

11.11.2022

# SWE3 Übung 03


s2010458016

MARCO SARKADY

WS 22/23

GESAMTAUFWAND: 10 STUNDEN

Medizin – und Bioinformatik



## Inhalt

Lösungsidee .....	3
Testfälle .....	4
Vergleichsoperatoren: .....	4
Is_pos, negative, zero .....	4
copy constructor & test as string .....	4
getter .....	4
Arithmetische Operatoren: .....	5

## Lösungsidee

Anstatt bei jeder Division bzw. Arithmetischer/Vergleichs-Operation zu überprüfen ob der Nenner 0 ist, direkt im Konstruktor sicherstellen, dass erst gar kein Objekt mit  $\text{Nenner} == 0$  erstellt werden kann. Da kein Vergleich oder Rechnung Sinn macht wo  $\text{Nenner} == 0$ , spart uns das eine Menge schreib Arbeit.

Ein Problem dabei ergibt sich bei der Division, da der Nenner 0 sein kann und wir mit dem Kehrwert multiplizieren, kommt es am Ende dann wieder zu einer Division durch 0, wenn  $\text{Nenner} == 0$ . Deshalb wirft man bei der Division zusätzlich den gleichen Error wie im Konstruktor.

Hier reicht es ebenfalls nur den Operator  $/=$  zu betrachten, da wir bei den anderen Operator/ zu diesen delegieren.

Vor jedem Vergleich zweier Objekte der Klasse `rational_t`, werden die Objekte normalisiert.

Beispielsweise wenn  $4/2$  und  $8/4$  verglichen werden, da sonst diese als ungleich anerkannt werden würden. Wenn man aber die Brüche kürzt, so erkennt man, dass es sich um den gleichen Wert handelt.

Es wird übrigens angenommen, dass sich die Werte der erstellten Objekt nicht verändern, deshalb wird nur in den Tests mit Division Null der Try-Catch-Block geschrieben

## Testfälle

### Vergleichsoperatoren:

```
test equal to
2:1
2:1
true

test unequal
2:1
2:1
false

test smaller than
2:1
2:1
false

test smaller than or equal to
2:1
2:1
true

test greater than
2:1
2:1
false

test greater than or equal to
2:1
2:1
true
```

```
test equal to
8:5
2:1
false

test unequal
8:5
2:1
true

test smaller than
8:5
2:1
true

test smaller than or equal to
8:5
2:1
true

test greater than
8:5
2:1
false

test greater than or equal to
8:5
2:1
false
```

### Is\_pos, negative, zero

```
test is_pos, neg, zero
2:1
is positive: true
is negative: false
is zero: false
0:1
is positive: false
is negative: false
is zero: true
-2:1
is positive: false
is negative: true
is zero: false
```

### scan & print

```
test scan function
Enter two numbers:
4
2
4:2
```

```
test scan function
Enter two numbers:
4
0
Divide By Zero Error
```

### copy constructor & test as string

```
test copy constructor
0:1
5:1
a = b
5:1
```

```
test as string
a: 2
b: 1
2 / 1
```

### getter

```
test getter
2:1
numerator: 2
denominator: 1
```

## Arithmetische Operatoren:

```
test addition (compound assignment operator)
2:1
5:2
result: 9:2

test subtraction (compound assignment operator)
2:1
5:2
result: 1:-2

test multiplication (compound assignment operator)
2:1
5:2
result: 5:1

test division (compound assignment operator)
2:1
5:2
result: 4:5

test division by zero (compound assignment operator)
Divide By Zero Error
```

```
test addition
2:1
5:2
result: 9:2

test subtraction
2:1
5:2
result: 1:-2

test multiplication
2:1
5:2
result: 5:1

test division
2:1
5:2
result: 4:5

test division by zero
Divide By Zero Error
```

```
test addition with int
2:1
4
result: 6:1

test subtraction with int
2:1
4
result: -2:1

test multiplication with int
2:1
4
result: 8:1

test division with int
2:1
4
result: 1:2

test division by zero with int
2:1
0
result: Divide By Zero Error
```

```
test addition with int lhs
4
2:1
result: 6:1

test subtraction with int lhs
4
2:1
result: 2:1

test multiplication with int lhs
4
2:1
result: 8:1

test division with int lhs
4
2:1
result: 2:1

test division by zero with int lhs
0
2:1
result: 0:1
```