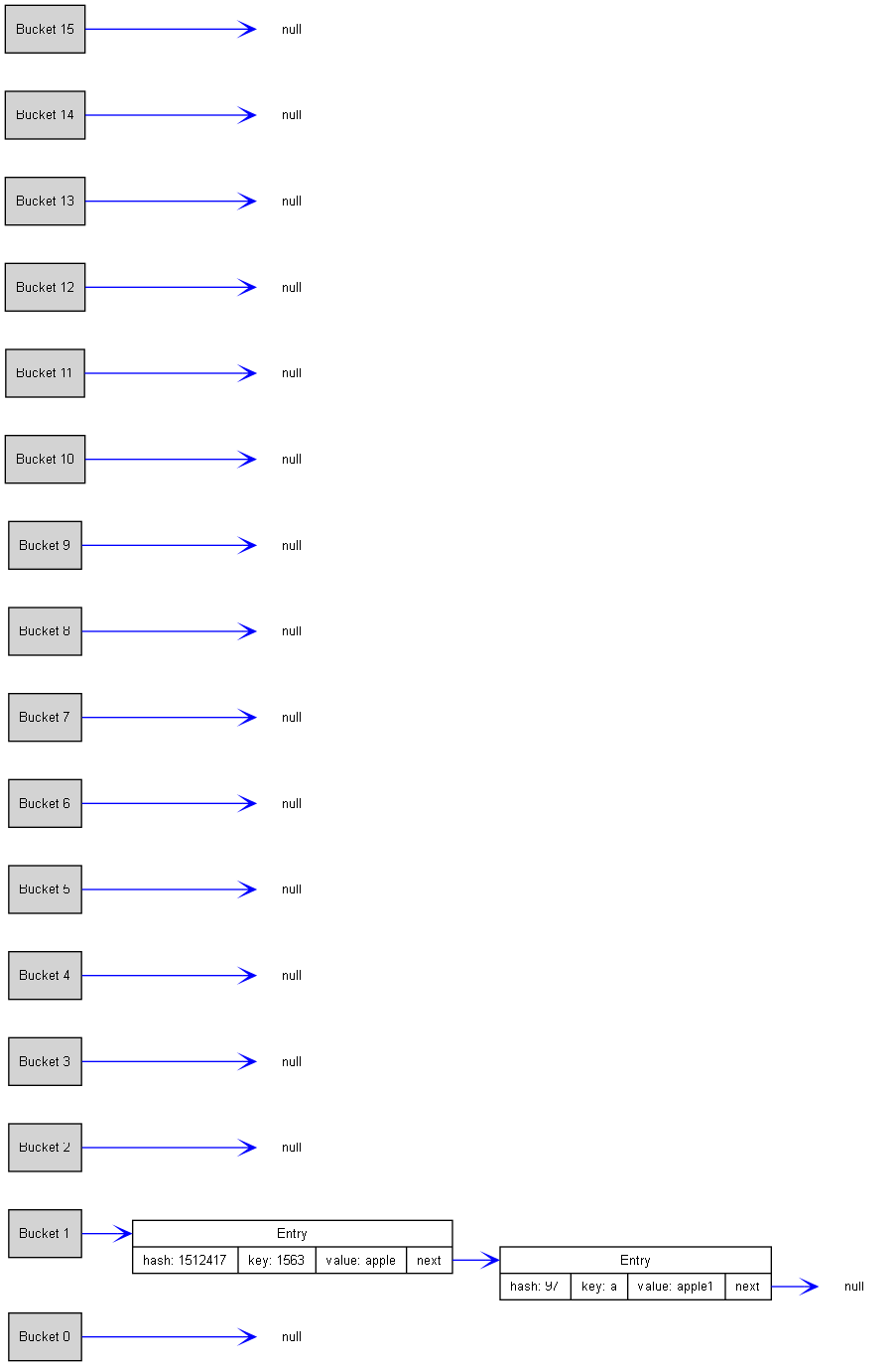


Рисунок 3 – Результат работы программы

Таким образом, HashMap обеспечивает эффективное хранение и доступ к данным, автоматически расширяясь при необходимости и корректно обрабатывая коллизии.

ЗАКЛЮЧЕНИЕ

В результате выполненной проектной работы были изучены принципы работы HashMap. Была реализована собственная версия HashMap на языке программирования Java, а также сделана ее визуализация в виде графа с помощью Graphviz.