Bartholomäus Berresheim
Hikari Sophia Stölzle
Viet Anh Pham

PROGRAMMIERAUFGABE

Berechnung von Reihen

PROF. DR. THIEL/ MATHEMATIK 1 GROUP 1 / 24.06.2019

Methoden für die Berechnung folgender Partialsummen

a. $s_n = \sum_{k=1}^{n} \frac{1}{k}$

Methode:

```
public double ReiheA(int n){
    double ergebnis = 0;
    for(double k = 1; k <=n; k++)
    {
        ergebnis= ergebnis + (1/k);
    }
    return ergebnis;
}
```

Test:

Die Folge geht gegen unendlich und ist somit divergent

```
/**
 * Test Methode für Aufgabe a
 */
public void TestA() {
    System.out.println("Aufgabe a\nn-te Iteration ");
    System.out.println("      1: " + ReiheA(1));
    System.out.println("     50: " + ReiheA(50));
    System.out.println("    100: " + ReiheA(100));
    System.out.println("    200: " + ReiheA(200));
    System.out.println("    500: " + ReiheA(500));
    System.out.println("   1000: " + ReiheA(1000));
    System.out.println("  10000: " + ReiheA(10000));
    System.out.println("1000000: " + ReiheA(1000000));
}
```

```
Reihen reihen1 = new Reihen();
reihen1.TestA();
Aufgabe a
n-te Iteration
      1: 1.0
     50: 4.499205338329423
    100: 5.187377517639621
    200: 5.878030948121446
    500: 6.79282342999052
   1000: 7.485470860550343
  10000: 9.787606036044348
1000000: 14.392726722864989
```

Bartholomäus Berresheim
Hikari Sophia Stölzle
Viet Anh Pham

b. $s_n = \sum_{k=1}^{n} (-1)^{k-1} \frac{1}{k}$

Methode:

```
 public double ReiheB(int n)
{
    double ergebnis = 0;
    for (double k = 1; k <= n; k++)
    {
        ergebnis = ergebnis + Math.pow(-1,(k-1)) * (1/k);
    }
    return ergebnis;
}
```

Test:

mit Grenzwert: ln(2) = 0.69314718

Testmethode 1.0:

```
/**
 * Test Methode für Aufgabe b
 */
public void TestB () {
    System.out.println("Aufgabe b\nn-te Iteration  0.69314718");
    System.out.println("      1: " + ReiheB(1));
    System.out.println("     50: " + ReiheB(50));
    System.out.println("    100: " + ReiheB(100));
    System.out.println("    200: " + ReiheB(200));
    System.out.println("    500: " + ReiheB(500));
    System.out.println("   1000: " + ReiheB(1000));
    System.out.println("  10000: " + ReiheB(10000));
    System.out.println("1000000: " + ReiheB(1000000));
}
```

```
 Reihen reihen1 = new Reihen();
reihen1.TestB();
Aufgabe b
n-te Iteration  0.69314718
      1: 1.0
     50: 0.6832471605759183
    100: 0.688172179310195
    200: 0.6906534304818243
    500: 0.6921481805579461
   1000: 0.6926474305598223
  10000: 0.6930971830599583
1000000: 0.6931466805602525
```

Es mussten 1000000 Iterationen durchführen werden, um die vorgegebene Genauigkeit von 0.69314 zu erreichen.

Testmethode 3.0:

```
/**
 * Test Methode für Aufgabe b
 */
public void TestingB(int n,int p)
{
    double d = Math.log(2);
    for (int i = 2; i<=n; i++){

        if((i%2 == 0)  && (d - ReiheB(i) < (1/Math.pow(10,p))))
        {System.out.println ("Erste naheliegende Iteration B:" + i);
         break;
        }
        if((i%2 == 1)  && (d -ReiheB(i) > (1/Math.pow(10,p))))
        {System.out.println ("Erste naheliegende Iteration B:" + i);
         break;
        }
        if(i == n)
        {
            System.out.println("Es konnte kein Ergebnis gefunden werden.\n" +
                               "Bitte versuchen Sie es mit einem " +
                               "größeren n- oder kleineren p- Wert.");
        }
    }
}
```

