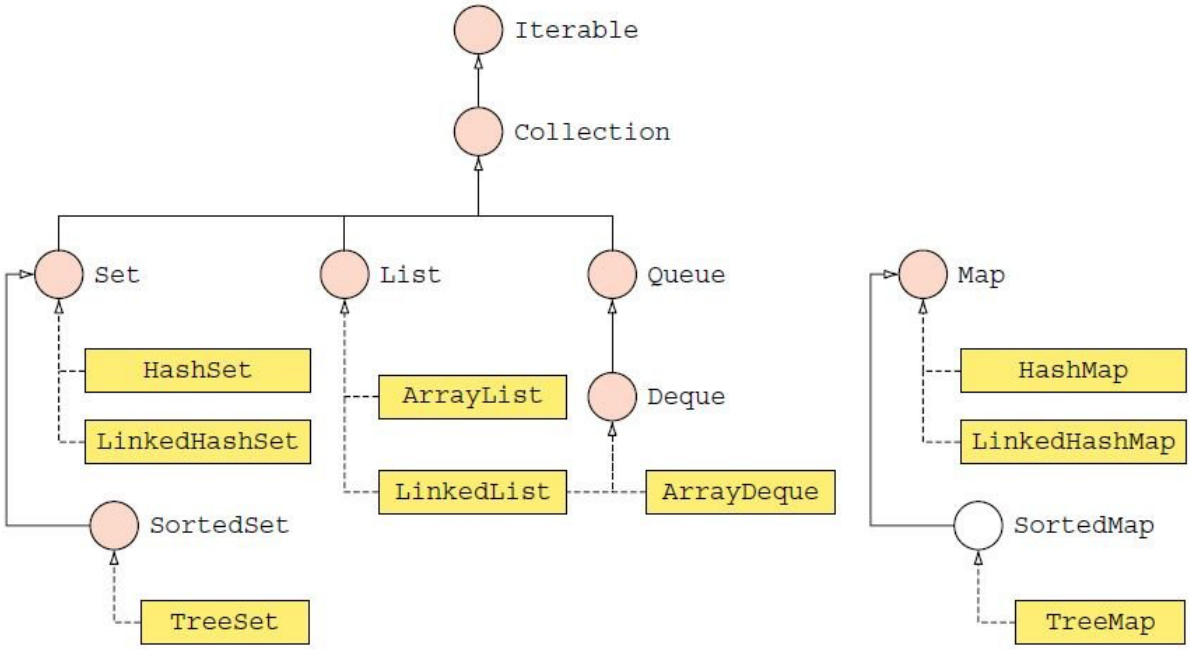
Лабораторная работа №5

Коллекции

Цель работы: изучение структуры Java Collections Framework и способов работы с ним.

Java Collections Framework

Коллекции — это наборы однородных элементов. Инструменты для работы с такими структурами в Java содержатся в Java Collections Framework. Фреймворк состоит из интерфейсов, их реализаций и утилитарных классов для работы со списками: сортировки, поиска, преобразования.



`Set` — это неупорядоченное множество уникальных элементов.

`List` — упорядоченный список, в котором у каждого элемента есть индекс. Дубликаты значений допускаются.

Например, последовательность букв в слове: буквы могут повторяться, при этом их порядок важен.

`Queue` — очередь. В таком списке элементы можно добавлять только в хвост, а удалять — только из начала. Так реализуется концепция FIFO (first in, first out) — «первым пришёл — первым ушёл».

`Deque` может выступать и как очередь, и как стек. Это значит, что элементы можно добавлять как в её начало, так и в конец. То же относится к удалению.

`Map` состоит из пар «ключ-значение». Ключи уникальны, а значения могут повторяться. Порядок элементов не гарантирован. `Map` позволяет искать объекты (значения) по ключу.

Не путайте интерфейс `Collection` и фреймворк `Collections`. `Map` не наследуется от интерфейса `Collection`, но входит в состав фреймворка `Collections`.

Реализации `List`

Класс `ArrayList` подойдёт в большинстве случаев, если нужен именно список.

Строится на базе обычного массива. Если при создании не указать размерность, то под значения выделяется 10 ячеек. При попытке добавить элемент, для которого места уже нет, массив автоматически расширяется — программисту об этом специально заботиться не нужно.

Список проиндексирован. При включении нового элемента в его середину все элементы с большим индексом сдвигаются вправо. При удалении элемента все остальные с большим индексом сдвигаются влево.

Класс `LinkedList` реализует одновременно `List` и `Deque`. Это список, в котором у каждого элемента есть ссылка на предыдущий и следующий элементы

Благодаря этому добавление и удаление элементов выполняется быстро — времязатраты не зависят от размера списка, так как элементы при этих операциях не сдвигаются: просто перестраиваются ссылки.

