LinkedList

ПОДГОТОВИЛ: АЛИШЕР ХАМИДОВ

Основные моменты:

План лекции

1. Какие неудобства есть у массивов?
2. Как создать LinkedList?
3. Преимущества LinkedList
4. Как добавить элемент?
5. Как получить элемент?
6. Как заменить элемент?
7. Как удалить элемент?
8. Как удалить все элементы?
9. Циклы
10. Важные замечания
11. LinkedList vs ArrayList,оценка сложности

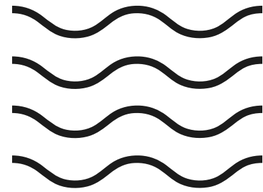
Какое самое большое неудобство массива?

LinkedList - работает "снаружи" так же, как ArrayList

resizable (изменяемый размер)

+

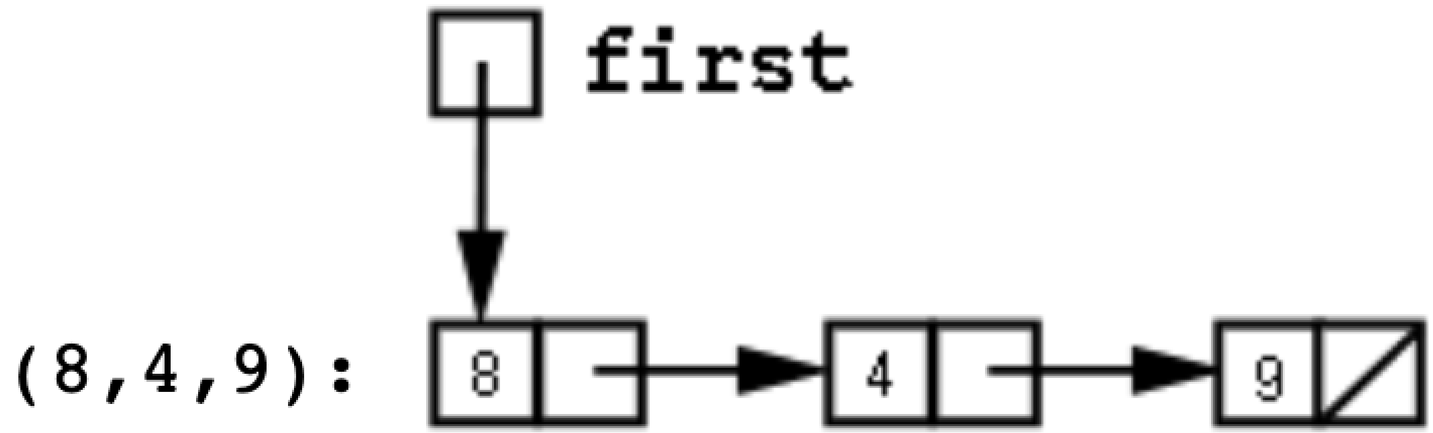
удобные методы

LinkedList - работает "снаружи" также, как ArrayList

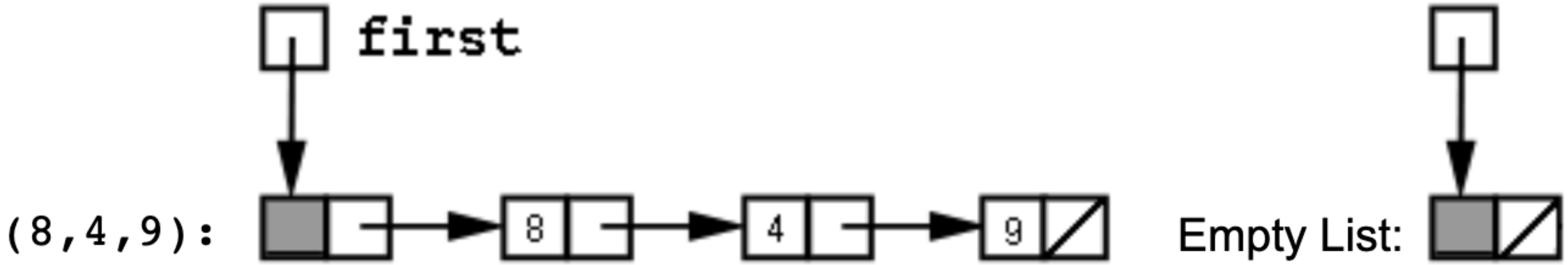
Вся разница между ними в том, что у них внутри