Wir haben später dann noch ein paar Änderungen an der Testmethode vorgenommen. Mit der neuen Änderung war es uns möglich den Genauigkeitsgrad mit dem Wert p anzugeben. Zusätzlich haben wir ein if statement hinzugefügt für den Falle, dass es kein Ergebnis gibt.
Zudem ist uns aufgefallen, dass bei einem ungeraden i-Wert *ReiheB(i) > d* beziehungsweise bei einem geraden i-Wert *ReiheB(i) < d* vorliegt. Dementsprechend haben wir nochmal an unserem Code die Änderungen vorgenommen.

c. $s_n = \sum_{k=1}^{n} \frac{1}{k^2}$

Methode:

```
 public double ReiheC(int n)
{
    double ergebnis = 0;
    for (double k = 1; k <= n; k++)
    {
        ergebnis = ergebnis + 1/(Math.pow(k,2));
    }
    return ergebnis;
}
```

Bartholomäus Berresheim
Hikari Sophia Stölzle
Viet Anh Pham

Test:

mit Grenzwert: $\frac{\pi^2}{6}$ = 1.644934067

Testmethode 1.0:

```java
/**
 * Test Methode für Aufgabe c
 */
public void TestC(){
    System.out.println("Aufgabe c\nn-te Iteration  1.644934067");
    System.out.println("      1: " + ReiheC(1));
    System.out.println("     50: " + ReiheC(50));
    System.out.println("    100: " + ReiheC(100));
    System.out.println("    200: " + ReiheC(200));
    System.out.println("    500: " + ReiheC(500));
    System.out.println("   1000: " + ReiheC(1000));
    System.out.println("  10000: " + ReiheC(10000));
    System.out.println("1000000: " + ReiheC(1000000));
}
```

```
Reihen reihen1 = new Reihen();
reihen1.TestC();
Aufgabe c
n-te Iteration  1.644934067
      1: 1.0
     50: 1.625132733621529
    100: 1.6349839001848923
    200: 1.6399465460149971
    500: 1.642936065514894
   1000: 1.6439345666815615
  10000: 1.6448340718480652
1000000: 1.64493306684877
```

Es mussten 1000000 Iterationen durchführen werden,
um die vorgegebene Genauigkeit von 1.64493 zu erreichen.

Testmethode 2.0:

```java
/**
 * Test Methode für Aufgabe c
 */
public void TestC(int n){
    boolean out = true;
    double ergebnis = 0;
    double soll = 1.644934067;
    for (double k = 1; k <= n; k++) //iteration
    {
        ergebnis = ergebnis + 1/(Math.pow(k,2));
        if(soll - ergebnis < 0.0001 && out == true){ //0.0001 = grenzwert
            System.out.println("Erste nahliegende Iteration: " + k);
            out = false;
        }
    }
}
```

Als Sollwert nahmen wir $\frac{\pi^2}{6}$ = 1.644934067

Das Ergebnis sollte ein Genauigkeit von 0.0001 haben.

Indem wir einen boolean *out* in die Methode eingebauten, wird bei der ersten gefundenen Iteration *out* auf false gesetzt und die if condition nicht mehr betreten. So wird nur die erste Iteration auf dem Terminal ausgegeben.

Als Parameter n nahmen wir: 100.000

```
Reihen reihen1 = new Reihen();
reihen1.TestC(100000);
Erste nahliegende Iteration: 10000.0
```

Bartholomäus Berresheim
Hikari Sophia Stölzle
Viet Anh Pham

Testmethode 3.0:

```
/**
 * Test Methode für Aufgabe c
 */

public void TestingC(int n,int p)
{
    double d = (Math.pow(Math.PI,2)/6);
    for (int i = 1; i<=n; i++){
        if(d - ReiheC(i) < (1/Math.pow(10,p)))
        {System.out.println ("Erste naheliegende Iteration:" + i);
         break;
        }
        if(i == n)
        {
            System.out.println("Es konnte kein Ergebnis gefunden werden.\n" +
                               "Bitte versuchen Sie es mit einem " +
                               "größeren n- oder kleineren p- Wert.");
        }
    }
}
```

```
reihen1.TestingC(100000, 7);
Es konnte kein Ergebnis gefunden werden.
Bitte versuchen Sie es mit einem größeren n- oder kleineren p- Wert.
reihen1.TestingC(100000, 6);
Es konnte kein Ergebnis gefunden werden.
Bitte versuchen Sie es mit einem größeren n- oder kleineren p- Wert.
reihen1.TestingC(100000, 5);
Erste naheliegende Iteration:100000
```

```
reihen1.TestingC(100000, 4);
Erste naheliegende Iteration:10000
```

