**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

Лашков Ю.Е.

Руководитель:

Москва, 2025 г.

**Цель работы:**Изучить особенности базового класса Object в Java и его ключевые методы, а также рассмотреть принципы работы хэш-таблиц и их реализацию в виде классов HashMap и Hashtable.

**Индивидуальное задание:**Разработать программу на Java, демонстрирующую использование хэш-таблиц (HashMap) для хранения и обработки объектов пользовательского класса. Создать класс Student с полями: имя, фамилия, возраст, номер студенческого билета и средний балл. Переопределить методы equals() и hashCode() для корректного сравнения объектов и их работы в хэш-коллекциях. Реализовать добавление, удаление, поиск и вывод студентов с использованием HashMap, где ключом является номер студенческого билета, а значением — объект класса Student.

**Основная часть:**  
Класс Object является **базовым классом всей иерархии классов в Java.** Все остальные классы, включая пользовательские, неявно наследуют Object. Это означает, что любой объект в Java имеет методы, определённые в этом классе. Среди них:

toString() — возвращает строковое представление объекта;

equals(Object obj) — проверяет равенство объектов;

hashCode() — возвращает целое значение хэш-кода объекта;

getClass() — возвращает метаданные класса;

clone() — создаёт копию объекта;

wait(), notify(), notifyAll() — используются при синхронизации потоков.

Особое значение имеют методы **equals()** и **hashCode()**, поскольку они определяют, как объекты сравниваются и хранятся в коллекциях, основанных на хэш-таблицах, таких как HashMap и Hashtable.

По умолчанию метод equals() сравнивает **ссылки на объекты**, а не их содержимое. Чтобы сравнение было логическим (по данным), этот метод необходимо переопределить. При этом важно также переопределить hashCode(), чтобы два равных объекта имели одинаковые хэш-коды.

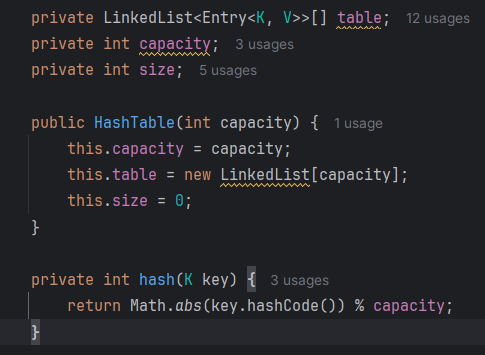
Нарушение этого контракта может привести к некорректной работе хэш-коллекций — например, объект не будет найден в HashMap, даже если он логически равен другому элементу.

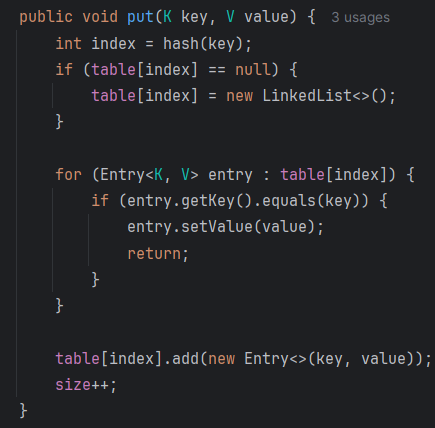
**Реализация задания лабораторной работы:**

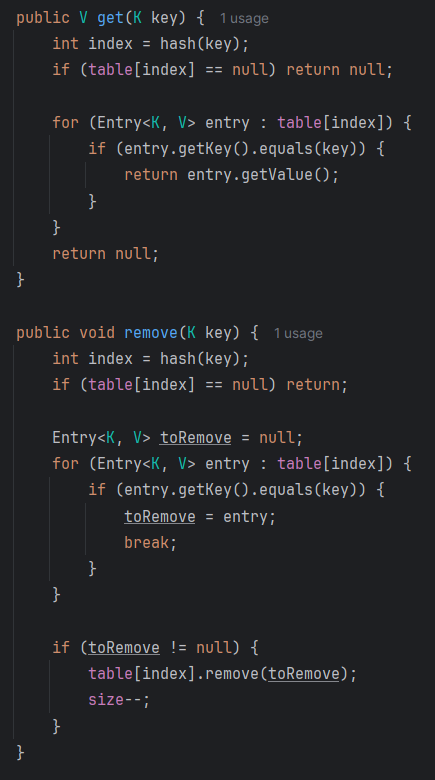
В рамках выполнения лабораторной работы была реализована система для демонстрации работы с хэш-таблицами. Первым шагом стал создание класса Entry, который представляет собой узел хэш-таблицы. Этот класс содержит пару "ключ-значение" и ссылку на следующий узел, что позволяет организовать цепочки для разрешения коллизий методом цепочек. Класс включает конструктор для инициализации полей и переопределение метода toString() для удобного вывода содержимого.

