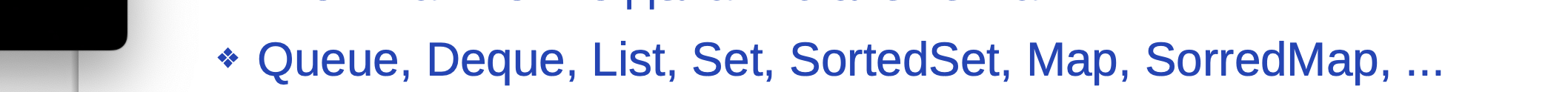


1)Коллекции. Сортировка элементов коллекции. Интерфейсы java.util.Comparable и java.util.Comparator.

Основные принципы, по которым создавался коллектион фреймворк.



Очередь, направленная очередь, список, мнодество мэп и так далее

Реализации- конкретные классы, которые эти коллекции реализуют определяют эти коллекции по следующим параметрам:

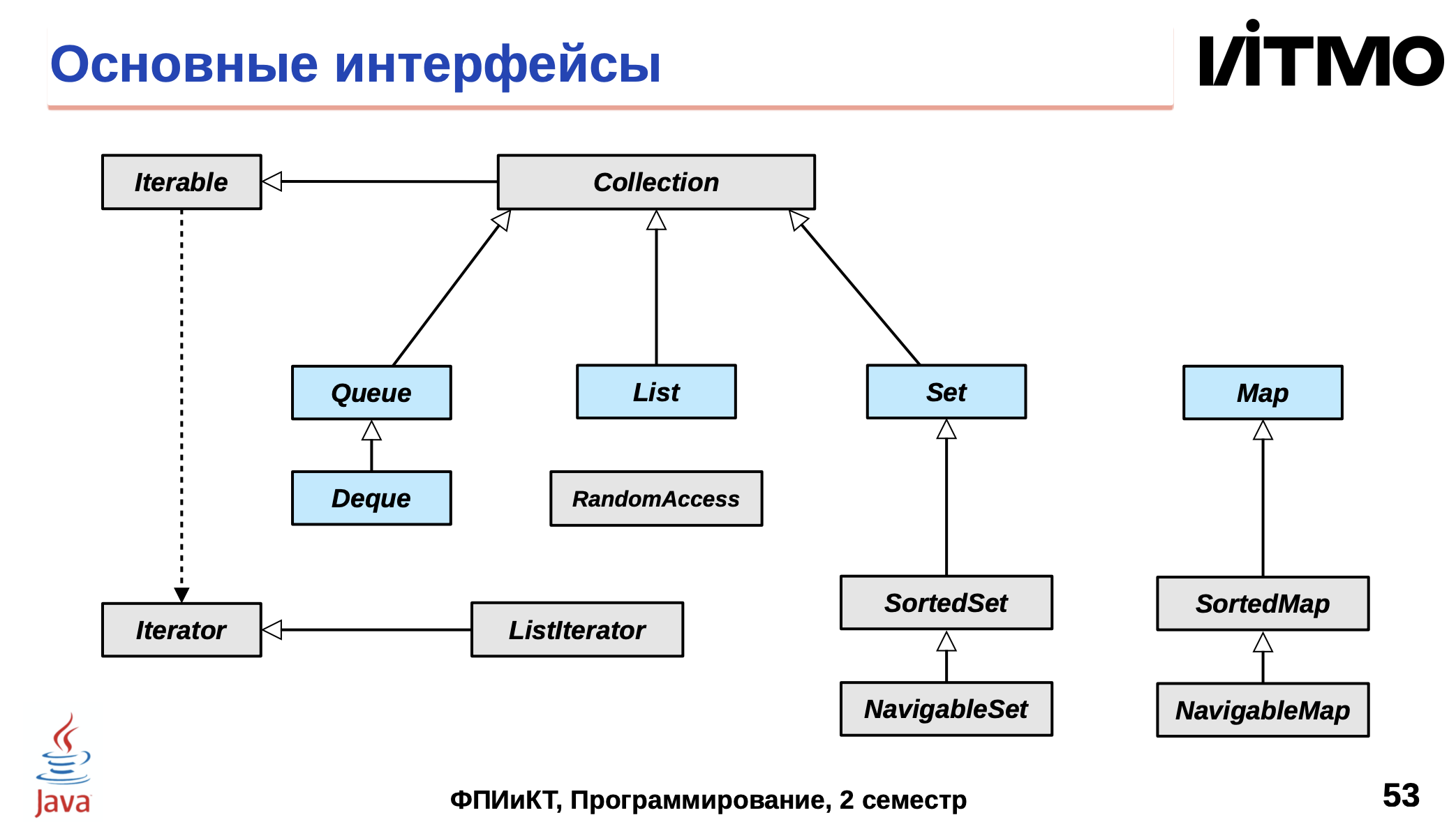
- скоррсть(на сколько быстро работает коллекция с операциями)

