**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**Ордена трудового Красного Знамени федеральное государственное бюджетное**

**образовательное учреждение высшего образования**

**«Московский технический университет связи и информатики»**

Кафедра Математическая кибернетика и информационные технологии

Лабораторная работа №3

**Класс Object. Работа с хэш-таблицами**

**по дисциплине  
«Информационные технологии и программирование»**

Выполнил: студент гр. БВТ2403

Титов Д.С.

Проверил:

Москва, 2025 г.

**Цель работы:** изучение и практическое применение структур данных типа **хэш-таблица** на языке **Java** путём самостоятельной реализации хэш-таблицы с помощью метода цепочек и использования встроенного класса HashMap, а также отработка операций вставки, поиска и удаления элементов.

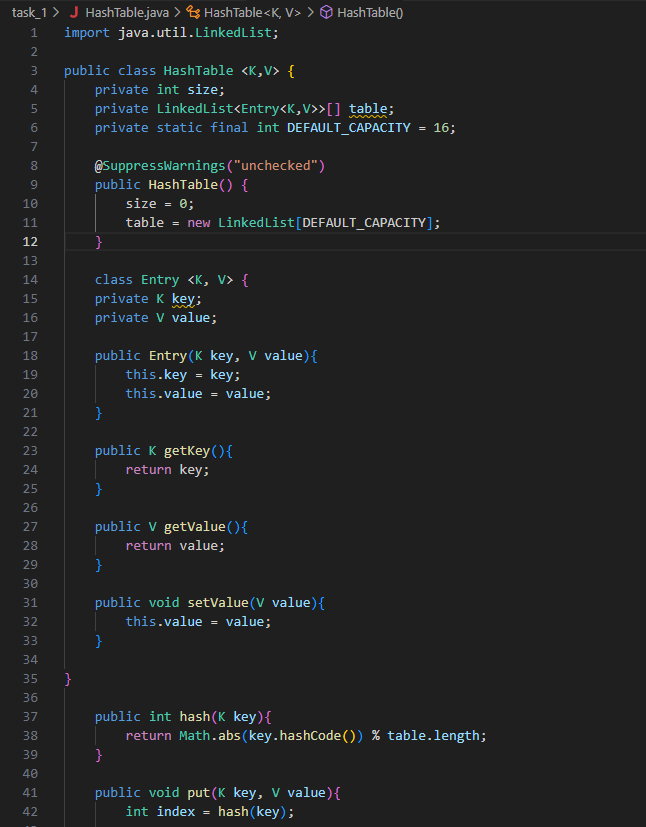
Основная часть

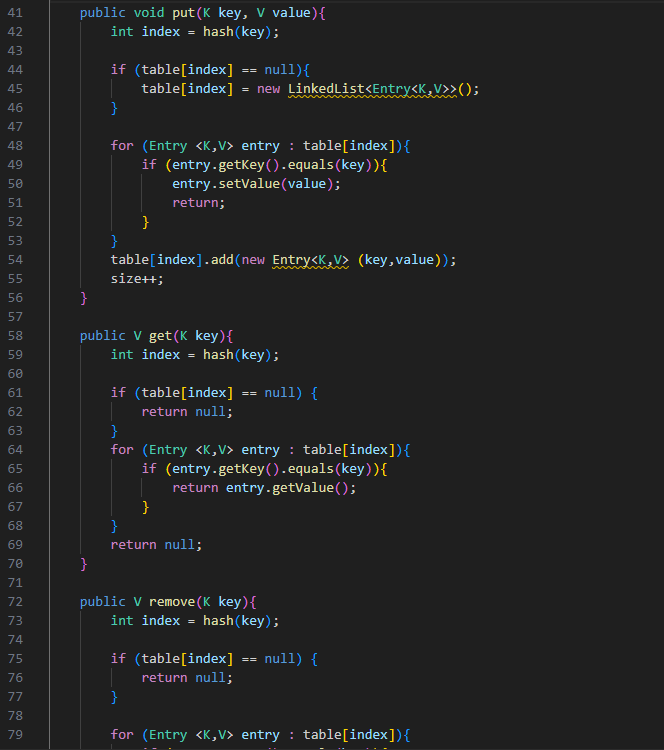
Задание 1. Реализация класса HashTable

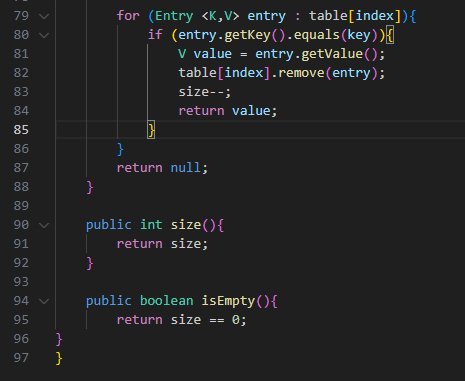
Для выполнения первого задания был разработан универсальный класс HashTable<K,V>, который реализует функционал хэш-таблицы. В качестве основного метода разрешения коллизий был выбран метод цепочек (см. условие задания на). Внутри класса использован приватный массив списков (LinkedList) для хранения элементов, что позволяет каждой ячейке массива хранить несколько пар «ключ-значение», образующих цепочку.

