Итераторы

Java Collections
7 уровень, 3 лекция

ОТКРЫТА

— Привет, Амиго!

— Привет, Элли!

— Сегодня я хочу рассказать тебе про итераторы.

Итераторы придумали практически тогда, когда и коллекции. Основная задача коллекций была – хранить элементы, а основная задача итератора – выдавать эти элементы по одному.

— А что сложного в том, чтобы выдать набор элементов?

— Во-первых, некоторые коллекции, как например Set не имеют установленного порядка элементов и/или он постоянно меняется.

Во-вторых, некоторые структуры данных могут хранить объекты очень сложно: различными группами, списками и т.д. Т.е. задача отдать последовательно все элементыбудет сложной и нетривиальной.

В третьих – коллекции имеют свойство меняться. Решил ты вывести на экран все содержимое коллекции, а прямо в середине вывода JVM переключилась на другую нить, которая половину элементов из этой коллекции заменила на другую. Вот и получишь ты вместо вывода не пойми что.

— М-да.

— Вот! Именно такие проблемы должен был решить итератор. Итератор – это специальный внутренний объект в коллекции, который с одной стороны имеет доступ ко всем ее private данным и знает ее внутреннюю структуру, с другой – реализует общедоступный интерфейс Iterator, благодаря чему все знают, как с ним работать.

Некоторые итераторы имеют внутри себя массив, куда копируются все элементы коллекции во время создания итератора. Это гарантирует, что последующее изменение коллекции не повлияет на порядок и количество элементов.

Думаю, ты уже сталкивался с тем, что при работе с **for each** нельзя одновременно «идти по коллекции циклом» и удалять из нее элементы. Это все именно из-за устройства итератора.

В новых коллекциях, добавленных в библиотеке concurrency, устройство итератора переработано, поэтому там такой проблемы нет.

Давай я тебе напомню, как устроен итератор.

В Java есть специальный интерфейс Iterator, вот какие у него методы:

Методы интерфейса Iterator<E>	**Описание**
`boolean hasNext()`	Проверяет, есть ли еще элементы
`E next()`	Возвращает текущий элемент и переключается на следующий.
`void remove()`	Удаляет текущий элемент

Итератор позволяет поочередно получить все элементы коллекции. Логичнее представить итератор чем-то вроде InputStream

Метод **next**() возвращает следующий (очередной) элемент коллекции.

Метод **hasNext**() используется, чтобы проверять, есть ли еще элементы.

Ну, а **remove**() – удаляет текущий элемент.

Вопросы есть?

— А почему методы называются так странно? Почему не isEmpty() или getNextElement()?

Разве так не логичнее?

— Логичнее, но такие названия пришли из языка C++, где итераторы появились раньше.

— Ясно. Продолжим.

Кроме итератора есть еще интерфейс Iterable – его должны реализовывать все коллекции, которые поддерживают итератор. У него есть единственный метод:

Методы interface Iterable<T>	**Описание**
`Iterator<T>iterator()`	Возвращает объект-итератор

С помощью этого метода у любой коллекции можно получить объект итератор для обхода ее элементов. Давай обойдем все элементы дерева в коллекции **TreeSet**:

Пример

```
1  TreeSet<String> set = new TreeSet<String>();
2  Iterator<String> iterator = set.iterator();
3
4  while (iterator.hasNext())
5  {
6   String item = iterator.next();
7   System.out.println(item);
8  }
```

Такое использование итератора не очень удобно – слишком много лишнего и очевидного кода. Ситуация упростилась, когда в Java появился цикл по итератору – **for-each.**

Теперь такой код гораздо компактнее и читабельнее:

Было

```
1  TreeSet<String> set = new TreeSet<String>();
2  Iterator<String> iterator = set.iterator();
3
4  while (iterator.hasNext())
5  {
6   String item = iterator.next();
7   System.out.println(item);
8  }
```

Стало

```
1  TreeSet<String> set = new TreeSet<String>(
2
3  for(String item : set)
4  {
5   System.out.println(item);
6  }
```

Это один и тот же код! Итератор используется и там, и там.

НАЧАТЬ ОБУЧЕНИЕ

Цикл **for-each** можно использовать для любых объектов, которые поддерживают итератор. Т.е. ты можешь написать свой класс, добавить ему метод **iterator**() и сможешь использовать его объекты в правой части конструкции **for-each.**

— Ого! Я, конечно, не рвусь писать собственные коллекции и итераторы, но предложение все равно заманчивое. Возьму на карандаш.

— Кроме того, есть еще одна популярная разновидность итераторов, для которой даже придумали свой интерфейс. Речь идет об итераторе для списков – **ListIterator**.

