Programowanie w języku JAVA

Laboratorium 2

Marcin Godfryd

Grupa 11

Triple.java

```java
import java.util.Comparator;
import java.util.Objects;

public class Triple<T1 extends Comparable<T1>, T2 extends Comparable<T2>,
T3 extends Comparable<T3>>
        implements Comparable<Triple<T1, T2, T3>> {

    private final T1 first;
    private final T2 second;
    private final T3 third;

    public Triple(T1 first, T2 second, T3 third) {
        this.first = first;
        this.second = second;
        this.third = third;
    }

    public T1 getFirst() {
        return first;
    }

    public T2 getSecond() {
        return second;
    }

    public T3 getThird() {
        return third;
    }

    @Override
    public String toString() {
        return "Triple<" + first + ", " + second + ", " + third + ">";
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof Triple)) return false;

        Triple<?, ?, ?> triple = (Triple<?, ?, ?>) o;

        return first.equals(triple.first) &&
                second.equals(triple.second) &&
                third.equals(triple.third);
    }

    @Override
    public int hashCode() {
        return Objects.hash(first, second, third);
    }

    @Override
    public int compareTo(Triple<T1, T2, T3> o) {
        int compareFirst = first.compareTo(o.first);
        if (compareFirst != 0) return compareFirst;

        int compareSecond = second.compareTo(o.second);
        if (compareSecond != 0) return compareSecond;
        return third.compareTo(o.third);
    }

    public static <T1 extends Comparable<T1>, T2 extends Comparable<T2>, T3
extends Comparable<T3>> Comparator<Triple<T1, T2, T3>>
reverseOrderComparator() {
        return new Comparator<Triple<T1, T2, T3>>() {
            @Override
            public int compare(Triple<T1, T2, T3> triple1, Triple<T1, T2,
T3> triple2) {
                int compareFirst =
triple2.getFirst().compareTo(triple1.getFirst());
                if (compareFirst != 0) {
                    return compareFirst;
                }

                int compareSecond =
triple2.getSecond().compareTo(triple1.getSecond());
                if (compareSecond != 0) {
                    return compareSecond;
                }

                return triple2.getThird().compareTo(triple1.getThird());
            }
        };
    }

}
```

Main.java

```java
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

public class Main {

    public static void main(String[] args) {

        Triple<Integer, Double, String> triple1 = new Triple<>(1, 2.5,
"A");
        Triple<Integer, Double, String> triple2 = new Triple<>(3, 2.5,
"B");
        Triple<Integer, Double, String> triple3 = new Triple<>(2, 3.25,
"C");
        Triple<Integer, Double, String> triple4 = new Triple<>(1, 3.3,
"D");
        Triple<Integer, Double, String> triple5 = new Triple<>(1, 2.5,
"A");

        System.out.println(triple1);

        System.out.println("Czy triple1 jest równy triple2? " +
triple1.equals(triple2));
        System.out.println("Czy triple1 jest równy triple5? " +
triple1.equals(triple5));

        List<Triple<Integer, Double, String>> list = new ArrayList<>();
        list.add(triple1);
        list.add(triple2);
        list.add(triple3);
        list.add(triple4);
        list.add(triple5);

        // Comparable
        Collections.sort(list);
        System.out.println("Posortowana lista:");
        for (Triple<Integer, Double, String> triple : list) {
            System.out.println(triple);
        }

        // Comparator
        Collections.sort(list, Triple.reverseOrderComparator());
        System.out.println("Lista posortowana odwrotnie:");
        for (Triple<Integer, Double, String> triple : list) {
            System.out.println(triple);
        }
    }
}
```