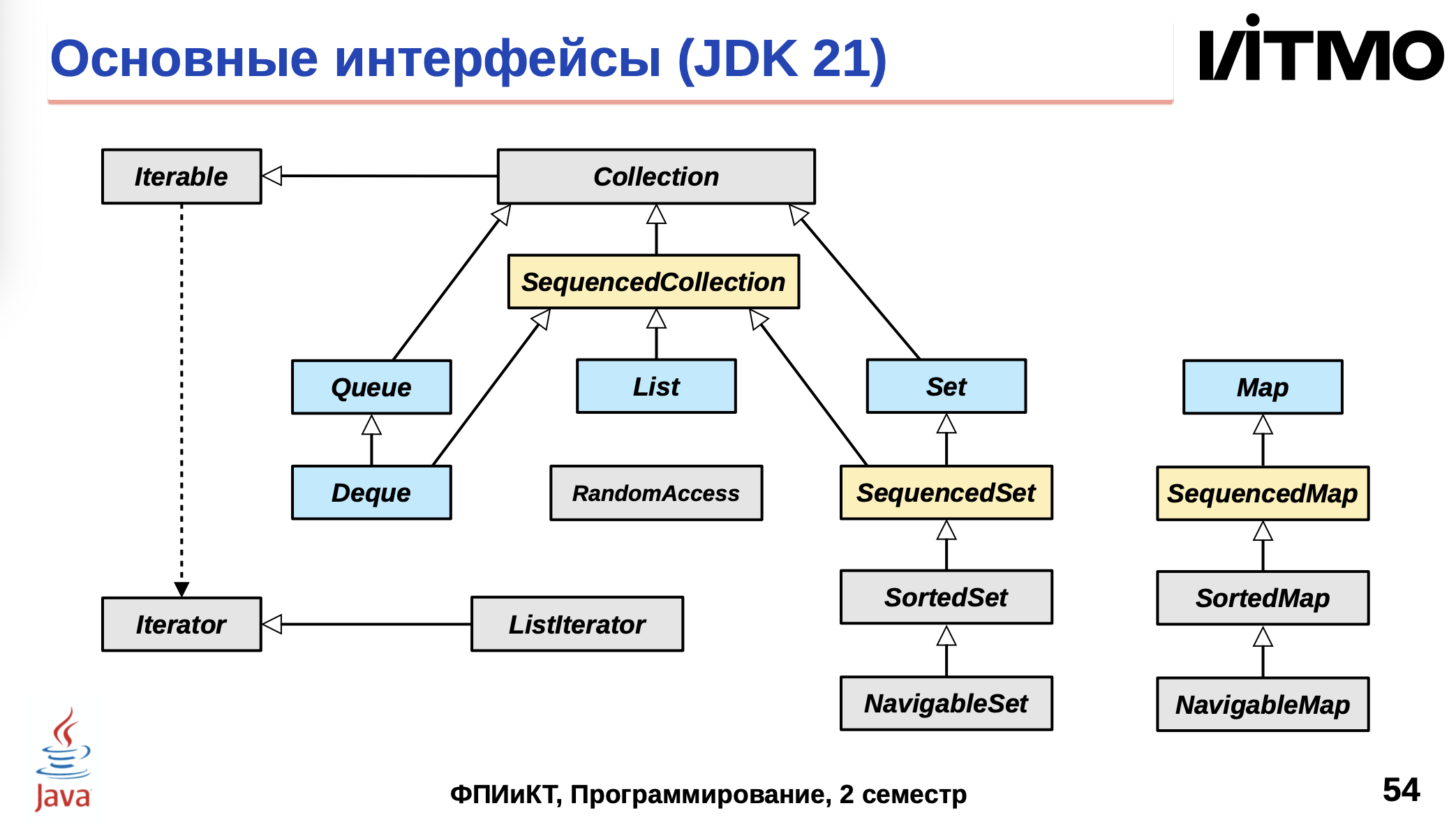
- сколько памяти она занимает

- на основе какой структуры построена коллекция



И применяется Big О-нотация





Интерфейсы определяют, по которым работают коллекции. Задают контракт того, как мы работаем.

Collection – коллекция элементов

SequencedCollection – упорядоченная коллекция

Map – отображение, массив, пары “ключ-значение”

Set – множество уникальных элементов(не может содержаться 2 одинаковых элемента)

SequencedMap – упорядоченное отображение

SequencedSet – упорядоченное множество

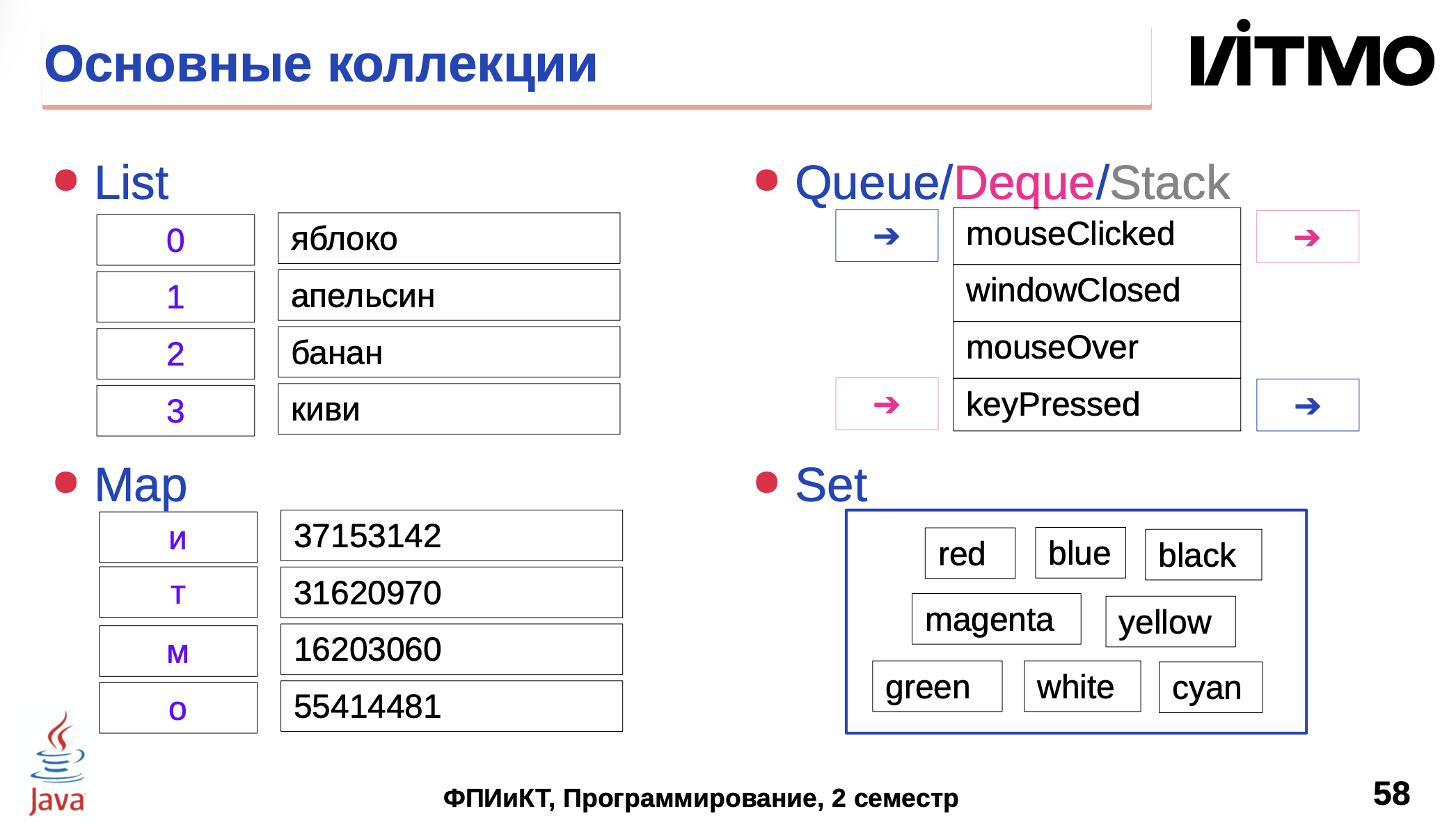
SortedMap / Set– отсортированное отображение(отсортированные ключи)/ множество