Если добавлять и удалять элементы с произвольными индексами в списке нужно чаще, чем итерироваться по нему, то лучше `LinkedList`. В остальных случаях — `ArrayList`.

При добавлении элементов в `ArrayList` (или их удалении) вызывается нативный метод `System.arraycopy`. В нём используются ассемблерные инструкции для копирования блоков памяти. Так что даже для больших массивов эти операции выполняются за приемлемое время.

Реализации `Queue`

Класс `ArrayDeque` — это реализация двунаправленной очереди в виде массива с переменным числом элементов.

Новые значения можно добавлять в начало или конец списка, и удалять оттуда же. Причём эти операции выполняются быстрее, чем при использовании `LinkedList`.

Класс `PriorityQueue` — упорядоченная очередь. По умолчанию элементы добавляются в естественном порядке: числа по возрастанию, строки по алфавиту и так далее, либо алгоритм сравнения задаёт разработчик.

Этот класс может быть полезен, например, для нахождения *n* минимальных чисел в большом неупорядоченном списке:

Реализации `Set`

Класс `HashSet` использует для хранения данных в хеш-таблице. Это значит, что при манипуляциях с элементами используется хеш-функция — `hashCode()` в Java.

Добавление, поиск и удаление элементов при такой организации происходит за постоянное время, независимо от числа элементов в коллекции.

Прежде чем добавить новый элемент в множество, вычисляется его `hashCode()` — чтобы определить бакет, куда он может быть помещён.

Если бакет пуст, элемент будет добавлен. Иначе уже добавленные элементы с таким же значением хеша сравниваются с кандидатом при помощи метода equals(). Если дубликат не найден, новый элемент становится частью множества. Он попадёт в тот же бакет.

О классе `TreeSet` вспоминают в тех случаях, когда множество должно быть упорядочено. Каким образом упорядочивать — определяет разработчик при создании нового `TreeSet`. По умолчанию элементы располагаются в естественном порядке.

Реализации `Map`

Класс `HashMap` хранит данные в виде хеш-таблицы, как и `HashSet`. Более того, `HashSet` внутри использует `HashMap`. При этом ключом выступает сам элемент.

Класс `TreeMap` строится тоже на базе красно-чёрного дерева. Элементы здесь упорядочены (в естественном или заданном при создании порядке) в каждый момент времени. При этом вставка и удаление более затратны, чем в случае с `HashMap`.

Класс `LinkedHashMap` расширяет возможности `HashMap` тем, что позволяет итерироваться по элементам в порядке их добавления. Как и в `LinkedList`, здесь каждая пара-значение содержит ссылку на предыдущий и последующий элементы.

Задание на лабораторную работу:

1. **Выполните данный код и проанализируйте результат.**

```
public static void main (String[] args){
       ArrayList<String> list = new ArrayList<>();
       list.add("test1");
       list.add("test2");
       list.add("test3");
       System.out.print(list.get(1)+":");
       list.add(1, "test4");
       System.out.print(list.get(1)+":");
       for (int i=0; i< list.size(); i++){
           System.out.print(list.get(i)+":");
       }
   }
```

2. **Дайте объяснение полученному результату**

```
public static void main(String args[]) {
       ArrayList<String> list = new ArrayList<>();
       String t1 = "test1";
       String t2 = "test2";
```

```
        list.add(t1);
        list.add(t2);
        System.out.print(list.size() + ":");
        t1 = "test3";
        list.remove(t1);
        System.out.print(list.size());
    }
```

3. Выполните данный код.

```
class Person {
   String name;
   Person(String name) { this.name = name; }
   public String toString() { return name; }
}

class TestHashSet {
   public static void main(String args[]) {
        HashSet<Person> set = new HashSet<>();
        Person p1 = new Person("Иван");
        Person p2 = new Person("Мария");
        Person p3 = new Person("Пётр");
        Person p4 = new Person("Мария");
        set.add(p1);
        set.add(p2);
        set.add(p3);
        set.add(p4);
        System.out.print(set.size());
   }
}
```

Какой результат рассчитывали получить? Почему его не получили? Как можно модифицировать данный код, чтобы получить нужны результат?

4. Сравнение скорости работы

Напишите метод, который добавляет 1000000 элементов в `ArrayList` и `LinkedList`. Напишите еще один метод, который выбирает из заполненного списка элемент наугад 100000 раз. Замерьте время, которое потрачено на это. Сравните результаты и предположите, почему они именно такие.

5. Коллекция элементов

Создать класс `Student`, содержащий следующие характеристики – имя, группа, курс, оценки по предметам (не менее 8 предметов).

