---

пример использования

// Example1.java (method iteratorGeneralExample())

public class IteratorPractice {

public static void main(String[] args) {

List<String> list = new ArrayList<>();

list.add("A");

list.add("B");

list.add("C");

list.add("D");

list.add("E");

Iterator<String> iterator = list.iterator();

System.out.println("List elements : ");

while (iterator.hasNext()) {

System.out.print(iterator.next() + " ");

}

System.out.println();

}

}

---

пример remove()

// Example1.java (method iteratorRemove())

public static void main(String[] args) {

ArrayList<Integer> list = new ArrayList<>();

for (int i = 0; i < 10; i++) {

list.add(i);

}

System.out.println(list);

//

// for (Integer e : list) {

// if (e % 2 != 0) {

// list.remove(e);

// }

// }

// for (int i = 0; i < 10; i++) {

// if (list.get(i) % 2 != 0) {

// list.remove(i);

// }

// }

for (int i = 0; i < list.size(); i++) {

if (list.get(i) % 2 != 0) {

list.remove(i);

}

}

System.out.println(list);

// Iterator<Integer> itr = list.iterator();

//

// while (itr.hasNext()) {

// int element = itr.next();

// System.out.print(element + " ");

//

// if (element % 2 != 0) {

// itr.remove();

// }

// }

//

// System.out.println();

// System.out.println(list);

}

Преимущества итератора

- можем использовать его для любого класса Collection.

- поддерживает операции READ и REMOVE.

- универсальный API для Collection.

- Названия методов просты и их легко использовать.

---

Enumeration

пример

// Example2.java

public static void main(String[] args) {

Vector<Integer> v = new Vector();

for (int i = 0; i < 10; i++) {

v.addElement(i);

}

System.out.println(v);

Enumeration<Integer> e = v.elements();

while (e.hasMoreElements()) {

int i = e.nextElement();

System.out.print(i + " ");

}

}

---

NOte -> проговорить основные различия

обьяснить что такое legacy collection

Унаследованные классы (Legacy Classes) Collection Framework – это классы языка Java 1.0/1.1.

List - Vector, Stack

Map - Dictionary, Hashtable, Properties

---

ListIterator

// Example3.java

public static void main(String[] args) {

ArrayList<String> list = new ArrayList<>();

list.add("A");

list.add("B");

list.add("C");

list.add("D");

list.add("E");

ListIterator<String> iterator = list.listIterator();

System.out.println("Displaying list elements in forward direction : ");

while (iterator.hasNext()) {

System.out.print(iterator.next() + " ");

}

System.out.println();

System.out.println("Displaying list elements in backward direction : ");

while (iterator.hasPrevious()) {

System.out.print(iterator.previous() + " ");

}

System.out.println();

}

---

Spliterator

Note -> 6,7,8 пункт не вдаемся в подробности, там пока много не известного для студентов

public static void main(String[] args)

{

ArrayList<Integer> list = new ArrayList<>();

list.add(1);

list.add(2);

list.add(-3);

list.add(-4);

list.add(5);

Spliterator<Integer> spr = list.spliterator();

// estimateSize method

System.out.println("estimate size : " + spr.estimateSize());

// getExactSizeIfKnown method

System.out.println("exact size : " + spr.getExactSizeIfKnown());

// hasCharacteristics and characteristics method

System.out.println(spr.hasCharacteristics(spr.characteristics()));

// getComparator method

System.out.println(spr.getComparator());

}

---

интерфейс Iterable

Note -> тут мог бы быть пример с реализацией кастомного итератора, на примере MyLinkedList, но это следующая тема. Там можно будет реализовать и вспомнить

---

// Example5.java

For-each

public static void main(String[] arg)

{

{

int[] arr = { 1, 13, 9, 11, 10 };

int max = getMax(arr);

System.out.println("The max is " + max);

}

}

public static int getMax(int[] arr)

{

int max = arr[0];

// for each loop

for (int num : arr){

if (num > max){

max = num;

}

}

return max;

}

---

for-each

Причина почему медленнее, но это не всегда истина!!!

- обычные массивы с циклом for-each, на этапе компиляции будут преобразованы в обычный цикл for с целочисленной счетчиком-переменной.

- если вы используете for-each с коллекциями, то на этапе компиляции цикл будет преобразован в for с итератором.

Note -> смотрим на время, как много его осталось - можно продемонстрировать в коде, можно скомпилировать код, можно написать сравнение.

---

Retrieving Elements

// Example6.java (method one())

#1

for (int i = 0, listSize = list.size(); i < listSize; i++) {

Integer integer = list.get(i);

System.out.println(integer);

}

// Example6.java (method two())

#2 - читаем «:», используемый в цикле for-each, как «in». цикл читается как «для каждого элемента в элементах»

for (Integer integer : list) {

System.out.println(integer);

}

// Example6.java (method three())

#3 - В Java 8, используя лямбда-выражения, мы можем просто заменить цикл for-each на

list.forEach (e -> System.out.println(e) );

// Example6.java (method four())

#4 - Итератор — это интерфейс, предоставляемый структурой для обхода коллекции и для последовательного доступа к элементам

for(Iterator<Integer> i = list.iterator(); i.hasNext();) {

System.out.println(i.next());

}

---

Разомнемся и отвлечемся на последок

Простая задачка на 5 минут

Note -> можно как самостоятельную

В стране Лимпопо колличество всех зверей равно 10 миллионов. Рождаемость составляет 14 зверей на 1000, смертность - 8 зверей.

Рассчитайте, склдбко зверей будет через 10 лет, принимая во внимание, что показатели рождаемости и смертности постоянны.