Java

Stack, Map

Комментарии по работам

- Можно присылать исправления
- Наименования
- Геттеры и сеттеры
- Примитивы и автобоксинг
- Гит

Наименования

- Классы - UpperCamelCase
- Параметры, поля, локальные переменные - lowerCamelCase
- Константы (static final) - SCREAMING_SNAKE
- Стоит давать переменным осмысленные имена
- Не принято использовать _ для обозначения не используемых параметров

Геттеры и сеттеры

- Нужны для скрытия реализации
- Генерируются IDEA
- Ключевое слово final

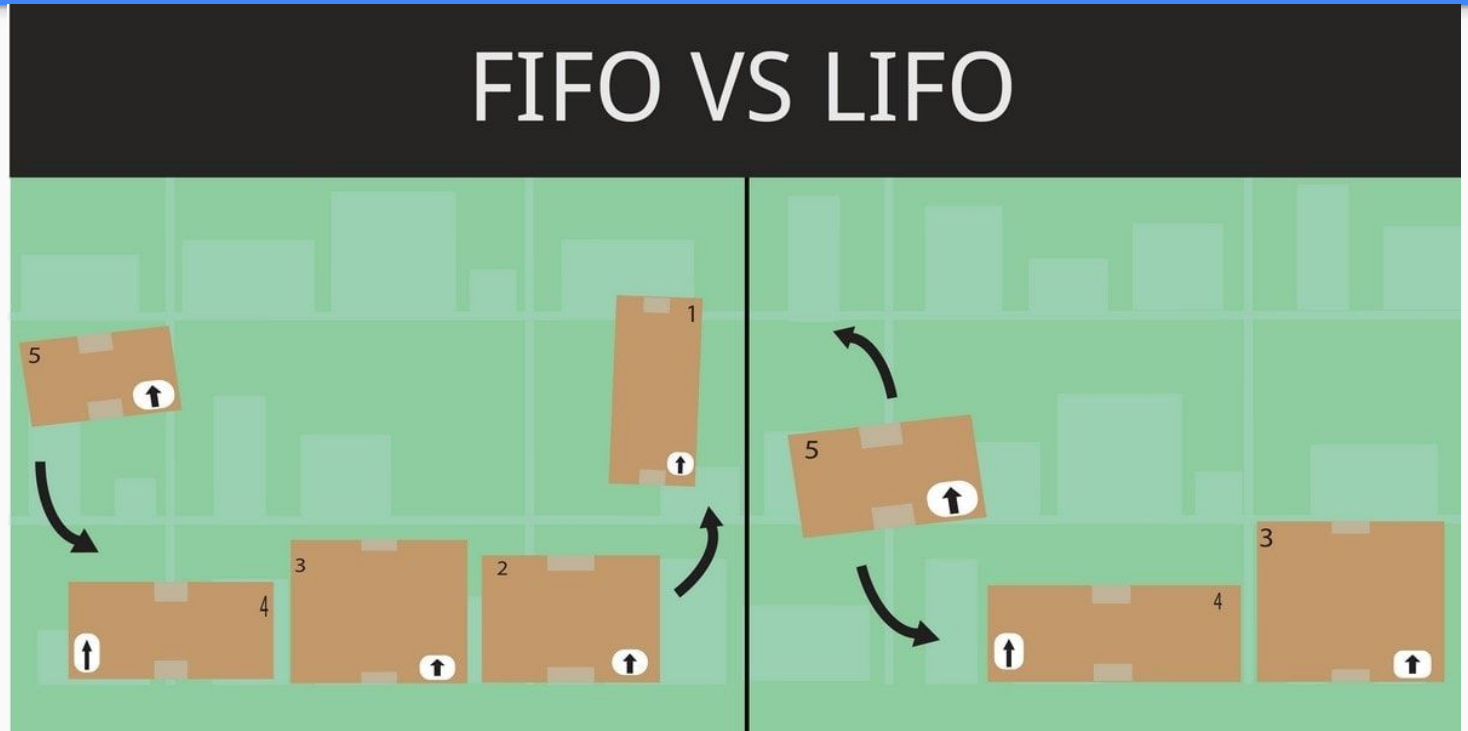
Примитивы, классы-обёртки, автобоксинг

- Класс-обёртка содержит:
  - значение примитивного типа
  - вспомогательные методы и поля
- Java обрабатывает автоматически
- Может быть null
- В обобщениях не может быть примитивного типа

