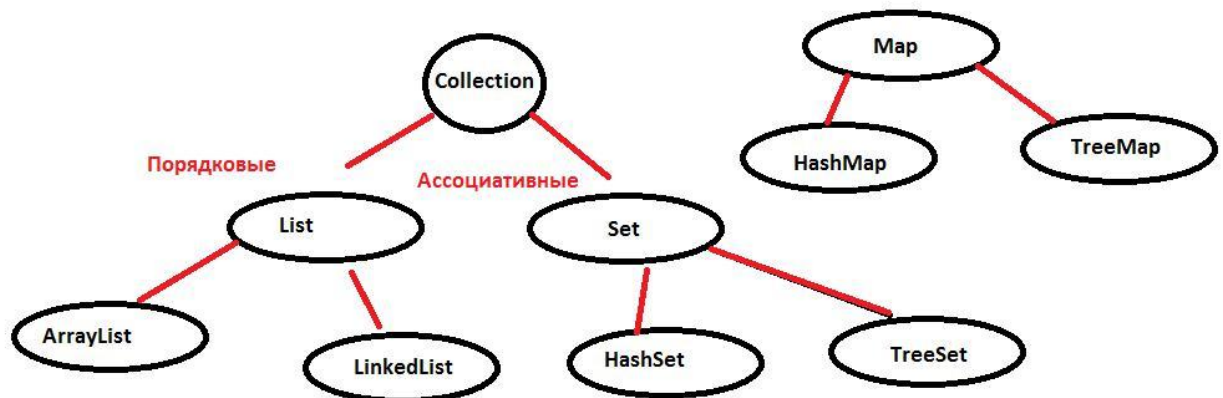


Коллекции

Общая схема



Коллекции в Джаве по сути тоже самое, что и контейнеры в Си++. Они созданы для эффективного хранения того, что вам нужно, чтобы обеспечить быстрый поиск или добавление и удаление элементов. То есть какую коллекцию выбирать вы делаете из соображений того, для чего она вам нужна. В отличие от Си++ в Джаве почти все коллекции являются наследниками одного интерфейса Collection. Собственно это нужно для того, чтобы можно было без изменения кода программы просто менять одну структуру данных на другую. Так как у многих коллекций есть общие методы и чтобы сменить коллекцию вы меняете всего лишь 1 слово в объявление.

Коллекции основанные на Map не относятся к общей структуре, так как в них заложена немного другая идеология, там ключ/значение, а это не связать никак с остальными контейнерами.

Основные общие методы Collection (они есть у всех коллекций, которые наследуются от этого интерфейса):

`int size()` – возвращает количество элементов

`Boolean contains (Object)` – проверяет есть ли объект в коллекции

`add (Object)` – добавление объекта в коллекцию

`remove (Object)` – удаление объекта из коллекции

`clear()` – очистка коллекции

`iterator()` – итератор для обхода коллекции

Итератор

Итератор – это абстрактный класс, позволяющий обойти сложную коллекцию не вникая в то, как там всё хранится.

Основные методы:

`next()` – возвращает текущий элемент и идёт на следующий

`hasNext()` – проверка на то, если ли дальше ещё элемент, если есть, то вернёт `true`, если это конец, то `false`

`remove()` – удаление текущего элемента, на котором стоит итератор

Интерфейс **Set** включает следующие методы :

К семейству интерфейса **Set** относятся *HashSet*, *TreeSet* . В множествах **Set** разные реализации используют разный порядок хранения элементов.

В **HashSet** порядок элементов оптимизирован для быстрого поиска.

TreeSet-это класс, содержащий набор свойств и методов для работы с уникальными отсортированными элементами с доступом по значению. Отсортировка происходит через компаратор.