NavigableMap /set – “Обходимое” отображение/ множество

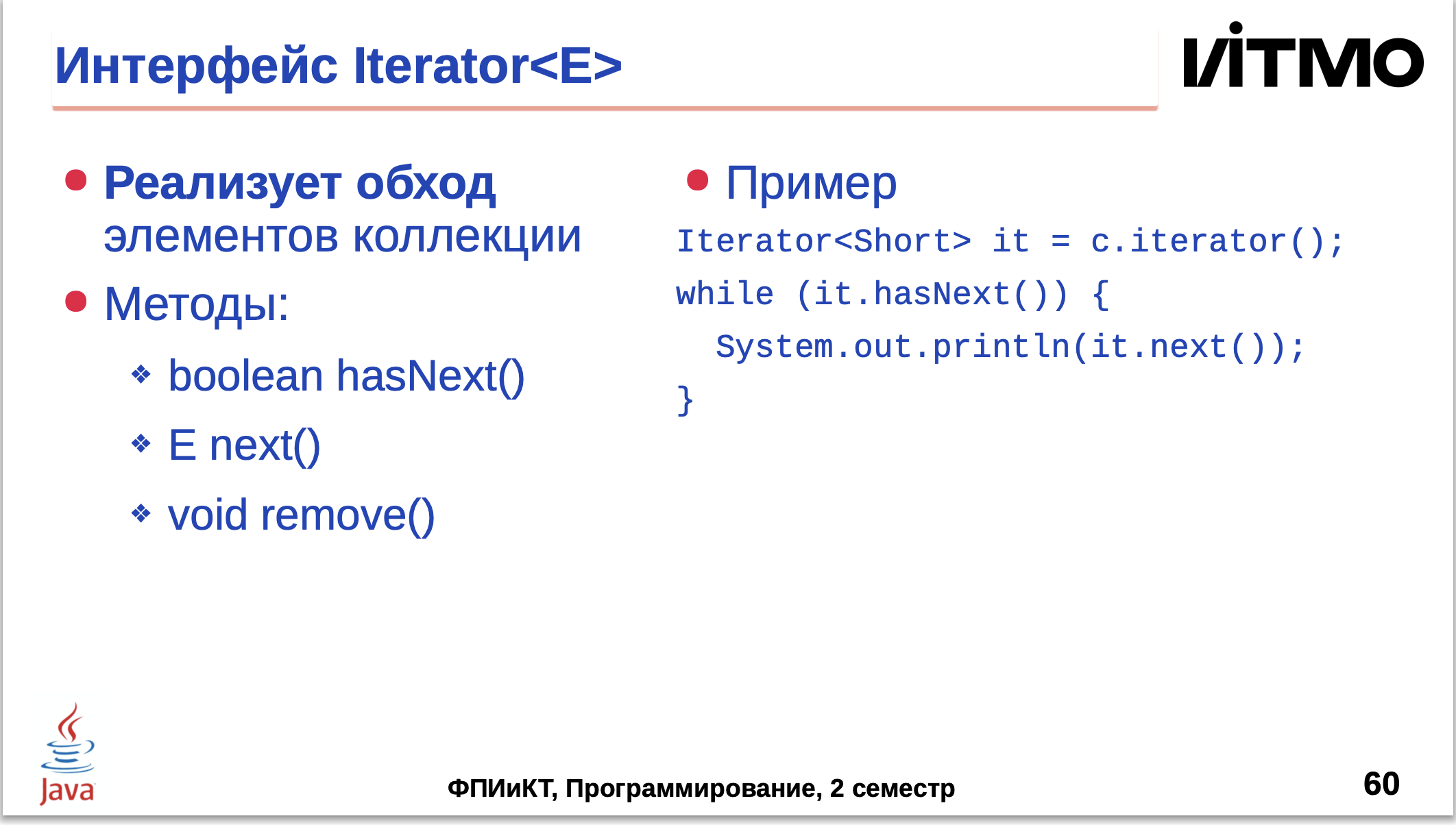
List – индексированный списокот 0 и так далее)

