NACHNAME:	**SEMESTER:** ☐ **M5** ☐ **M6** ☐ M3 ☐ M4 ☐ M7
VORNAME:	**VERTIEFUNG:** ☐ **FV** ☐ **IM**

Aufgabe 1: (15 Punkte)

Im gleichen `package` wie die aus den Übungen bekannte Klasse `Intervall` stehe die folgende, noch ganz rudimentäre Klasse `IntervallBisUnendlich`. Sie soll abgeschlossene, unendliche Intervalle der Form [a,∞) beschreiben:

```
public final class IntervallBisUnendlich {
     private double mVon;
}
```

Die Klasse soll nun schrittweise erweitert werden. `javadoc`-Kommentare sind dabei nicht verlangt.

a) Schreiben Sie den naheliegendsten und wichtigsten Konstruktor:

```
public IntervallBisUnendlich(double von) {
     mVon = von;
}
```

b) Schreiben Sie eine Methode `enthaelt` entsprechend der angegebenen Dokumentation:

```
/**
 * prueft, ob das angegebene (endliche) Intervall ganz
 * in diesem IntervallBisUnendlich enthalten ist.
 */
```

```
public boolean enthaelt(Intervall i) {
     return mVon <= i.untereGrenze();
}
```

c) Durch den Aufruf

```
System.out.println(a);
```

soll der abstrakte Wert des Objekts `a` vom Typ `IntervallBisUnendlich` ausgegeben werden. Nehmen Sie eine nötige Erweiterung der Klasse vor:

```
@Override
public String toString() {
    return "[" + mvon + ",unendlich)";
}
```

d) Um dem allgemeinen Vertrag von `Object` zu genügen, müssen weitere Methoden überschrieben werden. Geben Sie die vollständige Signatur (inklusive Rückgabetyp und Modifizierer) dieser Methoden an:

```
public boolean equals(Object o)
public int hashCode()
```

Aufgabe 2: (7 Punkte)

Die in **Aufgabe 1** entstandene Klasse werde durch folgende Methode erweitert:

```
public Intervall tuWasMit(Intervall i) {
    if (enthaelt(i)) return i;
    if (!i.enthaelt(mVon)) return null;
    return new Intervall(mVon, i.obereGrenze());
}
```

a) Welche Ergebnisse liefert diese Methode bei den nachstehenden Aufrufen:

dieses `IntervallBisUnendlich`	angegebenes `Intervall i`	Ergebnis
[0,∞)	[2,5]	**[2,5]**
[2,∞)	[0,3]	**[2,3]**
[4,∞)	[−2,1]	**null**
[6,∞)	[−4,6]	**[6,6]**

b) Um das Verhalten im Fehlerfalle zu verbessern, braucht nur eine (einzige) Zeile geändert zu werden. Geben Sie die neue Zeile an:

```
if (!i.enthaelt(mVon))
```

```
    throw new IllegalArgumentException();
```

Aufgabe 3: (8 Punkte)

Die in **Aufgabe 2** entstandene Klasse werde durch folgende Methode erweitert:

```
public void tuWasMit(IntervallBisUnendlich i) {
    if (i.mVon > mVon) mVon = i.mVon;
    else i.mVon = mVon;
}
```

a) Geben Sie an, wie die Variablen `a` und `b` beim folgenden Aufruf verändert werden:

```
a.tuWasMit(b);
```

	a	b
vorher	[3,∞)	[1,∞)
nachher	**[3,∞)**	**[3,∞)**
vorher	[2,∞)	[4,∞)
nachher	**[4,∞)**	**[4,∞)**

b) Welche zuvor vorhandene gute Eigenschaft der Klasse würde durch diese Methode zerstört?

Unveränderbarkeit

Durch welche Änderung in einer einzigen Zeile der Klasse könnte das versehentliche Zerstören dieser Eigenschaft verhindert werden? Geben Sie die neue Zeile an:

```
private final double mVon;
```