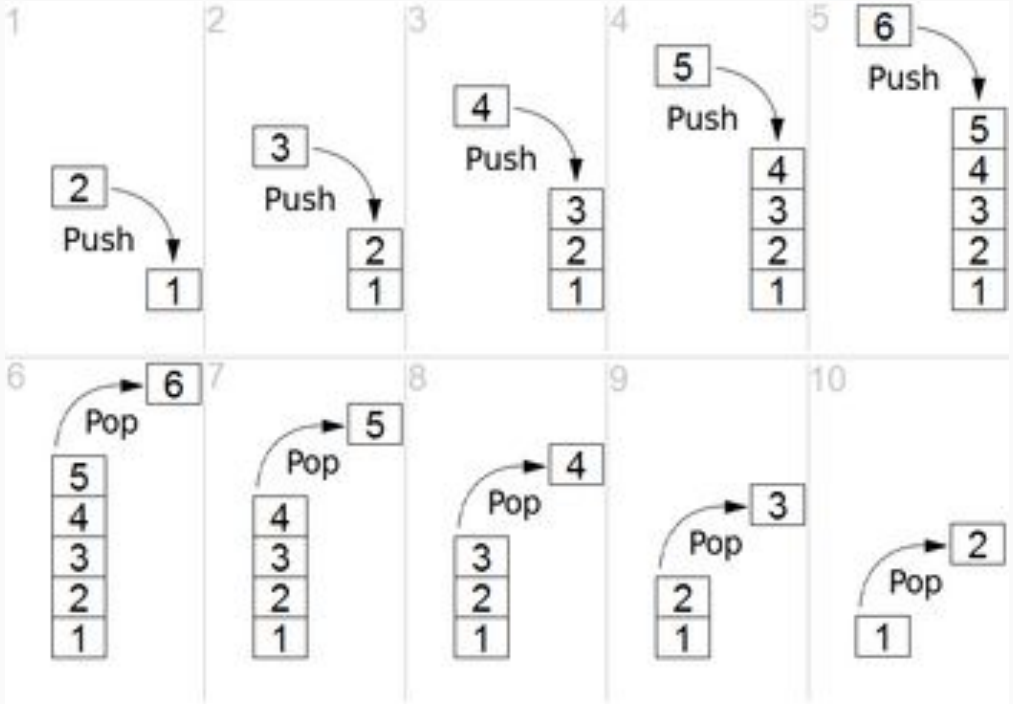
Гит

- Создать репозиторий
- Добавить README.MD
- Настроить .gitignore
- Создать ветку задания в IDEA
- Коммит и пуш