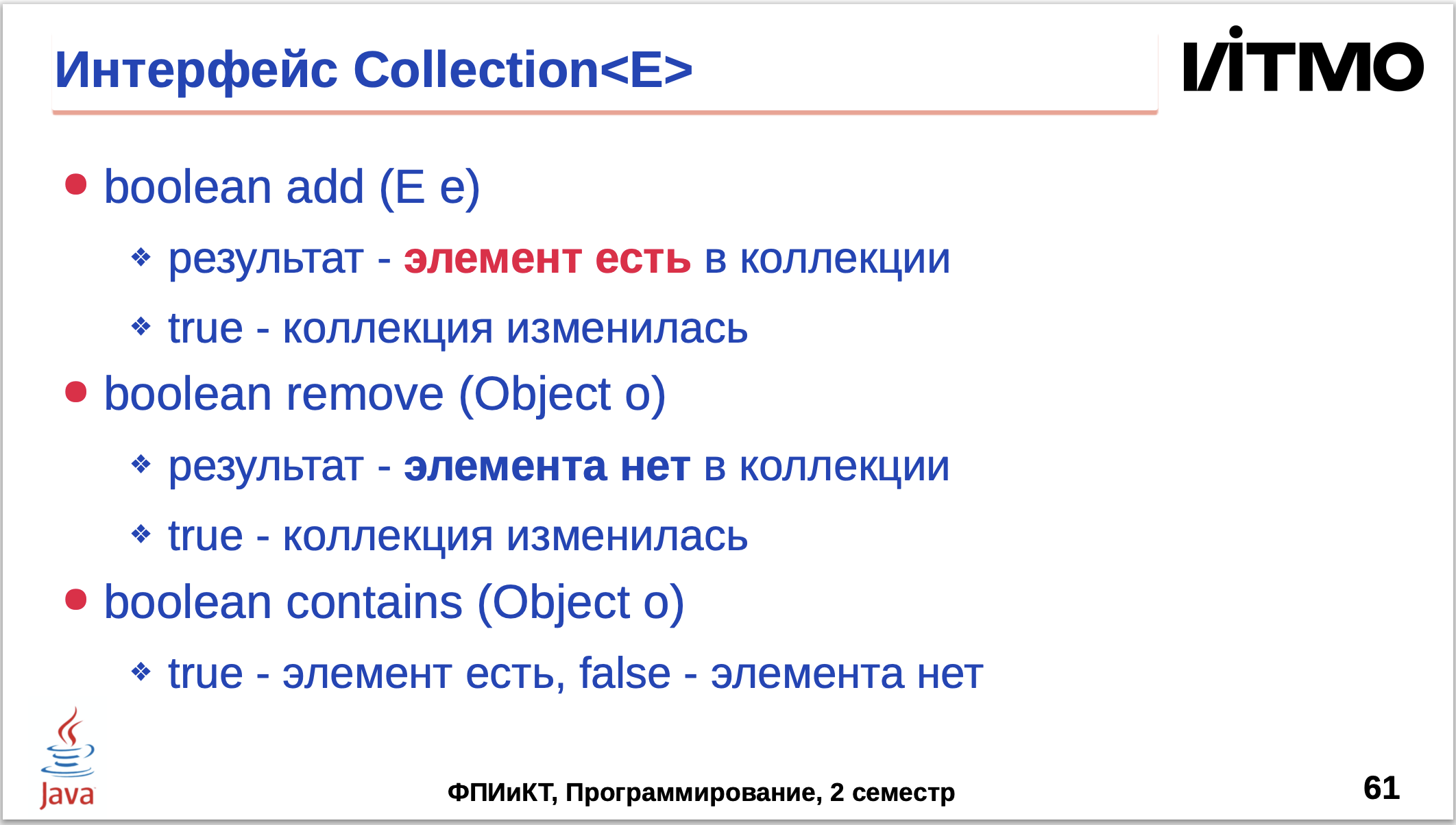
Queue – очередь

Deque -индексированная очередь

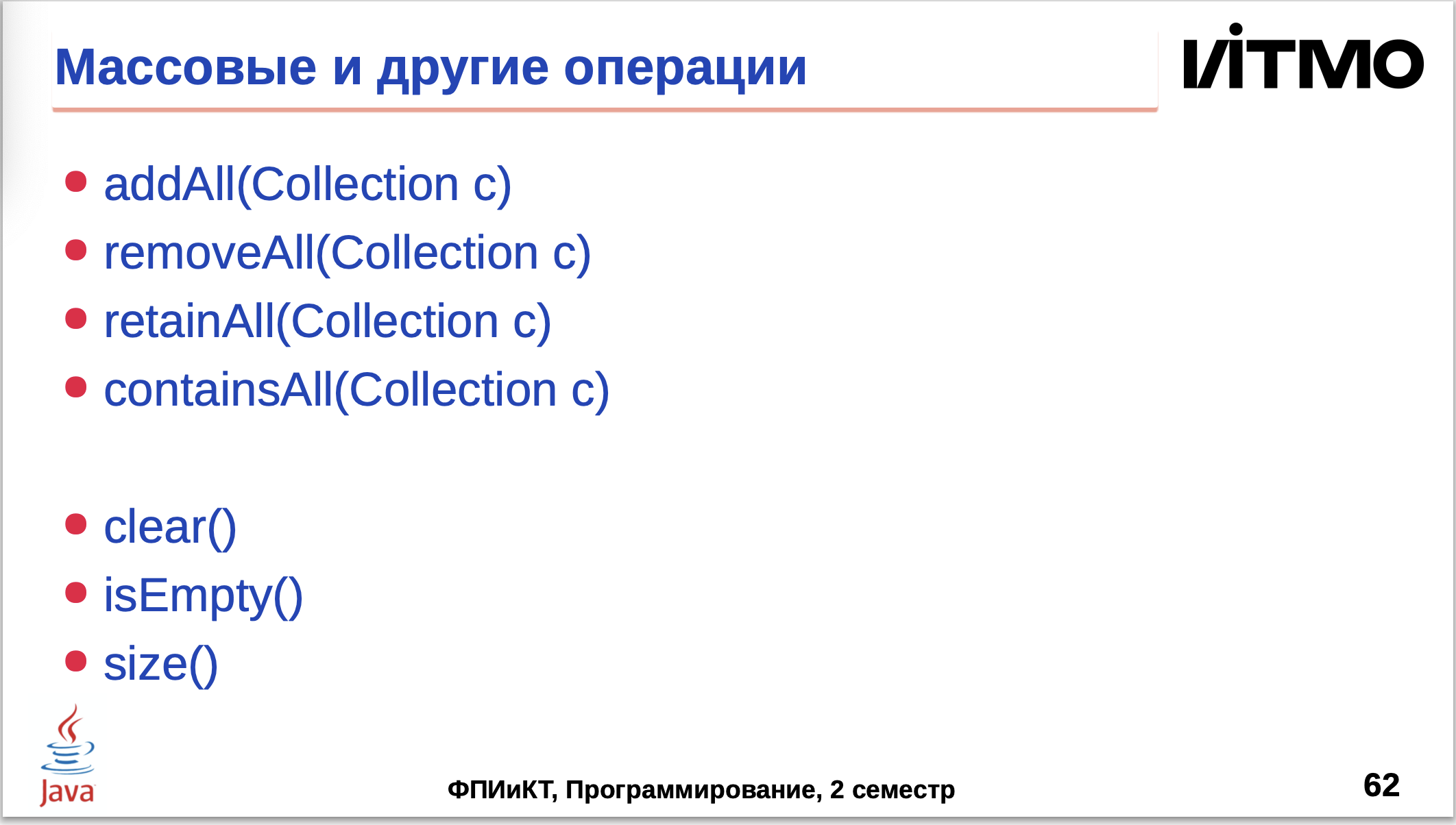


Пока метод hasNext – возвращает true, мы може получать очередной элеимент





Add- обеспечивает наличие элемента в коллекции. True – коллекция измениоась





Capacity – максимальный размер элемента(сколько элементов в эту коллекцию можно поместить), size – сколько элементов сейчас в этой коллекции.

