Java Professional
module #2

lecture #4. List implementations. LinkedList.

Mentor:

lecture #4. Lists implementations. LinkedList.

- Что такое LinkedList
- Операции над LinkedList
  - Создание связанного списка
  - Добавление объекта в конец связанного списка
  - Добавление объекта в середину связанного списка
  - Удаление объекта из списка
  - Clear()
  - Contains()
- LinkedList VS ArrayList

LinkedList

LinkedList — класс, реализующий два интерфейса — ***List*** и ***Deque***.

- Это обеспечивает возможность создания двунаправленной очереди из любых (в том числе и *null*) элементов.

- Каждый объект, помещенный в связанный список, является узлом (нодом).

- Каждый узел содержит элемент, ссылку на предыдущий и следующий узел.

- Фактически связанный список состоит из последовательности узлов, каждый из которых предназначен для хранения объекта определенного типа.

Строение LinkedList

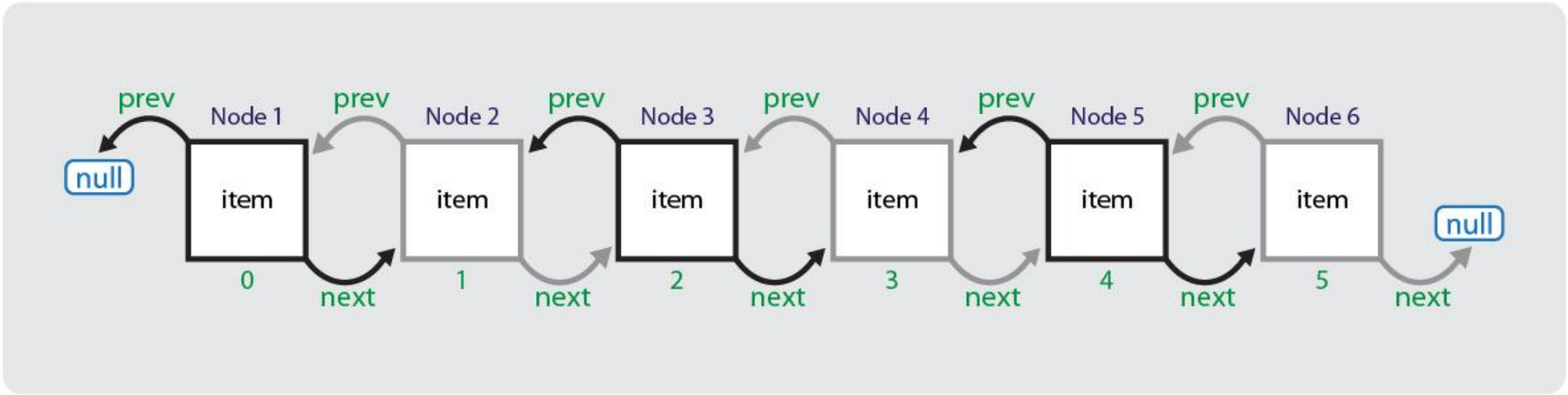


Рис. 1: Общий вид связанного списка

Создание связанного списка

size	0
first	null
last	null

Рис. 2: Состояние объекта сразу после создания

Добавление объекта в конец связанного списка

При каждом добавлении объекта в список создается один новый узел, а также изменяются значения полей связанного списка (*size*, *first*, *last*). В случае с добавлением первого элемента создается узел, у которого предыдущий и следующий элементы отсутствуют, т.е. являются *null*, размер коллекции увеличивается на 1, а созданный узел устанавливается как первый и последний элемент коллекции.