FIFO vs LIFO



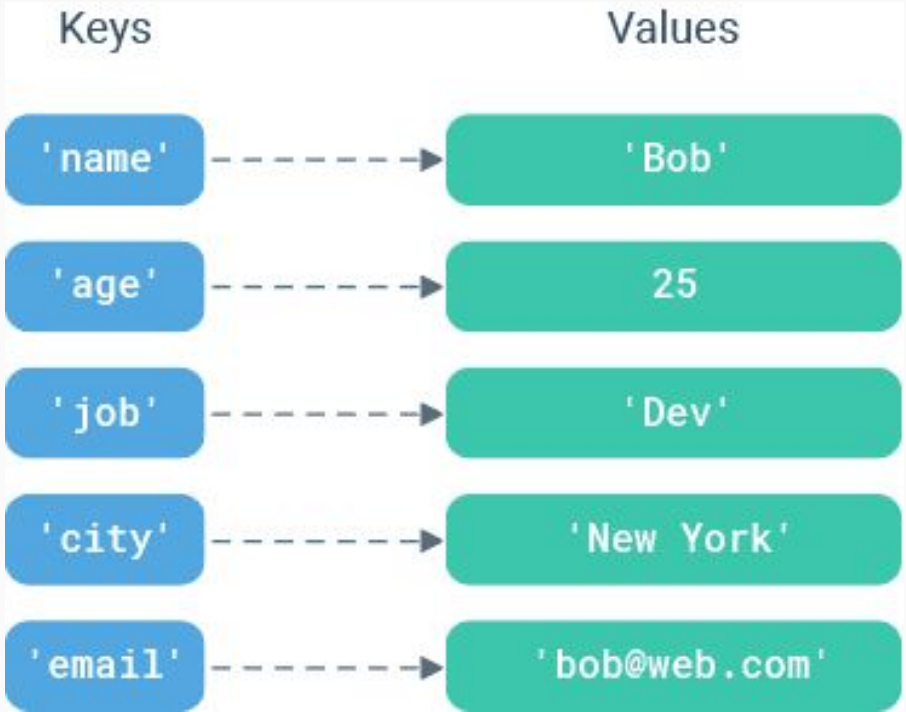
Stack


1
2
Push
1
2
3
Push
2
1
3
4
Push
3
2
1
4
5
Push
4
3
2
1
5
6
Push
5
4
3
2
1
6
6
Pop
5
4
3
2
1
7
5
Pop
4
3
2
1
8
4
Pop
3
2
1
9
3
Pop
2
1
10
2
Pop
1