Для этого списка быстро работают операции доступа по индексу(список с произвольным доступом).

Когда мы уже дошли до предела этого массива, нам надо его расширить.

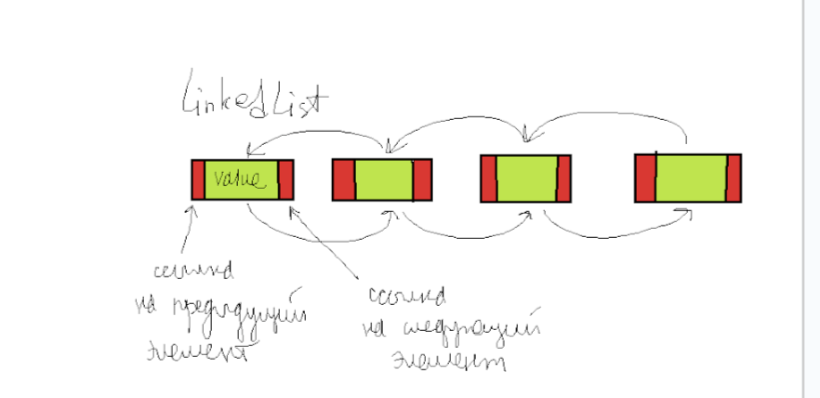
Как это? – все элементы масссива копируются в новый массив бОльшего размера, далее старый массив мы забываем (уходит в сборщик мусора), а с новым массивом начинаем работать.

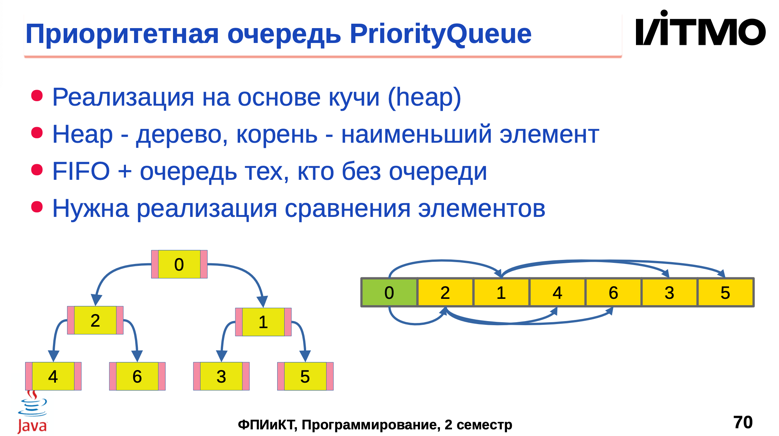
тот же самый лист, доступ по индексу тот же самый, но РЕАЛИЗАЦИЯ НА ОСНОВЕ ДВУСВЯЗНОГО СПИСКА.

Быстрый последовательный доступ.

Быстрое удаление и добавление

Но если нам нужно достать 1000 элемент, то мы долго будем до него добираться.