Списки, независимо от реализации, обладают порядком элементов, что в свою очередь позволяет работать с ними через итератор чуть более удобно.

Вот какие методы есть у интерфейса **ListIterator**<E>:

Метод	**Описание**
`boolean hasNext()`	Проверяет, есть ли еще элементы впереди.
`E next()`	Возвращает следующий элемент.
`int nextIndex()`	Возвращает индекс следующего элемента
`void set(E e)`	Меняет значение текущего элемента
`boolean hasPrevious()`	Проверяет, есть ли элементы позади.
`E previous()`	Возвращает предыдущий элемент
`int previousIndex()`	Возвращает индекс предыдущего элемента
`void remove()`	Удаляет текущий элемент
`void add(E e)`	Добавляет элемент в список.

Т.е. тут мы можем ходить не только вперед, но и назад. И еще пара фич по мелочи.

— Что ж, интересная штука. А где его используют?

— Например, ты хочешь двигаться туда-обратно по связному списку. При этом операция get будет довольно медленной, а операция next() очень быстрой.

— Хм. Убедила. Буду иметь ввиду.

Спасибо, Элли!

Предыдущая лекция

Виталий Уровень 41, Минск, Беларусь — 28 февраля, 14:19

Iterator - объект, с помощью которого происходит обход коллекции
Iterable - интерфейс, который есть у Iterator и с помощью которого Iterator знает как обойти данную коллекцию
iterator() - метод интерфейса Iterable, который возвращает Iterator
hasNext(), next(), remove() - методы класса Iterator

Ответить +3

Ars Уровень 41 — 27 ноября 2021, 10:06

Взять на карандáш - сделать мысленную или письменную заметку о ком-либо с целью в дальнейшем следить за ним, уделять ему пристальное внимание.

Этот термин появился в начале XIX века, сначала в жандармерии, а потом и в других государственных ведомствах Российской империи. Дело в том, что в списках "неблагонадёжных" в Отдельном Жандармском Корпусе (Губернские Жандармские Управления и Отделения), писавшихся чернилами, заметки ставились графитовым карандашом, чтобы, при необходимости их можно было бы просто стереть. Отсюда и пошло выражение "взять на карандаш". То есть в человеке или каком-то действии есть сомнение. Подтвердится - тогда в список или доклад чернилами, а нет, тогда стереть и никто не узнает. **То есть синоним - подозрение, но не имеющее доказательств.**

Взятие на карандаш возможности сделать свой итератор выглядит крайне не к месту...

Ответить +4

LuneFox инженер по сопровождению в **BIFIT** EXPERT — 24 февраля, 17:58

Необходимость создания своего итератора выглядит довольно подозрительно - не уверен, что это стоит записывать в список своих будущих дел. И нет никаких доказательств того, что мне это точно пригодится. Возьму на карандаш…

Ответить +4

Bagbich Уровень 51, Новосибирск, Russian Federation — 21 июня, 14:45

душно

Ответить +2

Андрей Овчаренко Уровень 41, Москва — 15 июля 2021, 22:09

```
1  При этом операция get будет довольно медленной, а операция next() очень быстрой.
```

вот только операция get в этой лекции не упоминалась. Что имел ввиду автор?

Ответить 0

Maks Panteleev Java Developer в **Bell Integrator** — 25 июля 2021, 16:13

get это стандартный метод получения элемента списка по индексу или по имени, речь о том, что если ты получил условный 101 элемент коллекции, а тебе нужен 102 - ты можешь вызвать метод get(102) а можешь вызывать у итератора next и скорость последнего решения будет гораздо быстрее. Если элементов миллион, то даже 100 раз подряд вызвать некст будет быстрее чем один раз гет)

Ответить +3

LuneFox инженер по сопровождению в **BIFIT** EXPERT — 24 февраля, 18:03

Лучше один раз y-get-ить, чем 100 раз y-next-ить :D

Ответить 0

alex_us Уровень 41, Симферополь — 26 января 2021, 22:03

Странный вопрос задам.. но все же... Почему бы им не перестроить теорию?)) зачем нужна эта статья на данном уровне?

Ответить 0

Андрей Лихтарович Уровень 40, Минск, Беларусь — 13 февраля 2021, 22:27

1. Я например неплохо освежаю знания на таких лекциях.
2. Такие лекции закрывают дыры в обучении (когда раньше это было непонятно, а после прочтения вновь некоторые моменты становятся очевидными) , таким образом выстраивается хорошая структурированная база знаний в голове а не просто (это-знаю ___ это не знаю)

Ответить +14

alex_us Уровень 41, Симферополь — 13 февраля 2021, 22:49

согласен. Жаль только не сделали на 41 уровне лекцию повтора как создавать обьект и работать с примитивами. Метод принтлн там . И конечно нужно было повторить создание хелло ворд

Ответить +1

Андрей Лихтарович Уровень 40, Минск, Беларусь — 18 февраля 2021, 23:58

это было-бы лучше чем просто картинка))))

НАЧАТЬ ОБУЧЕНИЕ

Ну про своевременность подачи материала тут уже писали... Очередное док-во того, что javarush это всё-таки не про теорию, а про практику.