Создать коллекцию, содержащую объекты класса `Student` (не менее 20). Инициализацию списка производить из текстового файла.

Написать метод, который удаляет студентов со средним баллом <3.

Если средний балл>=3, студент переводится на следующий курс.

Напишите метод

`List<Student> getStudents(List<Student> students, int course),`

который получает список студентов и номер курса. Метод возвращает список тех студентов, которые обучаются на данном курсе.

Напишите метод, который выводит в отформатированном виде информацию о студентах из переданного списка `void printStudents(List<Student> students)`. Выводить на экран используя следующий порядок сортировки: курс, группа, имя по алфавиту.

6. Коллекция элементов (по варианту)

Создайте класс сущности по варианту.

Создать коллекцию, содержащую объекты разработанного класса (не менее 20). Инициализацию списка производить из текстового файла.

Реализуйте метод в соответствии с заданием по варианту.

Напишите метод, который возвращает в отформатированном виде информацию о объекте из переданного списка `String printSorted(List<T> entity)`. Формировать строку используя в сортированном порядке (порядок сортровки выбрать самостоятельно исходя из хранимой сущности).

Содержание отчета:

1. Титульный лист
2. Модифицированный код задания 4.
3. Программный код задания 5.

Варианты заданий

Вариант	**Задание**
1 «Человек»	*фамилия; имя; отчество; пол; национальность; рост; вес; дата рождения (год, месяц число); номер телефона; домашний адрес (почтовый индекс, страна, область, район, город, улица, дом, квартира).* Вывести сведения о самом молодом человеке.

2 «Школьник»	*фамилия; имя; отчество; пол; национальность; рост; вес; дата рождения (год, месяц число); номер телефона; домашний адрес (почтовый индекс, страна, область, район, город, улица, дом, квартира); школа; класс.* Вывести сведения про всех учеников пятых классов.
3 «Студент»	*фамилия; имя; отчество; пол; национальность; рост; вес; дата рождения (год, месяц число); номер телефона; домашний адрес (почтовый индекс, страна, область, район, город, улица, дом, квартира); ВУЗ; курс; группа; средний бал; специальность.* Вывести сведения про всех студентов у которых средний балл ниже 70 баллов.
4 «Покупатель»	*фамилия; имя; отчество; пол; национальность; рост; вес; дата рождения (год, месяц число); номер телефона; домашний адрес (почтовый индекс, страна, область, район, город, улица, дом, квартира); номер кредитной карточки; банковского счета.* Вывести данные о покупателях с города Муром.
5 «Пациент»	*фамилия; имя; отчество; пол; национальность; рост; вес; дата рождения (год, месяц число); номер телефона; домашний адрес (почтовый индекс, страна, область, район, город, улица, дом, квартира); номер больницы; отделение; номер медицинской карты; диагноз; группа крови.* Вывести данные про пациентов с 18 отделения.
6 «Владелец автомобиля»	*фамилия; имя; отчество; номер телефона; домашний адрес (почтовый индекс, страна, область, район, город, улица, дом, квартира) марка автомобиля; номер автомобиля; номер техпаспорта.* Вывести данные про автомобили марки ”Ваз”.

7 «Военнослужащий»	*фамилия; имя; отчество; домашний адрес (почтовый индекс, страна, область, район, город, улица, дом, квартира); национальность; дата рождения (год, месяц число); должность; звание.* Вывести данные про военнослужащих в звании “лейтенант”.
8 «Рабочий»	*фамилия; имя; отчество; домашний адрес (почтовый индекс, страна, область, район, город, улица, дом, квартира); национальность; дата рождения (год, месяц число); No цеха; табельный номер; образование; год поступления на работу.* Вывести данные про рабочих, поступивших на работу в 2010 году.
9 «Владелец телефона»	*фамилия; имя; отчество; домашний адрес (почтовый индекс, страна, область, район, город, улица, дом, квартира); No телефона.* Вывести данные про владельцев телефона номер, которого начинается на 720.
10 «Абитуриент»	*фамилия; имя; отчество; пол; национальность; дата рождения (год, месяц число); домашний адрес (почтовый индекс, страна, область, район, город, улица, дом, квартира); оценки по экзаменам; проходной балл.* Вывести данные про абитуриентов, проходной балл которых равен больше 4 .
11 «Государство»	*название страны; столица; государственный язык; население; площадь территории; денежная единица; государственный строй; глава государства.* Вывести данные про государства, население которых больше 20 млн жителей.
12 «Автомобиль»	*марка; цвет; серийный номер; регистрационный номер; год выпуска; год техосмотра; цена.*

	Вывести данные про автомобили, которым больше 2 лет.