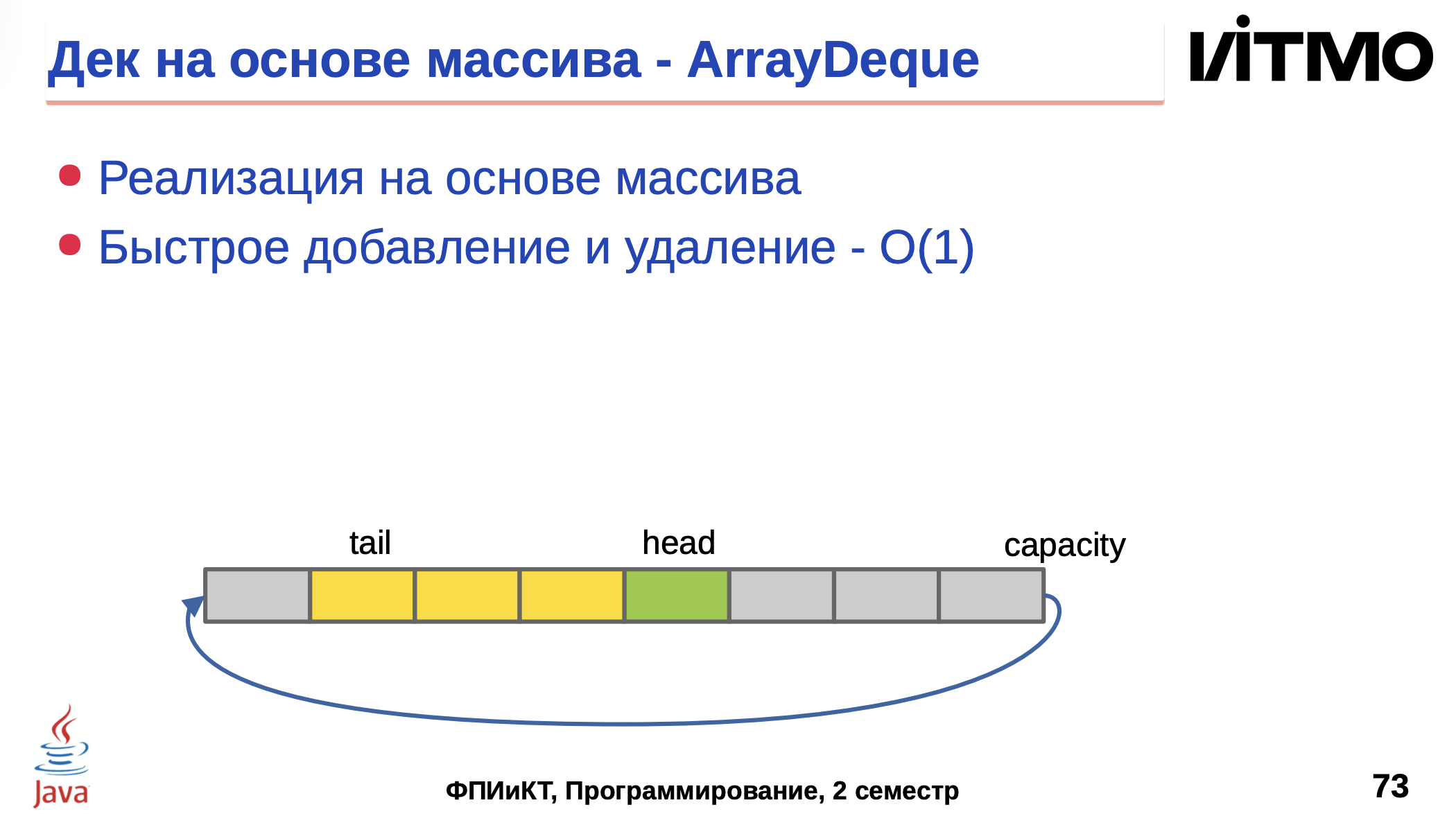


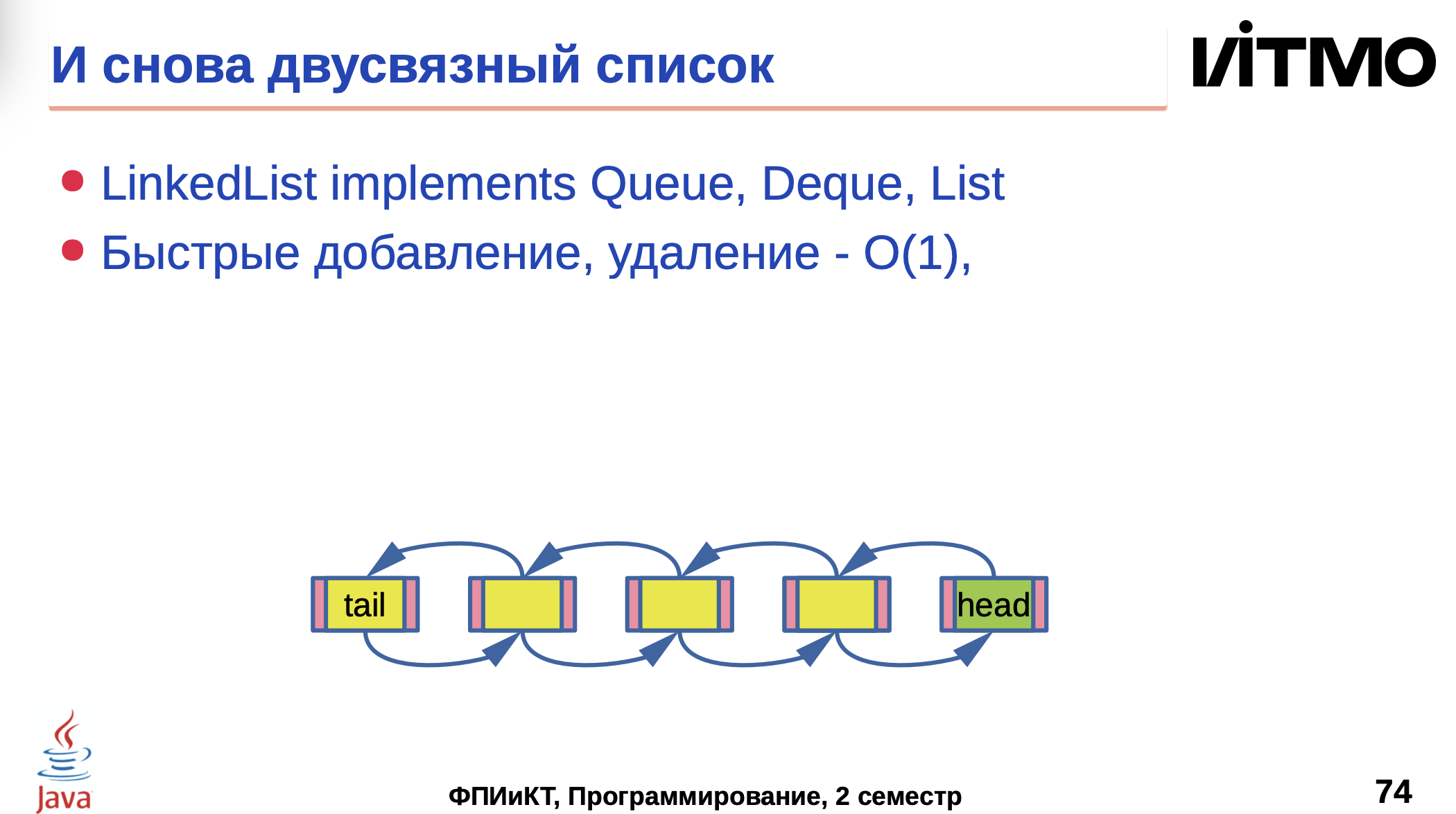
fifo – first in, first out

Чем меньше значение элемента, тем приоритет выше. Организована на основе кучи (дерево, в основе которого либо больший, либо меньший элемент. В java – меньший.

