# 8. Отображения (Map'ы) и их различные реализации

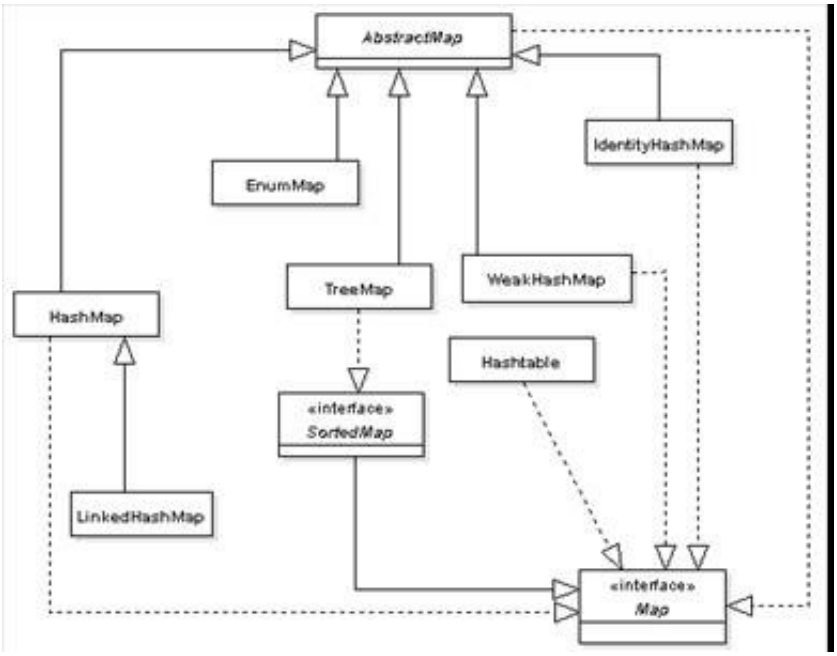
**Карта отображений** — это объект, который хранит пару “ключ-значение”. Поиск объекта (значения) облегчается по сравнению с множествами за счет того, что его можно найти по его уникальному ключу. Уникальность объектов-ключей должна обеспечиваться переопределением методов `hashCode()` и `equals()` пользовательским классом. Если элемент с указанным ключом отсутствует в карте, то возвращается значение `null`.

## Интерфейсы карт отображений

- `Map<K,V>` — отображает уникальные ключи и значения;
  - `void clear()`: очищает коллекцию
  - `boolean containsKey(Object k)`: возвращает true, если коллекция содержит ключ k
  - `boolean containsValue(Object v)`: возвращает true, если коллекция содержит значение v
  - `Set<Map.Entry<K, V>> entrySet()`: возвращает набор элементов коллекции. Все элементы представляют объект Map.Entry
  - `boolean equals(Object obj)`: возвращает true, если коллекция идентична коллекции, передаваемой через параметр obj
  - `boolean isEmpty`: возвращает true, если коллекция пуста
  - `V get(Object k)`: возвращает значение объекта, ключ которого равен k. Если такого элемента не окажется, то возвращается значение `null`
  - `V put(K k, V v)`: помещает в коллекцию новый объект с ключом k и значением v. Если в коллекции уже есть объект с подобным ключом, то он перезаписывается. После добавления возвращает предыдущее значение для ключа k, если он уже был в коллекции. Если же ключа еще не было в коллекции, то возвращается значение `null`
  - `Set<K> keySet()`: возвращает набор всех ключей отображения
  - `Collection<V> values()`: возвращает набор всех значений отображения
  - `void putAll(Map<? extends K, ? extends V> map)`: добавляет в коллекцию все объекты из отображения map
  - `V remove(Object k)`: удаляет объект с ключом k
  - `int size()`: возвращает количество элементов коллекции
- `Map.Entry<K,V>` — описывает пару “ключ-значение”;
- `SortedMap<K,V>` — содержит отсортированные ключи и значения;
- `NavigableMap<K,V>` — добавляет новые возможности поиска по ключу.

## Стандартные реализации карт отображений

- `AbstractMap<K,V>` — реализует интерфейс `Map<K,V>`;
- `HashMap<K,V>` — расширяет `AbstractMap<K,V>`, используя хэш-таблицу, в которой ключи отсортированы относительно значений их хэш-кодов;
- `TreeMap<K,V>` — расширяет `AbstractMap<K,V>`, используя дерево, где ключи расположены в виде дерева поиска в строгом порядке.
- `WeakHashMap<K,V>` — позволяет механизму сборки мусора удалять из карты значения по ключу, ссылка на который вышла из области видимости приложения.
- `LinkedHashMap<K,V>` — запоминает порядок добавления объектов в карту и образует при этом дважды связанный список ключей. Этот механизм эффективен, только если превышен коэффициент загруженности карты при работе с кэш-памятью и др.



Иерархия наследования карт

# Пример использования класса HashMap

```
import java.util.*;

public class CollectionApp {
    public static void main(String[] args) {
        Map<Integer, String> states = new HashMap<Integer, String>();
        states.put(1, "Германия");
        states.put(2, "Испания");
        states.put(4, "Франция");
        states.put(3, "Италия");

        // получим объект по ключу 2
        String first = states.get(2);
        System.out.println(first);

        // получим весь набор ключей
        Set<Integer> keys = states.keySet();
```

```
        // получить набор всех значений
        Collection<String> values = states.values();

        //заменить элемент
        states.replace(1, "Бельгия");

        // удаление элемента по ключу 2
        states.remove(2);

        // перебор элементов
        for (Map.Entry<Integer, String> item : states.entrySet()) {
        System.out.printf("Ключ: %d  Значение: %s \n", item.getKey(),
item.getValue());
        }

        Map<String, Person> people = new HashMap<String, Person>();
        people.put("1240i54", new Person("Tom"));
        people.put("1564i55", new Person("Bill"));
        people.put("4540i56", new Person("Nick"));

        for (Map.Entry<String, Person> item : people.entrySet()) {
        System.out.printf("Ключ: %s  Значение: %s \n", item.getKey(),
item.getValue().getName());
        }
    }
}

class Person {
    private String name;

    public Person(String value){
        name = value;
    }

    String getName(){
        return name;
    }
}
```