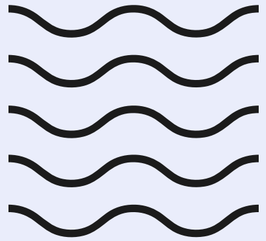
TreeSet

ПОДГОТОВИЛ: АЛИШЕР ХАМИДОВ



Основные моменты:

План лекции

1. Как создать?
2. Методы
3. Важные замечания
4. Внутреннее устройство
5. Дополнительная информация

Что такое TreeSet?

Ваши ассоциации

TreeSet это отсортированная коллекция, которая расширяет класс AbstractSet и имплементирует интерфейс NavigableSet

Некоторые важные черты TreeSet:

- Хранит уникальные элементы
- Не сохраняет порядок добавления элементов
- Сортирует элементы в порядке возрастания

Как создать?

```
Set<String> treeSet = new TreeSet<>(new Comparator(){//...});
```

При создании можно передать в конструктор компаратор.

Как добавить элемент?

```
TreeSet<String> names = new TreeSet<>();
names.add("John");
names.add("Bob");
```

Метод возвращает true, если элемент добавлен.
Элемент будет добавлен, только если его еще нет.

Как проверить наличие элемента?

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

names.add("John");
names.add("Bob");
names.contains("Bob"); // возвращает true
```

Как удалить элемента?

```
names.add("John");
names.add("Bob");
names.remove("John");
System.out.println(names); //[Bob]

// если элемент был удален, метод возвращает true
```

.

Как удалить элемента?

```
names.add("John");
names.add("Bob");
names.remove("John");
System.out.println(names); //[Bob]

// если элемент был удален, метод возвращает true
```

.

Как удалить все элементы?

```
names.add("John");
names.add("Bob");
names.clear();
System.out.println(names); //[]
```

Как получить первый элемент?

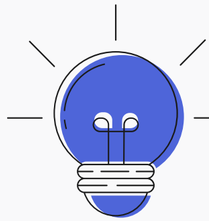
```
names.add("John");
names.add("Bob");
names.first(); // Bob
```

Как получить последний элемент?

```
names.add("John");
names.add("Bob");
names.last(); // John
```

Как получить подмножество?

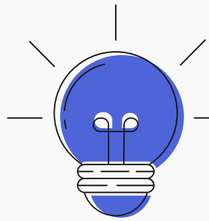
```
SortedSet<Integer> treeSet = new TreeSet<>();
treeSet.add(1);
treeSet.add(2);
treeSet.add(3);
treeSet.add(4);
treeSet.add(5);

Set<Integer> subSet = treeSet.subSet(2, 4); // [2, 3, 4].
```



Нижний интервал - включительно, верхний - не включительно

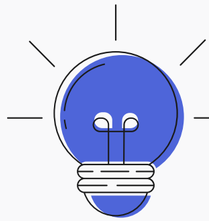
Как получить подмножество?

```
SortedSet<Integer> treeSet = new TreeSet<>();
treeSet.add(1);
treeSet.add(2);
treeSet.add(3);
treeSet.add(4);
treeSet.add(5);

Set<Integer> subSet = treeSet.headSet(4);
// возвращает все элементы меньше указанного
System.out.println(subSet); //[1, 2, 3]
```



Как получить подмножество?

```
SortedSet<Integer> treeSet = new TreeSet<>();
treeSet.add(1);
treeSet.add(2);
treeSet.add(3);
treeSet.add(4);
treeSet.add(5);

Set<Integer> subSet = treeSet.tailSet(4);
// возвращает все элементы, которые больше или равны указанному
System.out.println(subSet); //[4, 5]
```



Важное замечание

В TreeSet нельзя хранить null!

```
SortedSet<Integer> treeSet = new TreeSet<>();
treeSet.add(1);
treeSet.add(2);
treeSet.add(null); // NullPointerException
```

Важное замечание

Поскольку объекты располагаются в отсортированном порядке в TreeSet, требуется указать по какой логике идет сортировка.

Все элементы TreeSet должны либо имплементировать интерфейс **Comparable**, либо следует передать в конструктор TreeSet экземпляр класса **Comparator**.