Singly-linked list

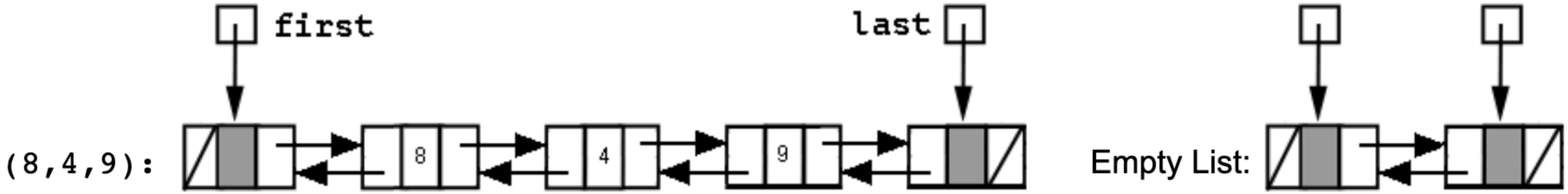


Empty List:

Singly-linked с болванкой на первом месте



Doubly-linked list



Как создать?

```
LinkedList<String> names = new LinkedList<>();
```

```
LinkedList<Тип хранимого значения> имя = new LinkedList<>
();
```

Как создать?

```
LinkedList<String> names = new LinkedList<>
(previousNames);
```

(Создание из другого листа)

Как добавить элемент?

Чтобы добавить элемент используйте метод **add().**

```
LinkedList<String> names = new LinkedList();
names.add("Bob");
names.add("John");
System.out.println(names); // [Bob, John]
```

Как получить элемент?

Чтобы получить элемент из LinkedList нужно воспользоваться методом **get()** и передать в него индекс.

```
String name = names.get(0);
System.out.println(name); // Bob
```

Как заменить элемент?

Чтобы заменить элемент в LinkedList нужно воспользоваться методом **set()** и передать в него индекс элемента, который мы хотим заменить и новое значение.

```
names.set(0, "Tim");
System.out.println(names); // [Tim, John]
```

Как удалить элемент?

Чтобы удалить элемент в ArrayList нужно воспользоваться методом **remove()** и передать в него **индекс** элемента, который мы хотим удалить.

```
LinkedList<String> names = new LinkedList();
names.add("Bob");
names.add("John");
names.remove(1);
System.out.println(names); // [Bob]
```

Как ещё удалить элемент?

Чтобы удалить элемент в LinkedList нужно воспользоваться методом **remove()** и передать в него **элемент**, который мы хотим удалить.

```
LinkedList<String> names = new LinkedList();
names.add("Bob");
names.add("John");
names.remove("Bob");
System.out.println(names); // [John]
```

Как удалить все элементы?

Чтобы удалить все элементы из LinkedList нужно воспользоваться методом **clear()**.

```
LinkedList<String> names = new LinkedList();
names.add("Bob");
names.add("John");
names.clear();
System.out.println(names); // []
```

Как узнать сколько элементов?

Чтобы узнать количество элементов используется метод **size()**.

```
LinkedList<String> names = new LinkedList();
names.add("Bob");
names.add("John");
System.out.println(names.size()); // 2
```

Как пройтись по LinkedList циклом for?

```
ArrayList<String> names = new ArrayList();
names.add("Bob");
names.add("John");
names.add("Ann");
for(int i = 0; i < names.size(); i++){
    System.out.println(names.get(i));
}
```

Как пройтись по LinkedList циклом for-each?

```
LinkedList<String> names = new LinkedList();
names.add("Bob");
names.add("John");
names.add("Ann");
for (String name : names) {
    System.out.println(name);
}
```

ArrayList vs LinkedList

get/set access

accessing an element (**get/set**) at an index is **O(1)**; for this reason the ArrayList implements the RandomAccess interface

accessing an element **(get/set)** at an index is **O(n)** because we must proceed through a sequence of node pointers to get to the correct position.

ArrayList vs LinkedList

add/remove at first and last positions

- **add at the last position** is O(1) in general; an additional O(n) cost is incurred when the array's capacity is increase;
- if the capacity is maintained efficiently, the overall average cost is still O(1)
- remove at the last position is always O(1)
- add/remove at the first position is always O(n) because the entire array is shifted

- add/remove at the first or last positions is O(1) (with doubly-linked lists)

ArrayList vs LinkedList

add/remove at arbitrary position

add/remove at arbitrary position is O(n) due to shifting

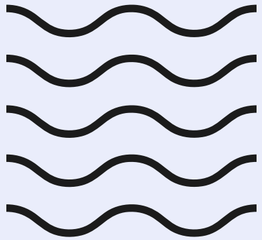
add/remove at arbitrary position is O(n) due to sequencing through pointers

Важное замечание

Элементы LinkedList на самом деле являются **объектами**. В примерах выше мы создавали объекты класса String.

Вы же помните, что String в Java является объектом (не примитивный тип).

Чтобы использовать другие типы, например int, вам нужно воспользоваться подходящим классом-оберткой: Integer.



Для других типов используйте: Boolean для boolean, Character для char, Double для double, и т.д.

Документация

По ссылке ниже вы сможете ознакомится с документацией и подробнее прочитать про все методы класса LinkedList

https://docs.oracle.com/javase/7/docs/api/java/util/LinkedList.html