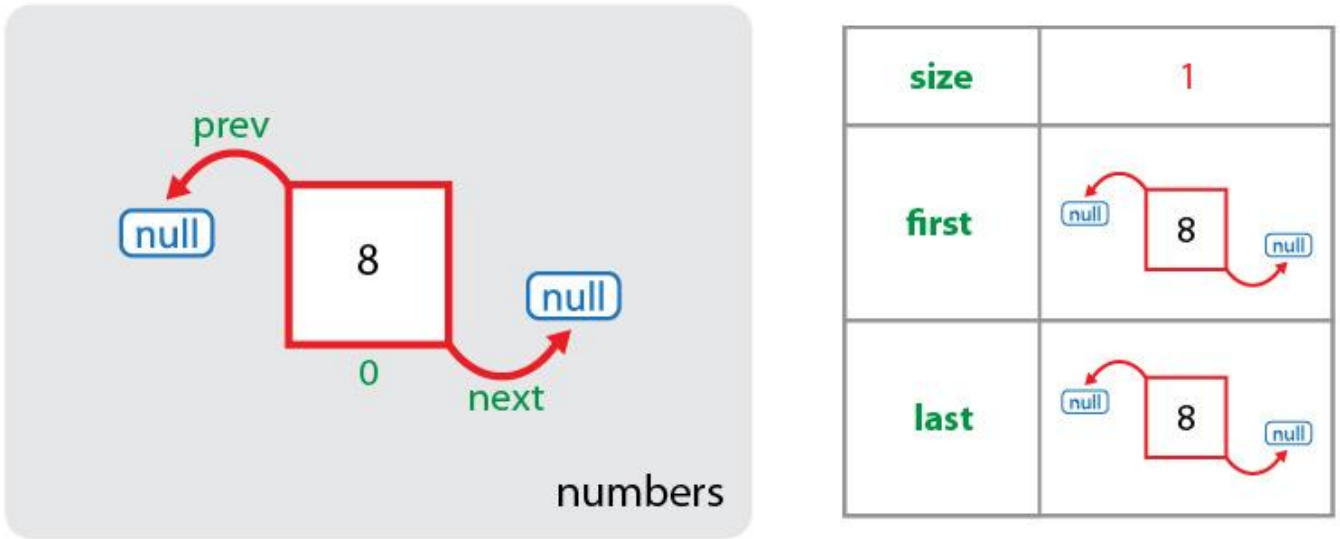
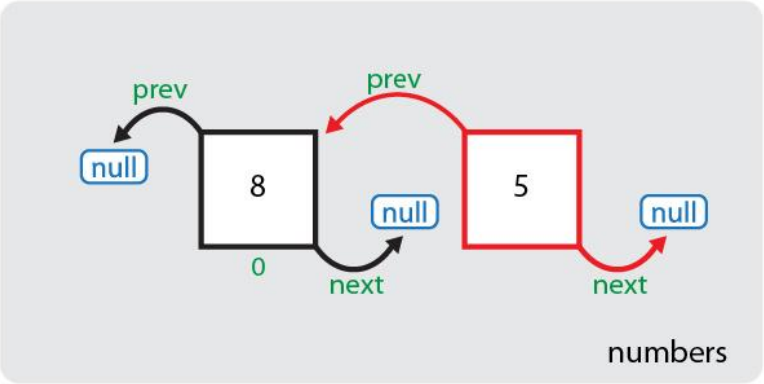
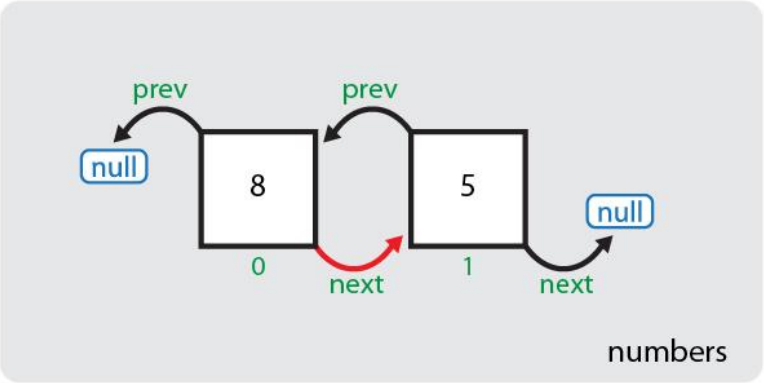


Рис. 3: Добавление первого объекта в связанный список



size	1
first	null 8 null
last	8 5 null

Рис. 4: Добавление второго объекта в связанный список (этап 1)



size	2
first	null 8 5
last	8 5 null

Рис. 5: Добавление второго объекта в связанный список (этап 2)