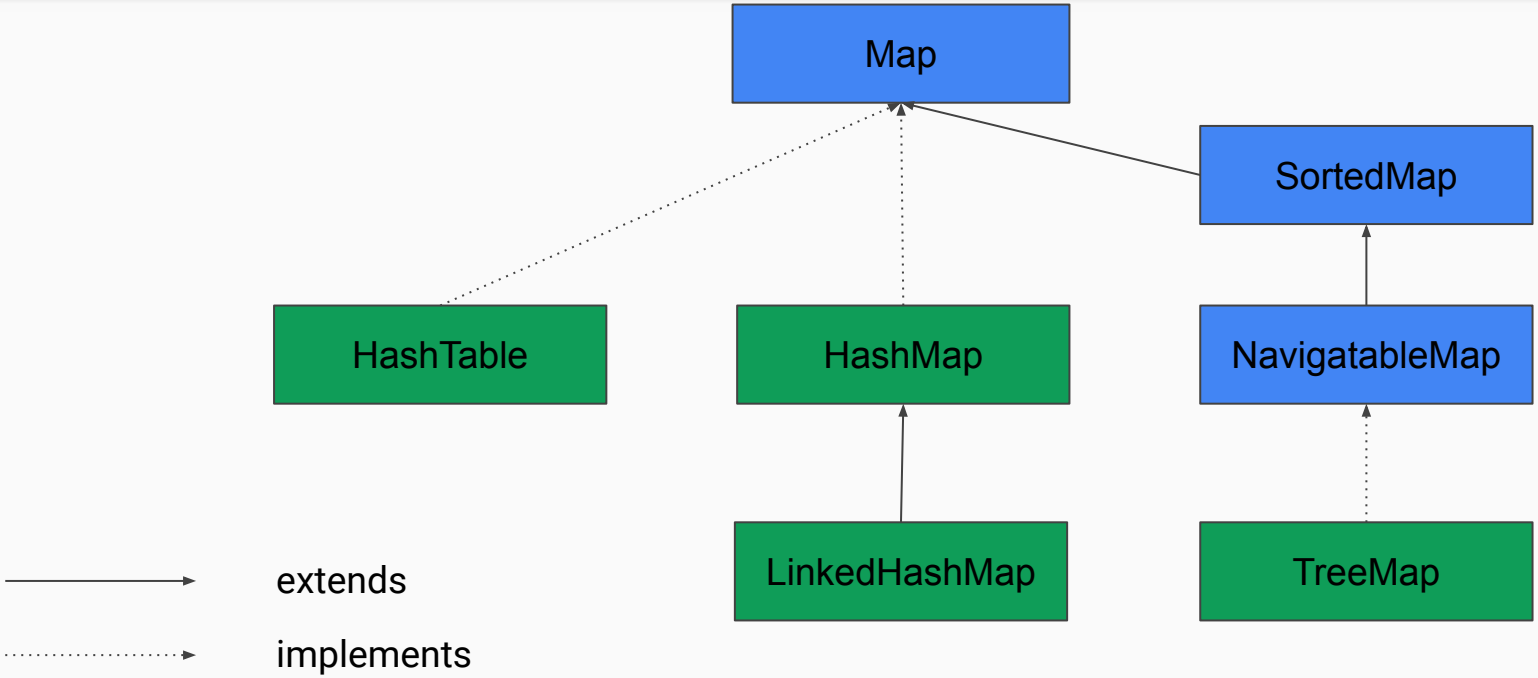
Queue

- peek(): T - получение объекта без удаления
- poll(): T - получение объекта с удалением

Map


Keys
Values
'name'
'Bob'
'age'
25
'job'
'Dev'
'city'
'New York'
'email'
'bob@web.com'

