

Prof. Dr. Pascal Laube  
Fakultät Informatik

## Algorithmen und Datenstrukturen

### Übung 2 (AIN2)

1. Implementieren Sie eine Klasse **Bruch**, mit der Brüche (z.B.  $\frac{1}{3}$ ) rundungsfehlerfrei dargestellt werden können, indem Zähler und Nenner jeweils als ganze Zahlen gespeichert werden.  
Überlegen Sie sich welche Member-Variablen und Methoden die Klasse sinnvollerweise haben sollte. Beispielsweise sollte folgende Anwendung möglich sein:

```
Bruch x = new Bruch(1,2);           // x= 1/2
Bruch y = new Bruch(2,3);           // y= 2/3
y.add(x);                           // 2/3 + 1/2 = 4/6 + 3/6
System.out.println("y= " + y.get()); // y= 7/6
```

Es wäre natürlich schön, wenn die Brüche nach jeder Rechnung auch gekürzt würden. Dazu ist die folgende Methode **ggt** hilfreich, die den größten gemeinsamen Teiler zweier positiver Zahlen  $a$  und  $b$  berechnet.

```
int ggt (int a, int b)
// Berechnet den größten gemeinsamen Teiler von a und b
{ if (b>a)
    return (ggt(b,a));
  else
    return ( (b==0) ? a : ggt(b, a%b) );
}
```

2. Testen Sie die Implementierung der Klasse **Bruch** indem Sie für einige Werte  $n$  die folgende Summe berechnen

$$\sum_{i=1}^n \frac{1}{i}$$

3. Implementieren Sie eine Funktion **Bruch e(int n)**, die gemäß folgender Formel zu gegebenem  $n$  eine Näherung für die Eulersche Zahl  $e$  als Bruch berechnet.

$$e(n) = \sum_{i=0}^n \frac{1}{i!} = 1 + \frac{1}{1} + \frac{1}{1 \cdot 2} + \frac{1}{1 \cdot 2 \cdot 3} + \dots + \frac{1}{1 \cdot 2 \cdot \dots \cdot n}$$