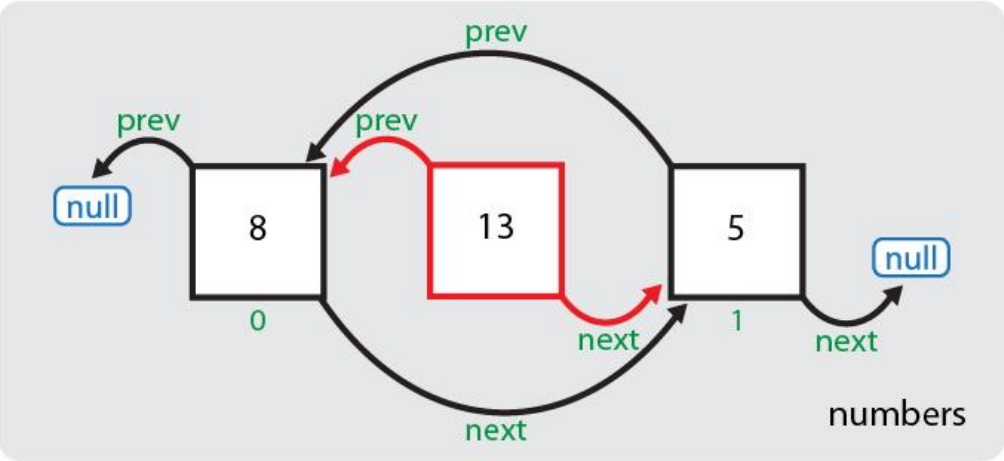
Добавим еще один элемент в нашу коллекцию:

```
numbers.add(5)
;
```

Добавление объекта в середину связанного списка

LinkedList позволяет добавить элемент в середину списка. Для этого используется метод **add(index, element)**, где *index* — это место в списке, куда будет вставлен элемент element.

Как и метод **add(element)**, данный метод вызывает несколько других методов. Сначала осуществляется проверка значения *index*, которое должно быть положительным числом, меньшим или равным размеру списка. Если *index* не удовлетворит этим условиям, то будет сгенерировано исключение *IndexOutOfBoundsException*.