Map<K, V>

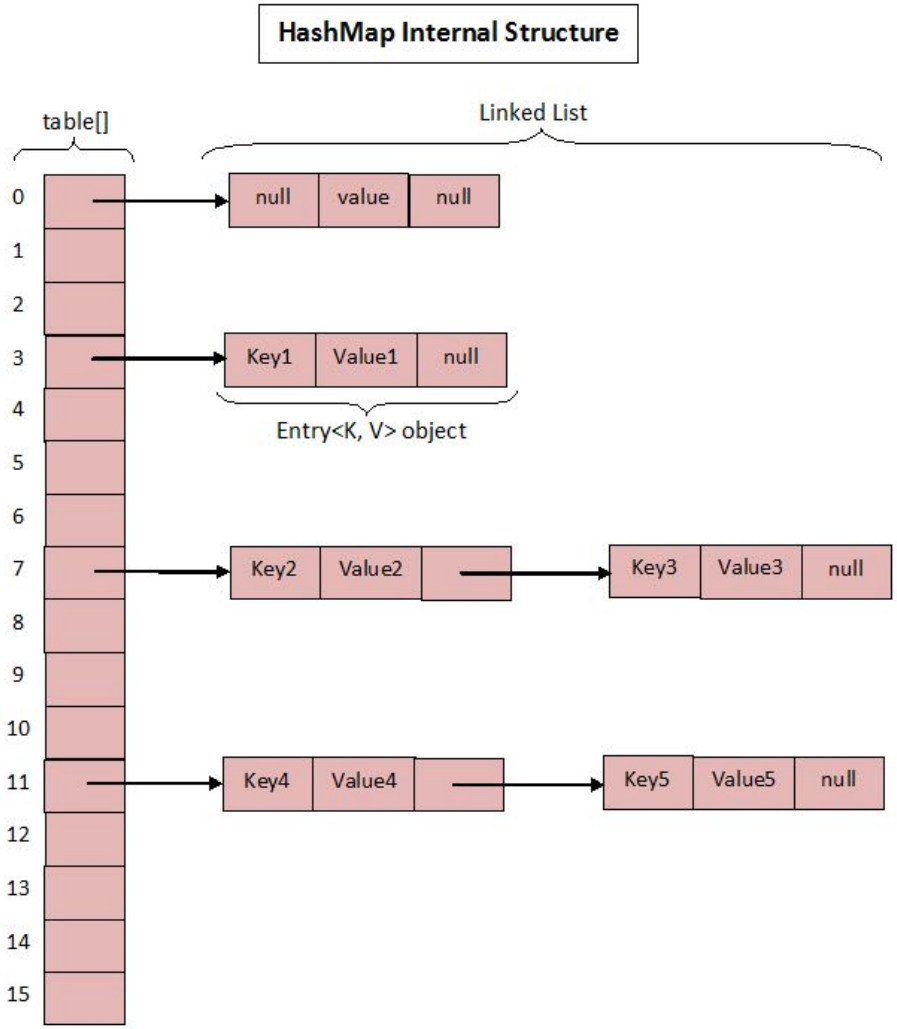

Map
SortedMap
HashTable
HashMap
NavigatableMap
LinkedHashMap
TreeMap
extends
implements

Интерфейс Map

- Не коллекция
- size() / isEmpty() / clear()
- get(Object o): V
- put(K, V): V
- remove(Object): V
- keySet(): Set<K>, values(): Collection<V>
- entrySet(): Set<Entry<K, V>>

HashMap

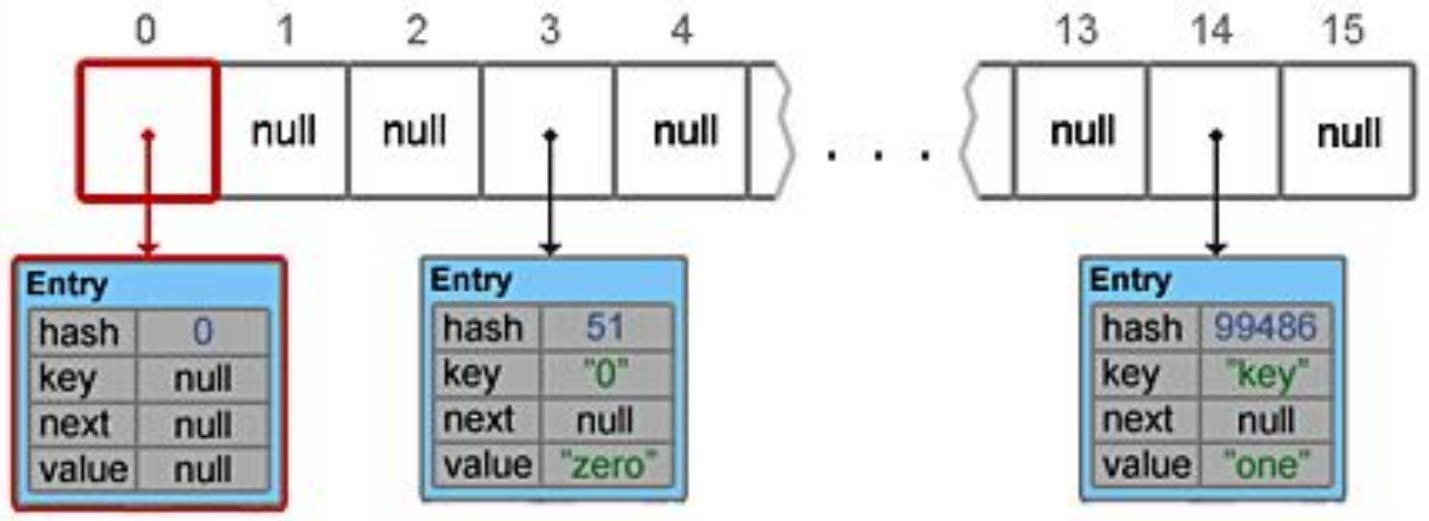
- table — Массив типа Entry[], который является хранилищем ссылок на списки (цепочки) значений;
- loadFactor — Коэффициент загрузки. Значение по умолчанию 0.75 является хорошим компромиссом между временем доступа и объемом хранимых данных;
- threshold — Предельное количество элементов, при достижении которого, размер хэш-таблицы увеличивается вдвое. Рассчитывается по формуле (capacity * loadFactor);
- size — Количество элементов HashMap-а;
- По умолчанию размер 16