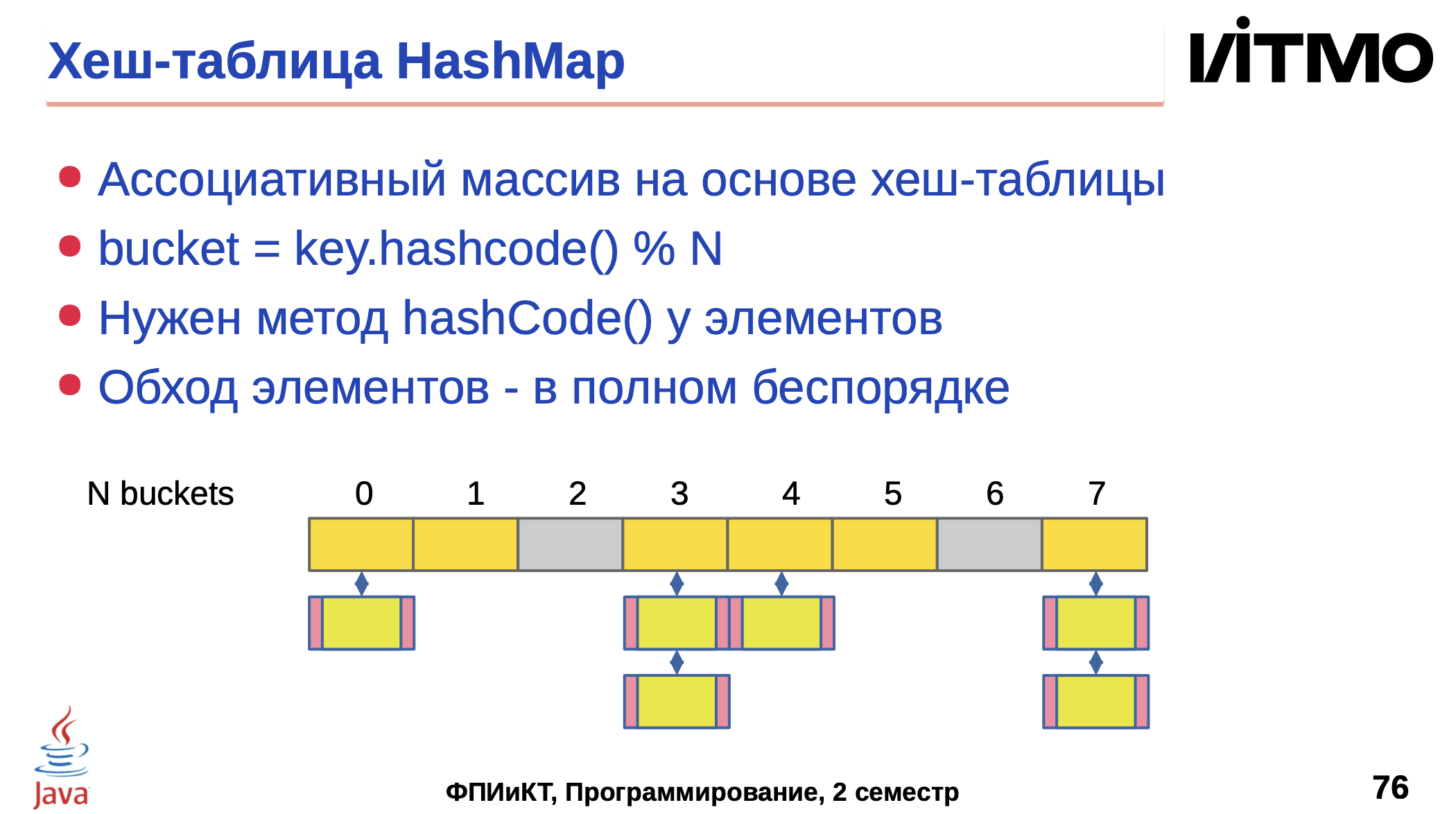
В корне- элемент, который первым будет выходить.

Чтобы реализовывать приоритетную очередь нужна какая-то реализация стравнения.