Метод	Описание
<code>add(Object o)</code>	Добавление элемента в коллекцию, если он отсутствует. Возвращает <code>true</code> , если элемент добавлен.
<code>addAll(Collection c)</code>	Добавление элементов коллекции, если они отсутствуют.
<code>clear()</code>	Очистка коллекции.
<code>contains(Object o)</code>	Проверка присутствия элемента в наборе. Возвращает <code>true</code> , если элемент найден.
<code>containsAll(Collection c)</code>	Проверка присутствия коллекции в наборе. Возвращает <code>true</code> , если все элементы содержатся в наборе.
<code>equals(Object o)</code>	Проверка на равенство.
<code>hashCode()</code>	Получение <code>hashCode</code> набора.
<code>isEmpty()</code>	Проверка наличия элементов. Возвращает <code>true</code> если в коллекции нет ни одного элемента.
<code>iterator()</code>	Функция получения итератора коллекции.
<code>remove(Object o)</code>	Удаление элемента из набора.
<code>removeAll(Collection c)</code>	Удаление из набора всех элементов переданной коллекции.
<code>retainAll(Collection c)</code>	Удаление элементов, не принадлежащих переданной коллекции.
<code>size()</code>	Количество элементов коллекции
<code>toArray()</code>	Преобразование набора в массив элементов.
<code>toArray(T[] a)</code>	Преобразование набора в массив элементов. В отличие от предыдущего метода, который возвращает массив объектов типа <code>Object</code> , данный метод возвращает массив объектов типа, переданного в параметре.

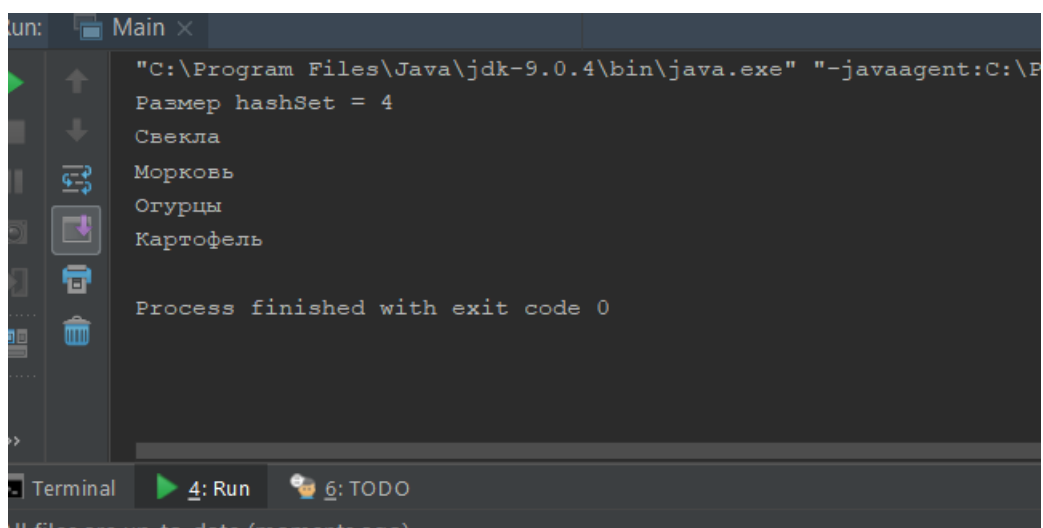
К семейству интерфейса **Set** относятся *HashSet*, *TreeSet* и *LinkedHashSet* (про него говорить не буду). В множествах **Set** разные реализации используют разный порядок хранения элементов. В *HashSet* порядок элементов оптимизирован для быстрого поиска. В контейнере *TreeSet* объекты хранятся отсортированными по возрастанию. *LinkedHashSet* хранит элементы в порядке добавления.

```
import java.util.Iterator;
import java.util.HashSet;
import java.util.Set;

public class Main {
    public static void main(String[] args) {
        Set<String> hashSet = new HashSet<String>();
        hashSet.add("Картофель");
        hashSet.add("Морковь");
        hashSet.add("Свекла");
        hashSet.add("Огурцы");
        hashSet.add("Картофель"); // Данная запись не должна попасть в набор

        // Вывести в консоль размер набора
        System.out.println("Размер hashSet = " + hashSet.size());

        // Вывести в консоль записи
        Iterator<String> itr = hashSet.iterator();
        while (itr.hasNext()) {
            System.out.println(itr.next().toString());
        }
    }
}
```



```
run: Main x
"C:\Program Files\Java\jdk-9.0.4\bin\java.exe" "-javaagent:C:\P
Размер hashSet = 4
Свекла
Морковь
Огурцы
Картофель

Process finished with exit code 0

Terminal 4: Run 6: TODO
All files are up to date (moments ago)
```