NACHNAME:	**SEMESTER:** ☐ **M6** ☐ **M7** ☐ M4 ☐ M5 ☐ M8
VORNAME:	**VERTIEFUNG:** ☐ **FV** ☐ **IM**

Aufgabe 1: (11 Punkte)

Die Klasse `GeometrischeReihe` beschreibt unendliche (konvergente oder divergente) geometrische Zahlen-Reihen:

$$1 + q + q^2 + q^3 + \ldots = \sum_{n=0}^{\infty} q^n \qquad (q \in \mathbb{R})$$

Die noch ganz rudimentäre Implementierung

```
public final class GeometrischeReihe {
      double q;
}
```

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

a) Schreiben Sie den naheliegendsten und wichtigsten Konstruktor:

```
public GeometrischeReihe(double quotient) {
      q = quotient;
}
```

b) Implementieren Sie (im Sinne der Abstraktion) die Methode `istKonvergent`:

```
public boolean istKonvergent() {
      return Math.abs(q) < 1.;
}
```

c) Schreiben Sie eine Methode `summe` entsprechend der angegebenen Dokumentation:

```
/**
 * liefert die Summe dieser unendlichen Reihe.
 * Bei einer divergenten Reihe wird eine
 * UnsupportedOperationException geworfen.
 */
```

```
public double summe() {
    if (!istKonvergent())
        throw new UnsupportedOperationException();
    return 1. / (1. - q);
}
```

Aufgabe 2: (8 Punkte)

Die Klasse `Reihen` stehe im gleichen `package` wie die in **Aufgabe 1** entstandene Klasse und enthalte folgende Methode:

```
public static void tuWasMit(GeometrischeReihe g) {
    if (!g.istKonvergent()) g.q = 1. / g.q;
    if (!g.istKonvergent()) g.q = 0.;
}
```

a) Aufgrund welches schwerwiegenden Mangels in dem bei **Aufgabe 1** vorgegebenen Code ist diese Implementierung überhaupt erst möglich:

Die Daten sind nicht private

sondern nur package private

b) Wie ändert sich die übergebene Reihe `g` beim Aufruf mit folgenden Werten:

abstrakter Wert vor dem Aufruf	abstrakter Wert nach dem Aufruf
1 + 0 + 0 + 0 + …	**1 + 0 + 0 + 0 + …**
1 + 3 + 9 + 27 + …	**1 + 1/3 + 1/9 + 1/27 – …**
1 + 1/2 + 1/4 + 1/8 + …	**1 + 1/2 + 1/4 + 1/8 + …**
1 – 1 + 1 – 1 +– …	**1 + 0 + 0 + 0 + …**

Aufgabe 3: (11 Punkte)

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

```
public Intervall tuWas() {
      if (!istKonvergent()) return new Intervall(1/2, q);
      if (q < 0.) return new Intervall(1 + q, 1);
      return new Intervall(1, summe());
}
```

Dabei ist `Intervall` die aus den Übungen bekannte Klasse.

a) Welche Ergebnisse liefert diese Methode bei den nachstehenden Aufrufen? Falls beim Aufruf eine Exception geworfen wird, geben Sie in der Ergebnis-Spalte den Namen der Exception an.

diese `GeometrischeReihe`	Ergebnis-`Intervall`
`1 + 1 + 1 + 1 + …`	**`[0.0,1.0]`**
`1 + 1/2 + 1/4 + 1/8 + …`	**`[1.0,2.0]`**
`1 – 2 + 4 – 8 +– …`	**`IllegalArgumentException`**
`1 – 1/2 + 1/4 – 1/8 +– …`	**`[0.5,1.0]`**

b) Welche **drei** Methoden von `Object` sollten überschrieben werden, wenn `GeometrischeReihe` letztlich eine gute Klasse werden soll. Geben Sie die vollständige Signatur (inklusive Rückgabetyp und Modifizierer) dieser Methoden an:

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