Изучить материал, сделать конспект лекции в тетрадь. Фото конспекта отправить на почту nazarovaolga.v@mail.ru до 17:00.
Образец: ИВАНОВ ИВАН группа 21-18 лекция от 17 апреля 2020 г.

Лекция на 17 апреля 2020 г.

Коллекции

Базовый функционал

Для работы с коллекциями разработчики Java создали специальный **Collection Framework**. Вся система **Collection Framework** может быть разделена на три составляющих:

- Набор базовых интерфейсов для нескольких типов коллекций
- Набор классов для реализации базовых интерфейсов с разными «потребительскими» характеристиками
- Набор алгоритмов для работы с коллекциями

Базовые интерфейсы

Список имеет свои особенности, множество — свои, очередь — свои. Набор методов для списка и для множества будет различаться, т.к. эти типы коллекций (список и множество) имеют некоторые важные отличия. Рассматривайте их как специализированные инструменты — например, для закручивания шурупов нужен шуруповерт, для бетонных стен — перфоратор и т.д., что они все имеют «одну природу», но каждый имеет некоторую специализацию:

- **java.util.Collection** — основной интерфейс, который описывает базовые методы, которыми должна обладать любая коллекция. Т.е. если какой-то класс претендует на звание КОЛЛЕКЦИЯ — он должен реализовать те методы, которые описаны в этом интерфейсе. Проводя аналогию с нашим набором сверлильных инструментов — интерфейс **java.util.Collection** их общий родитель — у него есть возможнсть сверлить. Советую зайти на сайт с документацией и честно просмотреть все его методы. Возможно, что Java версии 8 (и выше) покажется вам сложноватой, поэтому для начала советую зайти на документацию по Java версии 7. java.util.Collection. Большая часть методов говорит сама за себя, так что почитайте.
- **java.util.List** — интерфейс для операций с коллекцией, которая является списком. Список обладает следующими важными признаками:
    1. Список может включать одинаковые элементы

2. Элементы в списке хранятся в том порядке, в котором они туда помещались. Самопроизвольных перемещений элементов в нем не происходит — только с вашего ведома. Например, вы можете добавить элемент на какую-то позицию и тогда произойдет сдвиг других элементов.
3. Можно получить доступ к любому элементу по его порядковому номеру/индексу внутри списка

- Т.е. если вам требуется, чтобы коллекция обладала такими свойствами — выбирайте класс, который реализует интерфейс **java.util.List**
- **java.util.Set** — интерфейс для хранения множества. В отличии от **java.util.List** этот интерфейс как раз не может иметь одинаковые элементы (смотрим методы **equals** и **hashCode в гугл кому интересно, а лучше всем самостоятельно изучить будет полезно**) и порядок хранения элементов в множестве может меняться при добавлении/удалении/изменении элемента. Может возникнуть вопрос, зачем такая коллекция нужна — это удобно в случае, когда вы создаете набор уникальных элементов из какой-то группы элементов
- **java.util.SortedSet** — это наследник интерфейса **java.util.Set** и его дополнительным функционалом является автоматическое выстраивание элементов внутри множества по порядку. Как этот порядок настаивается, мы поговорим позже.
- **java.util.Queue** — интерфейс предлагает работать с коллекцией как с очередью, т.е. коллекция имеет метод для добавления элементов в один конец и метод для получения элемента с другого конца — т.е. настоящая очередь по принципу FIFO — First In First Out — если первым пришел, то первым и уйдешь. Для широкого круга задач такая конструкция работы с коллекцией бывает достаточно удобной структурой.
- **java.util.Map** — очень удобная конструкция, которая хранит данные не в виде списка значений, а в виде пары ключ-значение. Это очень востребованная форма, в которой вы получаете доступ к значению в коллекции по ключу. Например, доступ к данным пользователя на сайте может быть осуществлен по логину (по email например). Самих данных может быть достаточно много, но для поиска можно использовать очень короткую строку-ключ.

И еще раз скажу самое важное — коллекция позволяет вам работать с группой объектов и специализация коллекции определяется требованиями к самим данным и к тем операциям, которые нужно использовать при работе с данными.

