Java Professional
module #2

lecture #3. Iterators and Iterable. Foreach syntax.

Mentor: Iurii Avramenko

lecture #3. Iterators and Iterable. Foreach syntax.

- Iterator
  - use Iterator in Java
- Enumeration
- ListIterator
- Iterator and Spliterator
- Iterator and Iterable Interface
- For-each loop in Java
- Retrieving Elements from Collection in Java (additional)

Iterator

- Итератор используется в структуре Collection для извлечения элементов один за другим.
- Итератор применим к любому объекту Collection.
- Используя Iterator, мы можем выполнять как операции чтения, так и операции удаления.
- Итератор необходимо использовать всякий раз, когда мы хотим перечислить элементы во всех реализованных интерфейсах фреймворка Collection, таких как Set, List, Queue, Deque и во всех реализованных классах интерфейса Map.
- Итератор — единственный «переборщик», доступный для всей структуры Collection.
- Итератор – перебирает элементы в одном направлении

Iterator

Синтаксис:

```
Iterator itr = <collection>.iterator();
```

Методы интерфейса:

1. hasNext(): возвращает true, если итерация содержит еще элемент.
2. next(): возвращает следующий элемент в итерации.
   Исключение NoSuchElementException , если больше нет элементов.
3. remove(): удаляет следующий элемент в итерации.
   Исключение UnsupportedOperationException : если операция удаления не поддерживается этим итератором
   Исключение IllegalStateException : если next() метод еще не был вызван или метод remove() уже был вызван после последнего вызова next() метода.

Iterator

Синтаксис:

```
Iterator itr = <collection>.iterator();
```

Методы интерфейса:

1. hasNext(): возвращает true, если итерация содержит еще элемент.
2. next(): возвращает следующий элемент в итерации.
   Исключение NoSuchElementException , если больше нет элементов.
3. remove(): удаляет следующий элемент в итерации.
   Исключение UnsupportedOperationException: если операция удаления не поддерживается этим итератором
   Исключение IllegalStateException: если next() метод еще не был вызван или метод remove() уже был вызван после последнего вызова next() метода.

Enumeration

- Enumeration - интерфейс, используемый для получения элементов устаревших коллекций (Vector, Hashtable).

- Enumeration — это первый итератор, представленный в JDK 1.0, остальные включены в JDK 1.2 с большей функциональностью.

- Как создать объект Enumeration - вызвать метод elements() векторного класса для любого векторного объекта

Синтаксис:

```
Enumeration e = <vector>.elements();
```

Методы:

1. hasMoreElements(): этот метод проверяет, содержит ли это перечисление больше элементов или нет.
2. nextElement(): этот метод возвращает следующий элемент этого перечисления. Исключение NoSuchElementException, если больше нет элементов

Enumeration VS Iterator

Enumeration	Iterator
Java 1.0	Java 1.2
Устаревший	Современный
используется для итерации только классов Legacy Collection.	Используется для любого класса Collection.
поддерживает только операцию READ	поддерживает операции READ и DELETE
не универсальный	универсальный курсор

ListIterator

- ListIterator - применим только для реализованных классов коллекции List
- ListIterator - обеспечивает двунаправленную итерацию.
- ListIterator - необходимо использовать, когда мы хотим перечислить элементы списка.
- ListIterator - имеет больше методов, чем итератор.
- ListIterator - можно создать, вызвав метод listIterator() , присутствующий в интерфейсе List.

ListIterator

Синтаксис:

ListIterator iterator = <list>.listIterator();

Методы:

1. void add(Object object) : объект вставляется непосредственно перед элементом, возвращаемым функцией next().
2. boolean hasNext() : возвращает true, если в списке есть следующий элемент.
3. boolean hasPrevious() : возвращает true, если в списке есть предыдущий элемент.
4. Object next() : возвращает следующий элемент списка.
   Исключение NoSuchElementException, если в списке нет следующего элемента.
5. Object previous() : возвращает предыдущий элемент списка.
   Исключение NoSuchElementException, если предыдущего элемента нет.
6. void remove() : удаляет текущий элемент из списка.
   Исключение 'IllegalStateException', если эта функция вызывается до вызова next() или previous().