****

Далее был разработан основной класс HashTable, реализующий функционал хэш-таблицы. Класс содержит массив объектов Entry и предоставляет основные операции: добавление элемента (put), получение значения по ключу (get), удаление элемента (remove) и вывод всей таблицы (print). Особенностью реализации является использование метода цепочек для обработки коллизий - при совпадении хэш-кодов новые элементы добавляются в начало цепочки.

****

****

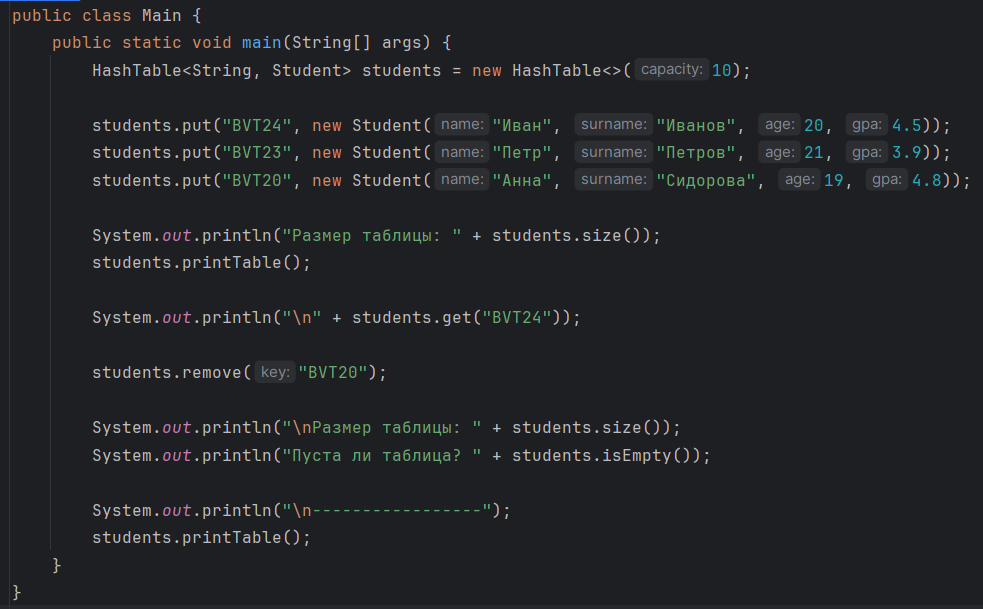
****

****

Для демонстрации работы хэш-таблицы был создан класс Student, содержащий информацию о студенте: имя, фамилия, возраст, номер студенческого билета и средний балл.

****

В демонстрационной программе была создана хэш-таблица, где ключами выступали номера студенческих билетов, а значениями - объекты класса Student. Были последовательно продемонстрированы все операции: добавление нескольких студентов, поиск по ключу, удаление элемента и вывод содержимого таблицы. Особое внимание было уделено проверке корректности работы переопределенных методов - при попытке добавления студента с уже существующим номером билета происходило обновление значения, что подтверждает правильность реализации механизма хэширования.

****

Отдельно проверялась обработка коллизий путем добавления элементов с разными ключами, но дающими одинаковый хэш-код. Все операции выполнялись корректно, что подтверждает правильность реализации структуры данных и соблюдение контракта между методами equals() и hashCode().

**Вывод:**

В ходе работы я освоил принципы работы хэш-таблиц и научился корректно переопределять методы equals() и hashCode(), соблюдая их контракт для обеспечения правильной работы с хэш-коллекциями.

<https://github.com/deylon999/ITAP/tree/master/3lab>