HashMap Internal Structure
table[]
Linked List
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
null
value
null
Key1
Value1
null
Entry<K, V> object
Key2
Value2
Key3
Value3
null
Key4
Value4
Key5
Value5
null

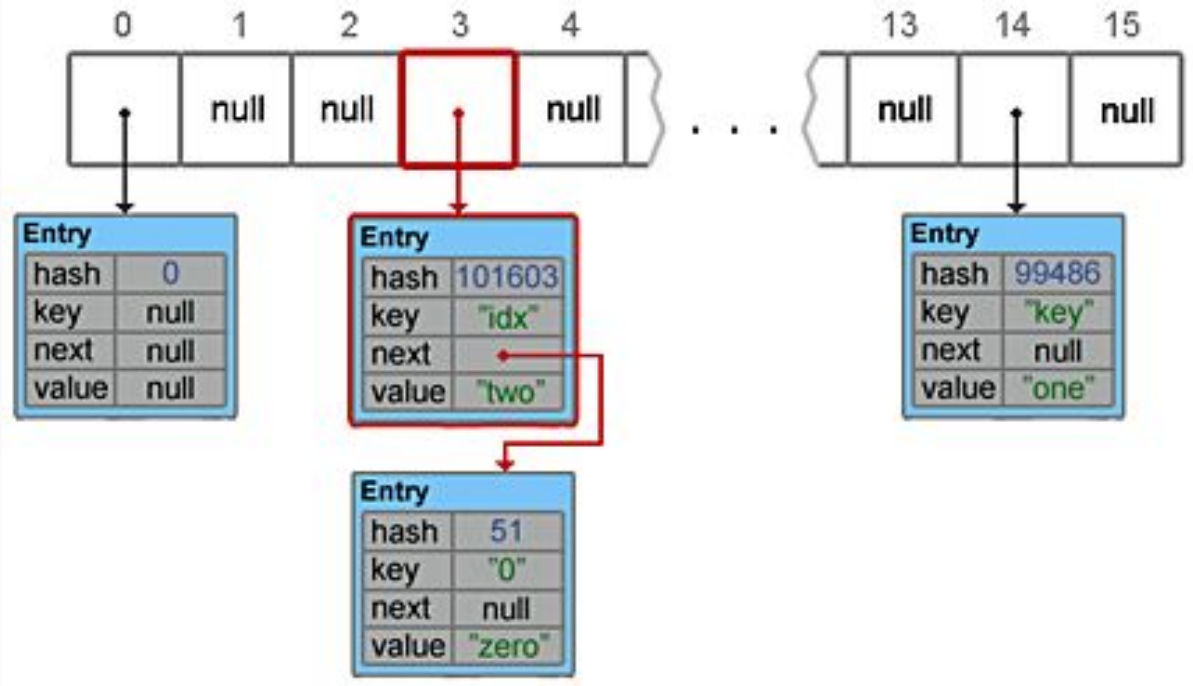
put(K, V)

1. Считается hashCode()
2. Внутри хранится bucket, который определяется hashCode()
3. index = hashCode() & (n-1), где n - число bucket
4. Если коллизия hashCode() то добавляется значение в связаный список
5. Хранит:
    a. hashCode
    b. Key
    c. Value
    d. Указатель на следующий элемент, если есть коллизия
6. Если ключ уже есть то значение перезапишется и вернётся старое

Вставка нового элемента



Вставка нового элемента



get()

1. hashCode()
2. Получаем индекс
3. Сравниваем ключи на equals()
4. Если не равны то смотрим есть ли следующий элемент
    a. Если есть следующий элемент то сравниваем его на equals …
    b. Если нет (null) то вернём null