Arbeitsblatt Immutability: Lösung

Ein Immutable-Objekt muss all seine *beobachtbaren* Attribute so kapseln, dass sie nur im Konstruktor gesetzt werden und danach nur noch gelesen werden können.

Regeln:

1. Die Methoden der Klasse dürfen die (von aussen beobachtbaren) Daten der Klasse nicht ändern.
2. Die Felder müssen privat deklariert sein, oder final, falls es sich um Referenzen auf immutable Objekte oder Felder von primitiven Datentypen handelt.
3. Nur Getter-Methoden definieren, keine Setter Methoden
4. Alle Rückgabe-Werte von Getter-Methoden müssen kopiert werden, falls es nicht Referenzen auf immutable Objekte oder Resultate mit primitivem Datentyp sind.
5. Die Klasse muss final deklariert werden damit das Substitutionsprinzip erfüllt bleibt, d.h. damit nicht in der Unterklasse die Immutability-Regeln verletzt werden können.
6. Parameter in den Konstruktoren, die nicht immutable oder von primitiven Datentypen sind müssen geklont werden.
7. Klasse darf nicht von einer Basisklasse abgeleitet werden welche nicht-statische veränderbare Felder enthält.
8. Die `this` Referenz darf während der Konstruktion nicht sichtbar werden.

Implementation:

```
public final class ImmutableLine {
    private final Point start, end;
    public Line(Point start, Point end) {
        this.start = (Point) start.clone();
        this.end = (Point) end.clone();
    }
    public Point getStartPoint () {
        return (Point) start.clone();
    }
    public Point getEndPoint () {
        return (Point) end.clone();
    }
    public ImmutableLine withStartPoint(Point start) {
        return this.start.equals(start)) ? this
            : return new ImmutableLine(start, end);
    }
    public ImmutableLine withEndPoint(Point end) {
        return this.end.equals(end)) ? this
            : return new ImmutableLine(start, end);
    }
    @Override
    public Object clone() {    // Verletzt die generelle in der Spec
        return this;           // beschriebene Idee dass x.clone() != x
    }                          // gelten soll (muss nicht zwingend gelten).
    // Als Alternative kann die clone() Methode natürlich auch tatsächlich eine
    // Kopie erzeugen, oder die Methode wird ganz weggelassen (wie z.B. bei
    // java.lang.String).
    @Override
    public String toString() {
        return String.format("[Line: start=%s, end=%s]", start, end);
    }
}
```

Test:

```
Point p1 = new Point(1, 2);
Point p2 = new Point(3, 4);
Line l1 = new Line(p1, p2);
System.out.println(l1); p1.x = 5;
System.out.println(l1); // sollte nochmals das gleiche ergeben!!
```