Ответить +4

Виталий Java Developer 18 апреля 2020, 19:01

Цикл for-each - это синтаксический сахар, который под капотом использует итератор, но при этом лишает нас доступа к методу remove() итератора.

Ответить +15

Саш Уровень 14, Москва, Россия 20 ноября 2019, 00:13

Почему 27 т.м? Лекция 7 лвл

Ответить 0

MartyMcAir Уровень 41, Россия 16 марта 2020, 18:09

7й lvl для квеста Collections
(обычно тут пиплы с уровнями >30, т.к. Это предпоследний квест, потом Games Quest..
У каждого квеста уровни от 0 до 10.)

Ответить 0

Ilya Sakharov Уровень 41, Москва 23 октября 2018, 19:18

А массивы?
Там тоже работает for-each, а итератора нет :)
Или есть?

Ответить +3

Justinian Judge в **Mega City One** MASTER 6 сентября 2019, 14:45

Судя по документации, которая различает for each для iterable объектов и отдельно массивов, получается, что итератора в массивах нет.
https://stackoverflow.com/questions/35518471/in-java-8-why-were-arrays-not-given-the-foreach-method-...

Ответить +5

Ярослав QA Automation Test Engine в **Ukraine** EXPERT 7 августа 2018, 18:13

WTF ?

E next() Возвращает **текущий** элемент и переключается на следующий.

Метод next() возвращает **следующий** (очередной) элемент коллекции.
P.S. Каким-же космосом мне казался итератор на 8 левеле ...

Ответить +15

Aleksandr Уровень 23, Санкт-Петербург 30 марта 2020, 10:23

По твоей логике, находясь на последнем элементе (пускай это будет ArrayList), вызывая метод next(), он должен вернуть что?! Следующий элемент? Т.е. null?

Ответить +2

Роман Тарнакин Уровень 35, Москва, Россия 2 июля 2020, 14:16

Если вы откроете оф.доки по интерфейсу Iterator<E>, то сможете прочесть следующее:

```
E next()
Returns the next element in the iteration.
Returns:
the next element in the iteration
Throws:
NoSuchElementException - if the iteration has no more elements
```

Ответить +3

Kes Чайник в **Банк** 26 января, 08:26

Написал Регине. Может подправят.
Действительно просто из-за игры слов становится не ясно. Хотя по сути.

```
list.add(1);
list.add(2);
list.add(3);
Iterator<Integer> it = list.iterator();
System.out.println("Первый i.next() возвращает: " + it.next());
System.out.println("Второй i.next() возвращает: " + it.next());
System.out.println("Второй i.next() возвращает: " + it.next());
System.out.println("Второй i.next() возвращает: " + it.next());
```

```
Первый i.next() возвращает: 1
Второй i.next() возвращает: 2
Второй i.next() возвращает: 3
Exception in thread "main" java.util.NoSuchElementException
```

next
1. Возвращает следующий элемент, если мы уже стоим на каком-то
2. Возвращает "первый" элемент, если мы еще ни разу не вызывали next
3. Генерирует исключение, если следующего элемента нет.

Таким образом не в первом ни во втором случае в лекции эта тонкость в явном виде не указана. Ее сложно тремя словами описать.

Ответить +1

Наиль Калимуллин Уровень 8, Уфа, Россия 4 июля 2018, 15:51

читать следующую лекцию за 27 кц .... откуда такие ценники? цена на луц выросла?

Ответить +12

Артем Прудников Уровень 35, Москва, Россия 20 января 2018, 15:33

То чувство когда ты все это изучал самостоятельно 20 уровней назад. Мммммм. Но в любом случае нужно понимать, что если эту информацию засунуть на 10 уровень, то тогда сместятся другие уровни.

Ответить +2

Андрей Петухов Уровень 40, Москва, Россия 29 августа 2017, 16:04

Материал дается так своевременно, что аж писать кипятком захотелось.

Ответить +48

MaxLich Уровень 40, Санкт-Петербург, Россия 19 июля 2017, 15:09

Тут ошибка: "void add(E e) Добавляет элемент в конец списка." Этот метода добавляет элемент после текущего элемента, не в конец списка.

Ответить +3

СКАЧИВАЙТЕ НАШИ ПРИЛОЖЕНИЯ