Класс включает приватный внутренний класс Entry<K,V> для инкапсуляции пары ключа и значения. Вся логика хэширования, которая преобразует хэш-код ключа в индекс массива, реализована в приватном методе hash(K key).

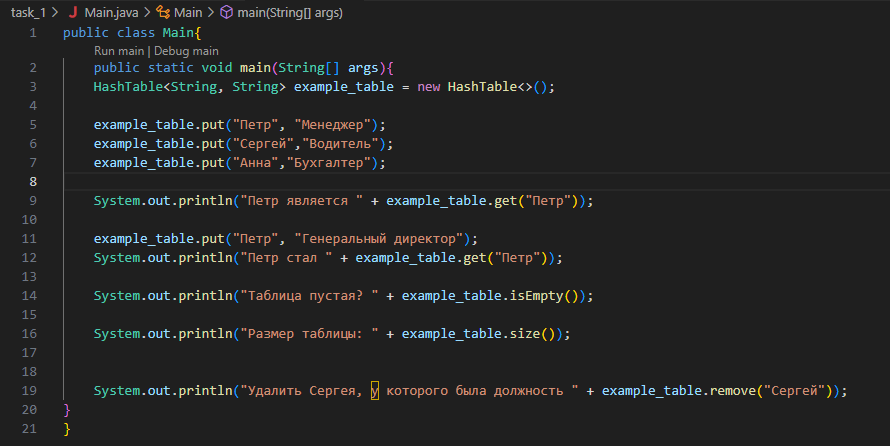
Основной функционал реализован через публичные методы, обеспечивающие полный цикл работы с хэш-таблицей. Метод put(K key, V value) отвечает за добавление новой пары или обновление значения, если ключ уже существует. При добавлении новой записи размер таблицы увеличивается. Метод get(K key) осуществляет поиск значения по ключу, последовательно перебирая элементы в соответствующей цепочке. Удаление элемента выполняется методом remove(K key), который находит и извлекает запись из списка, уменьшая при этом размер таблицы. Для контроля состояния таблицы также реализованы методы size() и isEmpty().







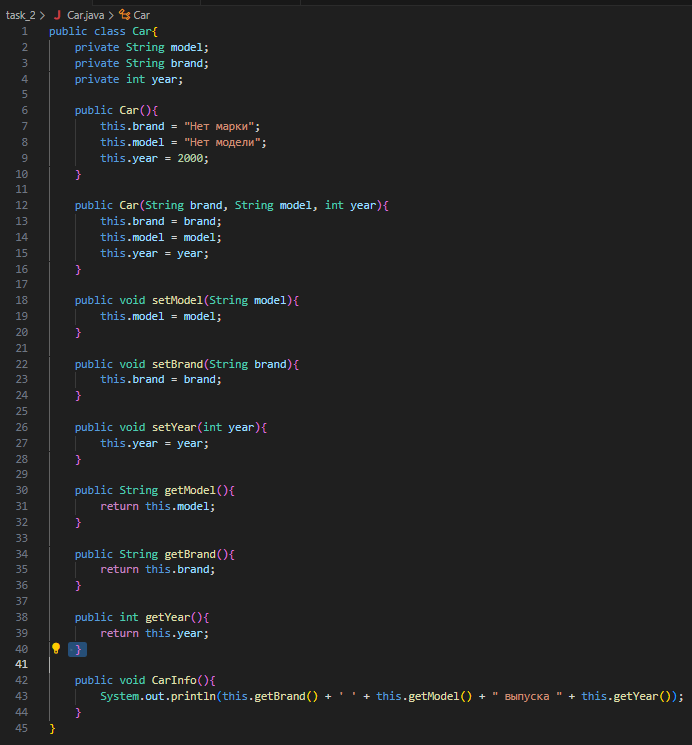
Демонстрация работы класса HashTable была проведена в классе Main. Тестирование включало в себя последовательное добавление элементов, проверку возможности обновления значения по существующему ключу, вывод состояния таблицы с помощью size() и isEmpty(), а также успешное удаление одной из записей.



