Java Professional module #2

Iterators and Iterable. Foreach syntax.

Mentor: Rustam Khakov

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)

Enumeration

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

- Enumeration — это первый итератор

- Создание - вызвать метод elements()

Синтаксис:

Enumeration e = <vector>.elements();

Методы:

- hasMoreElements():
- nextElement(): этот метод возвращает следующий элемент этого перечисления.
    - Исключение NoSuchElementException, если больше нет элементов

Iterator

- В структуре Collection для перечисления (любая реализация).
- Применим к любому объекту Collection.
- Операции
  - чтения
  - удаления.
- Единственный «переборщик», доступный для всей структуры Collection.
- Прямой проход (от начала в конец)

Iterator

Iterator itr = <collection>.iterator();

Методы:

- hasNext() - проверяет если еще элемент.
- next(): возвращает следующий элемент в итерации.
  - Исключение NoSuchElementException , если больше нет элементов.
- remove(): удаляет следующий элемент в итерации.
  - Исключение UnsupportedOperationException : если операция не поддерживается
  - Исключение IllegalStateException : если next() метод еще не был вызван или метод remove() уже был вызван после последнего вызова next() метода.

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 extends Iterator

- Применим только для реализованных классов коллекции List
- Обеспечивает двунаправленную итерацию.
- Необходимо использовать, когда мы хотим перечислить элементы списка.
- Имеет больше методов, чем итератор.
- Можно создать, вызвав метод listIterator() , присутствующий в интерфейсе List.

ListIterator

ListIterator iterator = <list>.listIterator();

Методы:

- boolean hasNext()
- Object next() :
- void remove()

- void add(Object object) : объект вставляется непосредственно перед элементом, возвращаемым функцией next().
  - Object previous() : возвращает предыдущий элемент списка.
    - Исключение NoSuchElementException, если предыдущего элемента нет.
  - boolean hasPrevious() : возвращает true, если в списке есть предыдущий элемент.

Iterator and Spliterator

- Как и другие итераторы, предназначен для обхода элементов Collection
- Включен в JDK 8 для поддержки эффективного параллельного обхода
- Можно использовать, даже если нет параллельнщго выполнение.

Spliterator

Spliterator spr = <collection>.spliterator();

Методы:

- int characteristics() - Возвращает набор характеристик этого Spliterator и его элементов

- long estimateSize( ) - возвращает оценку количества элементов, оставшихся для итерации, или возвращает Long.MAX_VALUE, если число бесконечно, неизвестно или слишком дорого для вычисления

- default long getExactSizeIfKnown( ) - метод, который возвращает размер оценки, если этот Spliterator имеет SIZED, иначе -1

- 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.

- default boolean hasCharacteristics(int val) - возвращает true, если характеристики этого Spliterator() содержат все заданные характеристики

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 значений

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

for-each также снижает производительность по сравнению с простой итерацией?
- и да и нет

Retrieving Elements from Collection in Java

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