Collections Generics

Василий Орлов, учебный центр NetCracker при МФТИ

NetCracker®

План лекции

- Понятие коллекции
- Различные типы коллекций, их сходства и различия
- Интерфейсы коллекций в Java
- Реализации интерфейсов коллекций в Java
- Специальные утилитные классы для работы с коллекциями в Java

- Понятие настраиваемого типа(generic)
- Различные примеры кода с generics
- Generics с ограничениями
- Маски

Коллекции

- **Коллекция(Collection)** – хранилище, поддерживающие разнообразные способы накопления и упорядочивания объектов с целью обеспечения возможностей эффективного доступа к ним.

- **Массив** — набор однотипных элементов, расположенных в памяти непосредственно друг за другом, доступ к которым осуществляется по индексу.

Классификация коллекций

- По логике организации:
  - Вектор(Vector)
  - Ассоциативный массив(Map)
  - Множество(Set)
- По реализации:
  - Массив(Array)
  - Связный список
  - Хеш-таблица(Hash table)

Вектор(vector)

- Элементы упорядочены, каждый имеет собственный номер, называемый индексом, по которому к нему можно в любой момент обратиться
- Как правило, в качестве индексов выступают последовательные целые числа
- Для обращения к элементу используется имя вектора и значение индекса
- Удаление элемента из вектора приводит к образованию пустого элемента

Ассоциативный массив(map)

- Неупорядоченная коллекция, хранящая пары «ключ — значение»
- Доступ к элементам производится по ключу
- Тип ключа должен допускать сравнение на равенство
- Любая пара может быть в любой момент удалена

Множество(set)

- Неупорядоченная коллекция, хранящая набор уникальных значений и поддерживающая для них операции добавления, удаления и определения вхождения
- По сути является ассоциативным массивом(map), где роль ключа играет сам элемент

Массив(array)

- **Массив** — набор однотипных элементов, расположенных в памяти непосредственно друг за другом, доступ к которым осуществляется по индексу.

- Сложность:
  - Вернуть значение по индексу: O(1)
  - Поиск: O(n)
  - Вставка: O(n)
  - Удаление: O(n)

Связный список

- **Связный список —** структура данных, состоящая из узлов, каждый из которых содержит как собственно данные, так и одну или две ссылки на следующий и/или предыдущий узел списка
- Сложность:
  - Вернуть значение по индексу: O(n)
  - Поиск: O(n)
  - Вставка: O(1)
  - Удаление: O(1)

Хеш-таблица(hash table)

- **Хеш-таблица —** структура данных, позволяющая хранить пары (ключ, значение) и выполнять три операции: добавления новой пары, операцию поиска и операцию удаления пары по ключу.
- Содержит некоторый массив, элементы которого есть списки пар.
- Выполнение операции в хеш-таблице начинается с вычисления хеш-функции от ключа. Получающееся хеш-значение играет роль индекса в массиве . Затем выполняемая операция (добавление, удаление или поиск) перенаправляется объекту, который хранится в соответствующей ячейке массива.

Хеш-таблица(hash table)

- Сложность:
  - Поиск: O(1)
  - Вставка: O(1)
  - Удаление: O(1)

Коллекции

- В Java коллекции разделены на интерфейсы, абстрагирующие общие принципы работы с коллекциями, и классы, реализующие конкретную функциональность
- Не все методы, заявленные в интерфейсах, должны в действительности реализовываться классами. Часть методов может просто выбрасывать исключение **`UnsupportedOperationException`**

Интерфейс Collection

- Является образующим для интерфейсов коллекций
- Определяет базовую функциональность любой коллекции
- Подразумевает добавление, удаление, выбор элементов в коллекции
- Допускает дубликаты и пустые элементы

Методы интерфейса Collection

- Добавление элементов
  `boolean add(Object o),`
  `boolean addAll(Collection c)`
- Исключение элементов
  `boolean remove(Object o),`
  `boolean removeAll(Collection c),`
  `boolean retainAll(Collection c),`
- Состояние коллекции
  `boolean contains(Object o),`
  `boolean containsAll(Collection c),`
  `boolean isEmpty(),`
  `int size()`
- Вспомогательные методы
  `Object[] toArray(),`
  `Iterator iterator()`

Интерфейс Set

- Расширяет интерфейс **`Collection`**
- Не разрешает наличие дубликатов
- Допускается наличие только одной ссылки **`null`**
- Объекты коллекции должны корректно реализовывать метод **`equals()`**

Интерфейс List

- Расширяет интерфейс `Collection`
- Подразумевает хранение упорядоченной последовательности объектов
- Порядок хранения определяется порядком добавления элементов
- Позволяет обращаться к элементам по их номеру

Специальные методы интерфейса List

- Адресное добавление
  `void add(int index, Object o)`,
  `boolean addAll(int index, Collection c)`
- Адресные операции с элементами
  `Object get(int index)`,
  `Object set(int index, Object o)`,
  `Object remove(int index)`
- Операции поиска
  `int indexOf(Object o)`,
  `int lastIndexOf(Object o)`
- Специальные операции
  `List subList(int from, int to)`

Интерфейс Iterator

Позволяет работать с коллекцией как с набором (серией) элементов:

- Получать следующий объект `Object next()`
- Проверять наличие следующего объекта `boolean hasNext()`

Интерфейс Map

- Не расширяет интерфейс **`Collection`**
- Подразумевает хранение набора объектов парами ключ/значение
- Ключи должны быть уникальными
- Порядок следования пар ключ/значение не определен
- Имеет расширение **`SortedMap`**, требующее упорядоченности по значениям ключей

Методы интерфейса Map

- Добавление объектов
  `Object put(Object key, Object value)`,
  `void putAll(Map t)`
- Исключение объектов
  `Object remove(Object key)`,
  `void clear()`
- Доступ к объекту по ключу
  `Object get(Object key)`
- Состояние
  `boolean containsValue(Object value)`,
  `boolean containsKey(Object key)`,
  `int size()`,
  `boolean isEmpty()`

Интерфейсы SortedMap и SortedSet

- SortedSet расширяет Set храня объекты в отсортированном порядке, требует чтобы объекты, которые содержит коллекция реализовывали интерфейс Comaprable либо требует задать специальный Comparator, который умел бы сравнивать объекты из коллекции.

- SortedMap расширяет Map храня значения в отсортированном по ключам порядке, требует чтобы ключи реализовывали интерфейс Comaprable либо требует задать специальный Comparator, который умел бы сравнивать ключи.

Классы коллекций

- Динамический массив:
  ArrayList (**List**)
- Двухсвязный список:
  LinkedList (**List**)
- B-деревья:
  TreeSet(**SortedSet**), **TreeMap** (**SortedMap**)
- Хеш-таблица:
  HashMap (**Map**), **HashSet** (**Set**)

java.util.Arrays

Содержит статические методы для работы с массивами

- Представление массива списком
  `List asList(Object[] a)`
- Поиск элемента в массиве
  `int binarySearch(…[] a, … key)`
- Сравнение массивов по элементам
  `boolean equals(…[] a1, …[] a2)`
- Заполнение массива элементами
  `fill(…[] a, int from, int to, … val)`
- Сортировка массива
  `sort(…[] a, int from, int to)`

Настраиваемые типы(generic)

- Позволяют создавать классы в которых типы полей, типы аргументов методов и типы возвращаемых методами значений могут меняться

```
class Gen<T> {
    private T value;
    public Gen(T v) { value = v;}
    public T getValue() { return value;}
    public void setValue (T v) { value = v;}
}

public class GenTest
{
    public static void main(String[] args)
        {
            Gen<Integer> intObj = new Gen<Integer>(777);
            Gen<String> strObj = new Gen<String>("Some text");
          Gen<String> strObj1 = new Gen<String>(555); // ошибка
компилятора
            int i = intObj.getValue() + 2;
            String s = strObj.getValue ().substring(2);
            intObj = strObj; // ошибка компилятора
        }
}
```

Настраиваемые типы(generic)

```
public class GenTest
{
    public static void main(String[] args)
        {
            Gen<Integer> intObj = new Gen<Integer>(777);
            Gen<String> strObj = new Gen<String>("Some text");
          Gen<String> strObj1 = new Gen<String>(555); // ошибка
компилятора
            int i = intObj.getValue () + 2;
            String s = strObj.getValue ().substring(2);
            intObj = strObj; // ошибка компилятора
        }
}

public class GenTest
{
    public static void main(String[] args)
        {
            Gen strObj = new Gen<String>("Some text");
            strObj.setValue(555); // сообщения об ошибке нет
            strObj.setValue("Some text");
            String s = (String)strObj.getValue();
            Integer i = (Integer)intObj.getValue(); // ошибка runtime
```

Несколько generic типов в одном классе

```
class Gen<T, V>
{
     public T ob1;
     public V ob2;
     public Gen(T ob1, V ob2)
     {
          this.ob1= ob1;
          this.ob2= ob2;
     }
}

public class GenTest
{
     public static void main(String[] args)
     {
          Gen<Integer, String> data =  new Gen<Integer, String>(10, "Test");
          int i = data.ob1 + 2;
          String s = data.ob2.substring(1);
          Gen<Integer, Object> object = new Gen<Integer, Object>(10, "Test");
          int k = object.ob1 + 2;
          String s = object.ob2.substring(1); // Ошибка.
          s = ((String)object.ob2).substring(1);
     }
}
```

Generic с ограничениями

```
class GenTest<T extends Animal>
{
    private T t;
    public GenTest(T t) {
            this.t=t;
    }
}

public static void main(String[] args)
{
        GenTest<Dog> genTest1 = new GenTest<Dog>(new Dog());

        /* ошибка */
        GenTest<Dog> genTest2 = new GenTest<Cat>(new Cat());

        /* ошибка */
        GenTest<Animal> genTest3 = new GenTest<Dog>(new Dog());
    }
}
```


Generic методы

- public static <T> T getFirst(Collection<T> col) {...}

- <Integer>swap(ints, 1, 3);
- strings.<Integer>zip(ints);

Маски

- void drawAll(Collection<? extends Glyph> glyphs) {…}
- <T extends Glyph> void drawAll(Collection<T> glyphs) {…}

- static void doSomeWork(Map<?, ? extends Glyph> map) {…}

Спасибо за внимание!

NetCracker®