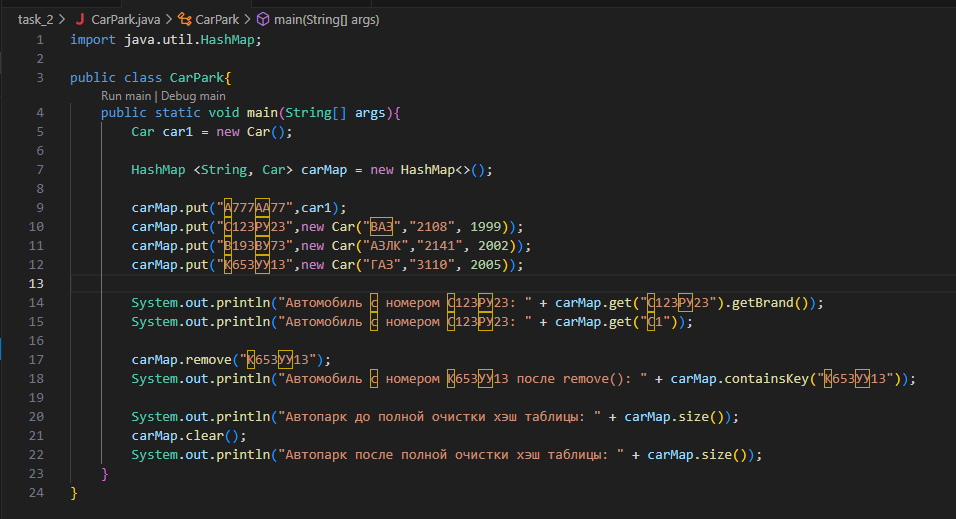
Задание 2. Работа со встроенным классом HashMap

Второе задание предполагало создание системы учета автомобилей в автопарке с использованием встроенного класса HashMap (см. условие задания на). В этой системе в качестве ключа используется номерной знак автомобиля (тип String), а в качестве значения — объект класса Car.

Был разработан вспомогательный класс Car, который инкапсулирует основные характеристики транспортного средства: модель, марку и год выпуска. Класс Car оснащен конструкторами, а также геттерами и сеттерами для контролируемого доступа к приватным полям.



В классе CarPark создается экземпляр HashMap<String, Car>. Здесь продемонстрированы основные операции по управлению автопарком. Вставка новых автомобилей (put) выполняется с использованием номерного знака в качестве уникального идентификатора. Поиск (get) позволяет быстро извлечь объект автомобиля по его номеру. Операция удаления (remove) демонстрирует, как можно исключить автомобиль из учета. Также в коде показаны служебные методы size() и clear(), которые позволяют контролировать общее количество автомобилей и полностью очищать учетную базу.



Вывод: в результате выполнения лабораторной работы были изучены и практически применены принципы работы со структурой данных хэш-таблица на языке Java. Успешно реализована собственная версия хэш-таблицы (HashTable) с использованием метода цепочек для разрешения коллизий, а также продемонстрирована эффективная работа со встроенным классом HashMap в рамках прикладной задачи по учету автопарка. Таким образом, поставленная цель по освоению механизмов хэширования, вставки, поиска и удаления элементов достигнута.