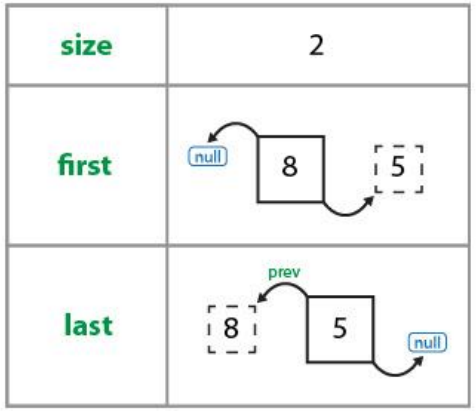


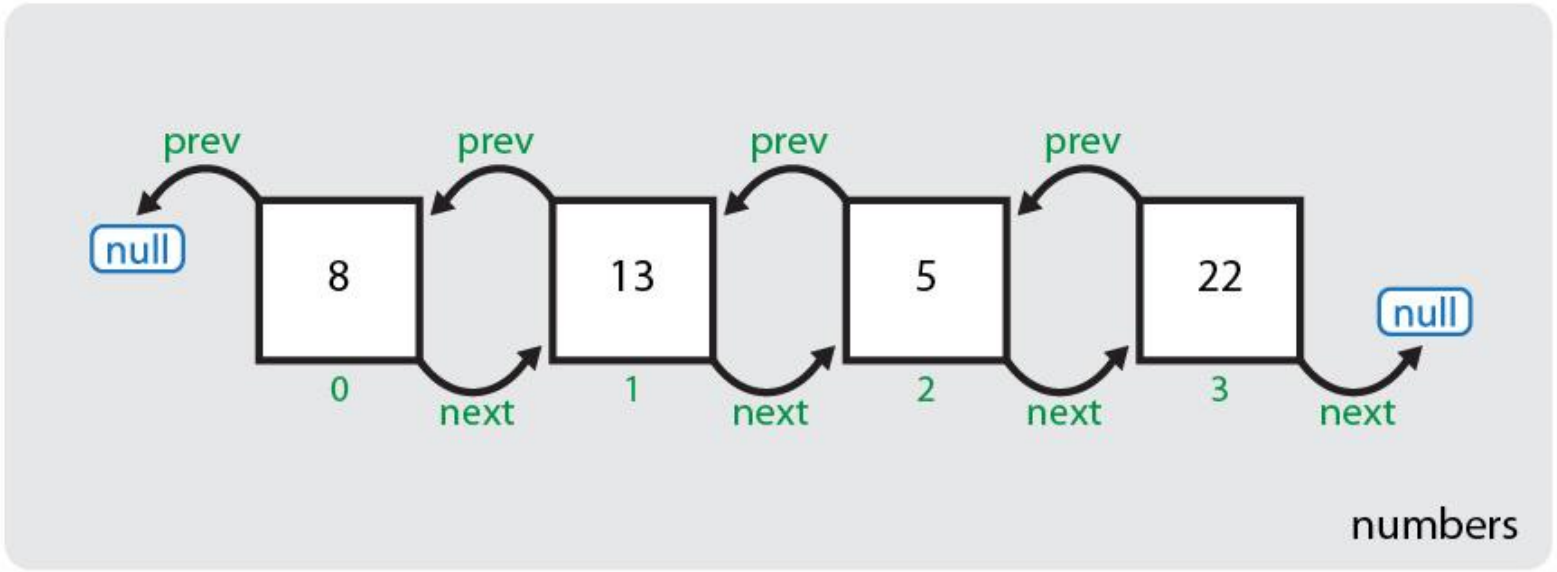
Рис. 6: Добавление объекта в середину связанного списка (этап 1)

Удаление объекта из списка

<table>
<tr><th>Метод</th><th>Удаляемый элемент</th><th>Если найден</th><th>Если не найден</th></tr>
<tr><td>poll()</td><td rowspan="4">первый</td><td rowspan="7">возвращает удаленный элемент</td><td rowspan="2">возвращает null</td></tr>
<tr><td>pollFirst()</td></tr>
<tr><td>remove()</td><td rowspan="2">NoSuchElementException</td></tr>
<tr><td>removeFirst()</td></tr>
<tr><td>pollLast()</td><td rowspan="2">последний</td><td>возвращает null</td></tr>
<tr><td>removeLast()</td><td>NoSuchElementException</td></tr>
<tr><td>remove(index)</td><td>элемент в указанной позиции</td><td>IndexOutOfBoundsException</td></tr>
<tr><td>remove(object)</td><td rowspan="2">первое вхождение указанного объекта</td><td rowspan="3">возвращает true</td><td rowspan="3">возвращает false</td></tr>
<tr><td>removeFirstOccurrence(object)</td></tr>
<tr><td>removeLastOccurrence(object)</td><td>последнее вхождение указанного объекта</td></tr>
</table>

Табл. 1: Методы LinkedList для удаления элемента

Рассмотрим удаление элемента из связанного списка
по его значению.
Удалим элемент со значением 5 из представленного



size	4
first	null 8 13
last	5 22 null

Рис. 8: Состояние связанного списка перед удалением элемента

Операции над LinkedList

- Методы добавления **addFirst()**, **addLast()**, **addAll()**.
- Получить объект **get()**, **getFirst()**, **getLast()**.
- Для изменения элемента используется метод **set()**.
- Для удаления - **remove()** и **clear()**.
- Метод **contains()** проверяет наличие конкретного элемента.
- Метод **element()** идентичен **getFirst()** и присутствует из-за принадлежности к интерфейсу **Queue**, который используется в **LinkedList**.
- Метод **peek()**, идентичен **getFirst()**, но не вызовет ошибку, если объект не существует, а просто вернёт **null**.
- Методы **peekFist()** и **peekLast()**, которые аналогичны **peek()**.

ArrayList vs. LinkedList