Java Collection Framework

|  |  |  |  |
| --- | --- | --- | --- |
| № урока: | 2 | Курс: | Java Professional |
|  | | |  |
| Средства обучения: | | | Компьютер с установленной IntelliJ IDEA. |

# Обзор, цель и назначение урока

Рассмотрение интерфейсов «List», «Set», «Map», «Queue», «Collection». А также классы такие как «ArrayList»,

«LinkedList», «PriorityQueue», «ArrayDeque», «TreeSet», «HashSet», «TreeMap», «HashMap»

**Изучив материал данного занятия, учащийся сможет:**

* Понимать какие коллекции и карты содержаний существуют.
* Разбираться, какую коллекцию стоит применять для той или иной ситуации.
* Использовать карты изображения по назначению.
* Применять множество методов коллекций и карт изображений.

# Содержание урока

1. Коллекции
2. Интерфейсы «List», «Set», «Map», «Queue», «Collection».
3. Классы «ArrayList», «LinkedList», «PriorityQueue», «ArrayDeque», «TreeSet», «HashSet», «TreeMap», «HashMap».
4. Основные методы вставка/удаление/добавление/изменение
5. Интерфейс Comparator, Comparable.

# Резюме

**Collection** - коллекция содержит набор объектов (элементов). Здесь определены основные методы для манипуляции с данными, такие как вставка (**add**, **addAll**), удаление (**remove**, **removeAll**, **clear**), поиск (**contains**).

В интерфейсе **Collection<E>** определены методы, которые работают на всех коллекциях:

* **boolean add (E obj)** – добавляет **obj** к вызывающей коллекции и возвращает **true**, если объект добавлен, и **false**, если **obj** уже элемент коллекции;
* **boolean** **addAll** **(Collection<? Extends E> c)** – добавляет все элементы коллекции к вызывающей коллекции;
* **void clear()** – удаляет все элементы из коллекции;
* **boolean contains (Object obj**) – возвращает **true**, если вызывающая коллекция содержит элемент **obj**;
* **boolean equals (Object obj)** – возвращает **true**, если соответствующие элементы коллекции эквивалентны;
* **boolean isEmpty ()** – возвращает **true**, если коллекция пуста;
* **Iterator<E> iterator ()** – извлекает итератор;
* **boolean remove (Object obj)** – удаляет **obj** из коллекции;
* **int size ()** – возвращает количество элементов в коллекции;
* **Object[ ] toArray ()** – копирует элементы коллекции в массив объектов;
* **<T> T[ ] toArray (T a[ ])** – копирует элементы коллекции в массив объектов определенного типа.

**List** – упорядоченная коллекция иногда называемая списком или последовательностью.

* Список может содержать повторяющиеся элементы.
* Разработчик имеет абсолютный контроль над тем, в каком месте списка будет вставлен каждый элемент.
* Пользователь может получить доступ к элементам списка по индексу.
* Все коллекции реализуют интерфейсы **Serializable, Cloneable** (кроме **WeakHashMap**).

**ArrayList** – автоматически расширяемый массив,**может изменять свой размер в процессе выполнения программы.**

Класс **LinkedList<E>** реализует интерфейс **Queue<E>** и **List<E>**.

* Является представителем двунаправленного списка, где каждый элемент структуры содержит указатели на предыдущий и следующий элементы.
* Итератор поддерживает обход в обе стороны.
* Реализует методы получения, удаления и вставки в начало, середину и конец списка.
* Позволяет добавлять любые элементы в том числе и **null**.
* Список может содержать повторяющиеся элементы.

**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 обеспечивается переопределением методов **hashCode()** и **equals()** пользовательским классом.
* Если элемент с указанным ключом отсутствует в карте, то возвращается значение **null**.

**HashMap** — основан на хэш-таблицах, реализует интерфейс **Map**. Ключи и значения могут быть любых типов, в том числе и **null**. Данная реализация не дает гарантий относительно порядка элементов с течением времени.

**LinkedHashMap** – расширяет класс **HashMap**. Он создает связный список элементов в карте, расположенных в том порядке, в котором они вставлялись. Это позволяет организовать перебор карты в порядке вставки.

**TreeMap** – расширяет класс **AbstractMap** и реализует интерфейс **NavigatebleMap**. Он создает коллекцию, которая для хранения элементов применяет дерево. Объекты сохраняются в отсортированном порядке по возрастанию. Время доступа и извлечения элементов достаточно мало, что делает класс **TreeMap** блестящим выбором для хранения больших объемов отсортированной информации, которая должна быть быстро найдена.

**WeakHashMap** – коллекция, использующая слабые ссылки для ключей (а не значений).

**Слабая** **ссылка** (англ. **weak** **reference**) — специфический вид ссылок на динамически создаваемые объекты в системах со сборкой мусора. Отличается от обычных ссылок тем, что не учитывается сборщиком мусора при выявлении объектов, подлежащих удалению. Ссылки, не являющиеся слабыми, также иногда именуют «сильными».

# Закрепление материала

* Что такое Collection?
* Что такое Map?
* Какие интерфейсы Collection вы знаете?
* Какие интерфейсы Map вы знаете?
* Какие классы Collection вы знаете?
* Какие классы Map вы знаете?
* Какие методы в Collection/Map вы знаете?
* В каких ситуациях стоит использовать ArrayList/LinkedList?

# Дополнительное задание

**Задание**

Вводить с клавиатуры строки, пока пользователь не введёт строку 'end':

1. Создать список строк.
2. Ввести строки с клавиатуры и добавить их в список.
3. Вводить с клавиатуры строки, пока пользователь не введёт строку "end". "end" не учитывать.
4. Вывести строки на экран, каждую с новой строки.

# Самостоятельная деятельность учащегося

Задание 1

В любой из профильных **книг (Хорстман, Эккель)** найти соответствующие темы и закрепить материал. Использование **YouTube, Quizful** приветствуется.

Задание 2

Используя коллекцию удвойте слово:

1. Введите с клавиатуры 5 слов в список строк.
2. Метод doubleValues должен удваивать слова по принципу a,b,c -> a,a,b,b,c,c.
3. Используя цикл for выведите результат на экран, каждое значение с новой строки.

Задание 3

Минимальное из N чисел(использовать LinkedList):

1. Ввести с клавиатуры число N.
2. Считать N целых чисел и заполнить ими список - метод getIntegerList.
3. Найти минимальное число среди элементов списка - метод getMinimum.

Задание 4

Программа определяет, какая семья (фамилия) живёт в городе:

Пример ввода:

* Москва
* Ивановы
* Киев
* Петровы
* Лондон
* Абрамовичи
* **Лондон**

Пример вывода:

* **Абрамовичи**

# Рекомендуемые ресурсы

Oracle: Коллекции

<https://docs.oracle.com/javase/tutorial/collections/interfaces/index.html>

Oracle: Карты изображения

<https://docs.oracle.com/javase/tutorial/collections/interfaces/map.html>