Practice6

Везде в коде, при реализации функциональных интерфейсов использовать лямбда выражения или ссылки на методы.

Помните, что контейнеры Hash/TreeSet, Hash/TreeMap не контролируют потенциальное появление дубликатов при изменении никак не отслеживают изменения, то есть, если изменение состояния ключа какой ни будь пары может произойти, то контейнер это никак не отслеживает.

Задание 5

Пакет: **ua.nure.name.Practice6**
Класс: **Part5**, **Tree**

Создать generic класс **Tree**, который реализует структуру данных "двоичное дерево поиска". Контейнерные классы не использовать.

Tree.java

```
public class Tree<E extends Comparable<E>> {
        // добавляет элемент в контейнер
        // если в контейнере есть элемент равный по compareTo добавляемому,
        // то добавления не происходит и метод возвращает false
        // в противном случае элемент попадает в контейнер и метод возвращает true
        // первый добавляемый элемент становится корнем дерева
        public boolean add(E element) {...}

        // добавляет все элементы из массива в контейнер (вызов в цикле метода add, см. выше)
        public void add(E[] elements) {...}

    // удаляет элемент из контейнера
        // если удаляемого элемента в контейнере нет, то возвращает false
        // в противном случае удаляет элемент и возвращает true
        // ВАЖНО! при удалении элемента дерево не должно потерять свойства
бинарного дерева поиска
        public boolean remove(E element) {...}

        // распечатывает дерево, так чтобы было видно его древовидную структуру,
см. ниже пример
        public void print() {...}
```

```
        // вложенный класс, объекты этого класса составляют дерево
        private static class Node<E> {...}
}
```

Код

```
Tree<Integer> tree = new Tree<>();

System.out.println(tree.add(3));
System.out.println(tree.add(3));

System.out.println("~~~~~~~");
tree.add(new Integer[] {1, 2, 5, 4, 6, 0});
tree.print();

System.out.println("~~~~~~~");
System.out.println(tree.remove(5));
System.out.println(tree.remove(5));

System.out.println("~~~~~~~");
tree.print();
```

Вывод

```
true
false
~~~~~~~
        0
  1
        2
3
        4
  5
        6
~~~~~~~
true
false
~~~~~~~
        0
    1
        2
4
    5
```

6

Продемонстрировать работу приложения (**Part5.main**).