Iterator and Spliterator

- Spliterator, как и другие итераторы, предназначен для обхода элементов Collection
- Spliterator включен в JDK 8 для поддержки эффективного параллельного обхода
- Spliterator можно использовать, даже если нет параллельнщго выполнение. Одна из причин, по которой может понадобиться это сделать, заключается в том, что Spliterator объединяет операции hasNext и next в один метод

Spliterator

Синтаксис:
Spliterator spr = <collection>.spliterator();

Методы:

1. int characteristics() - Возвращает набор характеристик этого Spliterator и его элементов
2. long estimateSize( ) - возвращает оценку количества элементов, оставшихся для итерации, или возвращает Long.MAX_VALUE, если число бесконечно, неизвестно или слишком дорого для вычисления
3. default long getExactSizeIfKnown( ) - метод, который возвращает размер оценки, если этот Spliterator имеет SIZED, иначе -1
4. default Comparator<? super T> getComparator( ) - Если коллекция этого Spliterator SORTED компаратором, возвращает этот компаратор. Если коллекция SORTED в естественном порядке, возвращает null. Если источник не SORTED, генерируется исключение IllegalStateException.
5. default boolean hasCharacteristics(int val) - возвращает true, если характеристики этого Spliterator() содержат все заданные характеристики
6. *boolean tryAdvance(Consumer<? super T> action)*
7. *default void forEachRemaining(Consumer<? super T>action)*
8. *Spliterator<T> trySplit( )*

Iterator and Iterable Interface

Итераторы используются в структуре коллекции в Java для извлечения элементов один за другим

- Интерфейс Iterable был представлен в JDK 1.5

- объект, реализующий Iterable, позволяет выполнять итерацию

- Существует три способа итерирования объектов Iterable
  - Использование for-each loop
  - Использование Iterable forEach loop
  - Использование Iterator<T> interface

Каждый класс, который соответствующим образом реализует интерфейс Iterable, может использоваться в цикле for-each.
Необходимость реализации интерфейса Iterator возникает при разработке пользовательских структур данных.

Iterator and Iterable Interface

Итераторы используются в структуре коллекции в Java для извлечения элементов один за другим

- Интерфейс Iterable был представлен в JDK 1.5

- объект, реализующий Iterable, позволяет выполнять итерацию

- Существует три способа итерирования объектов Iterable
  - Использование for-each loop
  - Использование Iterable forEach loop
  - Использование Iterator<T> interface

Каждый класс, который соответствующим образом реализует интерфейс Iterable, может использоваться в цикле for-each.
Необходимость реализации интерфейса Iterator возникает при разработке пользовательских структур данных.

For-each loop in Java

For-each — это еще один метод обхода коллекции, как цикл for, цикл while, цикл do-while, представленный в Java5

- начинается с ключевого слова for , как обычный цикл for.
- Вместо объявления и инициализации переменной счетчика цикла вы объявляете переменную того же типа, что и базовый тип массива, за которым следует двоеточие, за которым следует имя коллекции.
- В теле цикла вы можете использовать созданную вами переменную цикла, а не индексированный элемент массива.

- обычно используется для перебора массива или класса Collections (например, ArrayList).

Синтаксис:

```
for (type var : array) {
    // statements;
}
```

For-each ограничения

1. for-each не подходит, если вы хотите изменить массив
2. for-each не отслеживают индекс
3. for-each выполняет итерацию только вперед по массиву за один шаг
4. for-each также снижает производительность по сравнению с простой итерацией

Retrieving Elements from Collection in Java (additional)

Четыре способа извлечения любых элементов из объекта коллекции

1. Classic for
2. For-each loop
3. С Java 8 используя lambda expressions
4. Iterator Interface