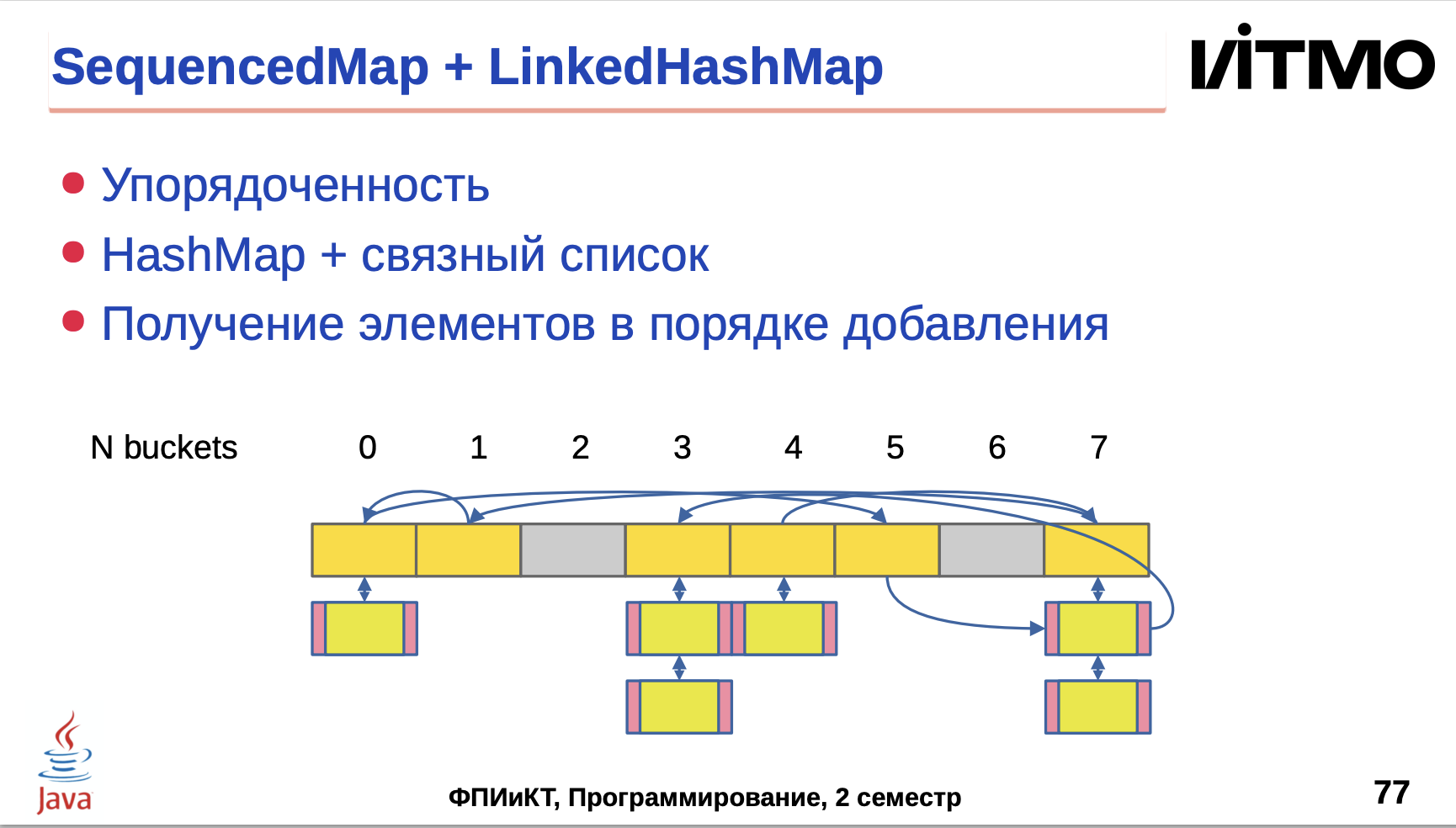




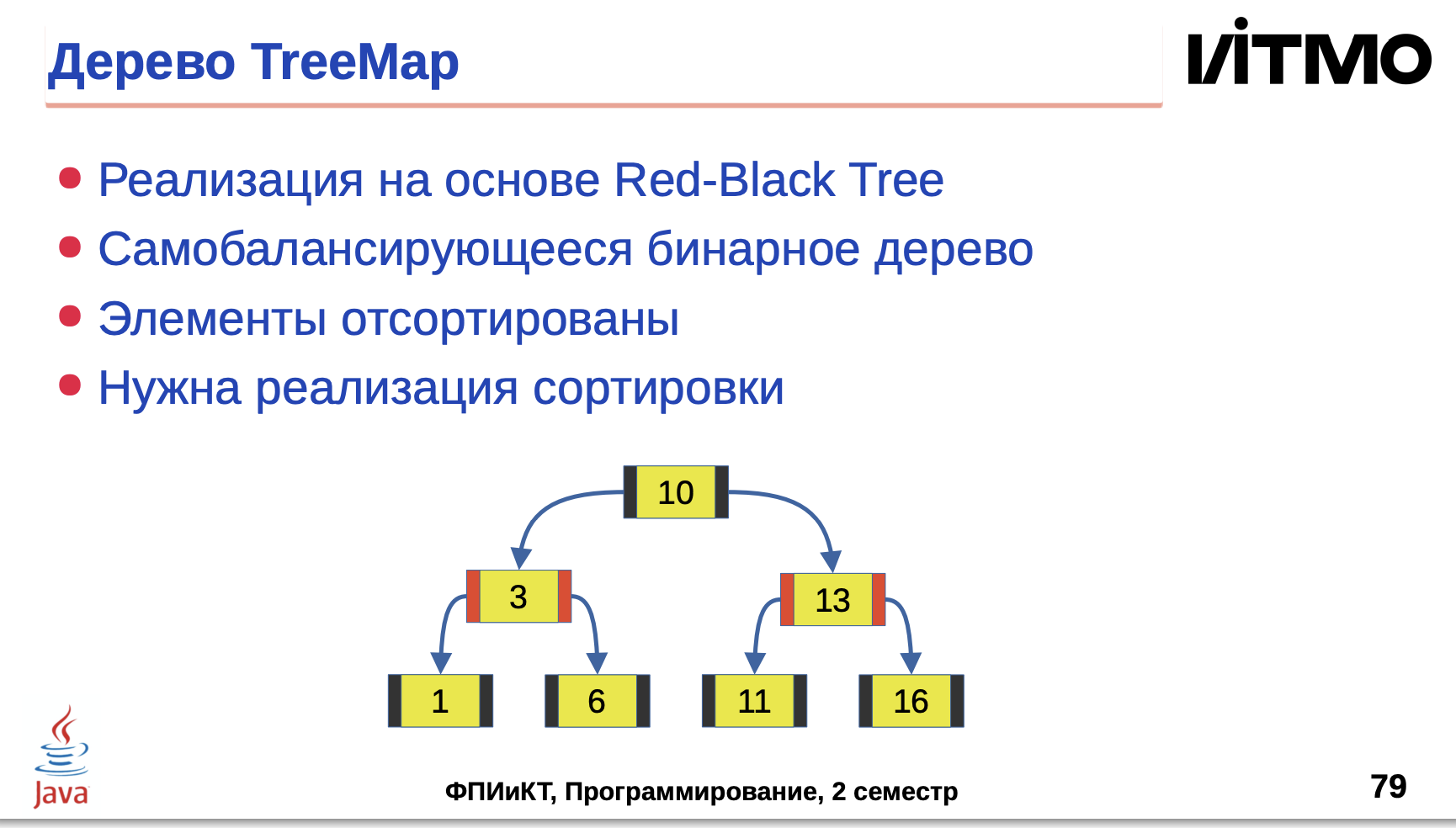
У нас есть некоторое кол-во бакетов. После чего мы берем ключ элемента, который пришел, берем hashCode этого ключа, получившийся hashCode мы по модулю делим на кол-во бакетов(остаток от деления берем) и получаем номер бакета, в котором элемент должен храниться.

Если там есть элемент, то мы кладем в дусвязный список, если элемента нет, то просто появится.

HashMap – нужен для того, чтобы быстро получать доступ к элементу по ключу.



Тот же самый Map, только элементы упорядочены. Элементы, которые поступают следующими, на них ведет ссылка из предыдущего элемента.



Двоичное дерево. Двоичное- потому что у каждого элемента есть 2 потомка. Правый и левый. Один из низ больше, другой меньше.

Стандартно: мы берем элемент, дальше мы получаем очередной элемент, если он меньше нашего, то мы его слева добавляем, если меньше нашего, то справа.

Проблема дерева: если элементы были изначально отсортированы, то получится так, что на одной стороне деерева будет длинная ветка, а на другой ее может и не быть, при жтом дерево будет не сбалансированным и будут потери производительности.

Красно-черно дерево жти проблемы решает. Во время построения дерева, дерево пытается ограничить и уравнять величину своих ветвей.

Задаются ограничения:

В алгоритме(условно) красным и черным цветом раскрашиваются узлы дерева.

Причем задаются ограничения: у красного узла не может быть красных ветвей, черный узел должен быть последним, в корне должен быть черный узел и нельзя 2 красных подряд.

Comparable – импоементируется