Простой пример использования коллекций

Предлагаю посмотреть пример (демонстрацию) использования основных методов интерфейса java.util.Collection.

```
import java.util.ArrayList;
import java.util.Collection;
import java.util.Iterator;


public class ExampleCollection
{
    public static void main(String[] args) {
        // Создаем коллекции для демонстрации
        Collection col1 = createFirstCollection();
        Collection col2 = createSecondCollection();

        // Демонстрация прохода по коллекции
        System.out.println("============== Проход по коллекции");
        for(Object o : col1) {
            System.out.println("Item:" + o);
        }

        System.out.println();
        // Демонстрация прохода по коллекции через итератор
        System.out.println("============== Проход по коллекции через итератор");
        for (Iterator it = col1.iterator(); it.hasNext(); ) {
            String s = (String)it.next();
            System.out.println("Item:" + s);
        }
        System.out.println();

        // Демонстрации групповых операций
        System.out.println();
        System.out.println("============== Групповые операции");
        // Можно проверить сожержаться ли ВСЕ элементы col2 в col
        if(col1.containsAll(col2)) {
            System.out.println("Коллекция col содержит все от col2");
        }

        System.out.println("============== Добавление всех элементов в col1 из col2");
        // Можно добавить элементы из col2 в col1
        col1.addAll(col2);
        for(Object o : col1) {
            System.out.println("Item:" + o);
        }

        System.out.println("============== Удаление всех элементов col2, которые есть в col1");
        // Можно удалить ВСЕ элементы col2, которые есть в col1
        col1.removeAll(col2);
        for(Object o : col1) {
            System.out.println("Item:" + o);
        }

        // Пересоздаем коллекции для дпальнейшей демонстрации
        col1 = createFirstCollection();
        col2 = createSecondCollection();
        System.out.println("============== Удаление элементов из col1, которых нет в col2");
        col1.retainAll(col2);
        for(Object o : col1) {
            System.out.println("Item:" + o);
        }
```

```
        System.out.println("============== Очистка коллекции - не будет элементов");
        col1.clear();
        for(Object o : col1) {
            System.out.println("Item:" + o);
        }
        System.out.println();

        // Удаление элемента коллекции
        // Снова создаем коллекцию для демонстрации
        col1 = createFirstCollection();
        // Удаляем один элемент
        col1.remove("1");
        System.out.println("============== Удаляем элемент '1' - его не будет в списке");
        for(Object o : col1) {
            System.out.println("Item:" + o);
        }

        // Удаление коллекции через итератор
        // Снова создаем коллекцию для демонстрации
        col1 = createFirstCollection();
        System.out.println("============== Удаление через итератор");
        while(!col1.isEmpty()) {
            Iterator it = col1.iterator();
            Object o = it.next();
            System.out.println("Удаляем:" + o);
            // Удаляем элемент
            it.remove();
        }
    }

    // Первая коллекция для примера
    private static Collection createFirstCollection() {
        // Создать коллекцию на основе стандартного класса ArrayList
        Collection col = new ArrayList();
        // Добавление в коллекцию
        col.add("1");
        col.add("2");
        col.add("3");
        col.add("4");
        col.add("5");
        col.add("6");
        col.add("7");
        return col;
    }

    // Вторая коллекция для примера
    private static Collection createSecondCollection() {
        // Создать коллекцию на основе стандартного класса ArrayList
        Collection col2 = new ArrayList();
        col2.add("1");
        col2.add("2");
        col2.add("3");
        return col2;
    }
}
```

В принципе все достаточно просто — есть возможность добавлять в коллекцию элемент, есть возможность его удалять, есть возможность пройти по всему списку элементов и некоторые другие операции. Давайте смотреть

маленькими кусочками и делать комментарии. Для начала рассмотрим два метода, где мы создаем коллекции.

```
// Первая коллекция для примера
private static Collection createFirstCollection() {
    // Создать коллекцию на основе стандартного класса ArrayList
    Collection col = new ArrayList();
    // Добавление в коллекцию
    col.add("1");
    col.add("2");
    col.add("3");
    col.add("4");
    col.add("5");
    col.add("6");
    col.add("7");
    return col;
}

// Вторая коллекция для примера
private static Collection createSecondCollection() {
    // Создать коллекцию на основе стандартного класса ArrayList
    Collection col2 = new ArrayList();
    col2.add("1");
    col2.add("2");
    col2.add("3");
    return col2;
}
```

Как видите— берем нужный класс и создаем его экземпляр.
Я для примера взяла java.util.ArrayList. Т.к. этот класс реализует (имплементирует) интерфейс java.util.Collection, то у него есть все методы, которые в интерфейсе описаны. Добавление в коллекцию происходит очень просто — вызываем метод add. Вызывали — теперь ваш объект уже в коллекции. Добавляйте сколько угодно строк или другие объекты.
В коллекцию можно поместить объект любого класса, но нельзя туда поместить элементарный тип — int, char, long.

Еще немного о методах и библиотеках.
java.util.Iterator — это интерфейс, который позволяет перемещаться по списку элементов. При вызове метода iterator() вы получаете указатель на начало коллекции, но — ВНИМАНИЕ — не на первый элемент. Метод итератора hasNext() возвращает true в случае, если итератор может переместиться к следующему элементу (есть следующий за текущим), если получаем false — значит элементов больше нет.
Метод итератора next() перемещается на следующий элемент и возвращает его значение — объект типа . Еще раз — это объект типа Object. Обратите внимание — я здесь специально продемонстрировал (как я называю «жесткое») приведение типа — т.к. я знаю, что в коллекции находятся объекты типа String, то я преобразую ссылку на объект типа Object на ссылку на объект типа String.

Д/з Изучить самостоятельно типы данных Queue, Deque, Stack, их методы, способы инициализации и способы применения, стандартные алгоритмы работы.