Bartholomäus Berresheim
Hikari Sophia Stölzle
Viet Anh Pham

d. $s_n = \sum_{k=0}^{n} \frac{1}{k!}$

Methode:

```
public double ReiheD(int n)
{
    double ergebnis = 0;
    for (double k = 0; k <= n; k++)
    {
        ergebnis = ergebnis + 1/FakultätVon(k);
    }
    return ergebnis;
}
//gehört zu Aufgabe d
private double FakultätVon (double k)
{
    return k <= 1? 1 : k*FakultätVon(k-1);
}
```

Test:

mit Grenzwert: **e** = 2.718281828

Testmethode 1.0:

```
/**
 * Test Methode für Aufgabe d
 */
public void TestD(){
    System.out.println("Aufgabe 4\nn-te Iteration 2.718281828");
    System.out.println("     1: " + ReiheD(1));
    System.out.println("    10: " + ReiheD(10));
    System.out.println("    25: " + ReiheD(25));
    System.out.println("    50: " + ReiheD(50));
}
```

```
Reihen reihen1 = new Reihen();
reihen1.TestD();
Aufgabe 4
n-te Iteration 2.718281828
     1: 2.0
    10: 2.7182818011463845
    25: 2.7182818284590455
    50: 2.7182818284590455
```

Es mussten 25 Iterationen durchführen werden, um die vorgegebene Genauigkeit von 2.718281828 zu erreichen.

Testmethode 2.0:

```
public void TesteD(int n)
{
    boolean out = true;
    double ergebnis = 0;
    double soll = 2.718281828;
    for (double k = 0; k <= n; k++)
    {
        ergebnis = ergebnis + 1/FakultätVon(k);
        if(soll - ergebnis < 0.0000000001 && out == true){
            System.out.println("Erste nahliegende Iteration: " + k);
            out = false;
        }
    }
}
```

Testmethode 2.1:

```
public void TestingD(int n)
{
    double soll = 2.718281828;
    for(int k =1; k <= n; k++){
        if(soll - ReiheD(k) < 0.0000000001){
            System.out.println("Erste nahliegende Iteration: " + k);
            break;
        }
    }

}
```

```
Reihen reihen1 = new Reihen();
reihen1.TestingD(10000);
Erste nahliegende Iteration: 12
```

Testmethode 3.0:

```
public void TestingD(int n,int p)
{
    double d = Math.E;
    for (int i = 1; i<=n; i++){
        if(d - ReiheD(i) < (1/Math.pow(10,p)))
        {System.out.println ("Erste naheliegende Iteration:" + i);
         break;
        }
        if(i == n)
        {
            System.out.println("Es konnte kein Ergebnis gefunden werden.\n" +
                               "Bitte versuchen Sie es mit einem " +
                               "größeren n- oder kleineren p- Wert.");
        }
    }
}
```

```
reihen1.TestingD(10000, 9);
Erste naheliegende Iteration:12
```

Bartholomäus Berresheim
Hikari Sophia Stölzle
Viet Anh Pham

Code

Reihen

```
/**
 * Aufgabe 4 Berechnung von Reihen
 *
 * @author Viet Bartholomäus Hikari
 * @version 22.06.2019
 */
public class Reihen
{
    /**
     * Constructor for objects of class Reihen
     */
    public Reihen()
    {

    }

    /**
     * a.)
     */
    public double ReiheA(int n){
        double ergebnis = 0;
        for(double k = 1; k <=n; k++)
        {
            ergebnis= ergebnis + (1/k);
        }
        return ergebnis;
    }

    /**
     * b.)
     */
     public double ReiheB(int n)
    {
        double ergebnis = 0;
        for (double k = 1; k <= n; k++)
        {
            ergebnis = ergebnis + Math.pow(-1,(k-1)) * (1/k);
        }
        return ergebnis;
    }

    /**
```

```
 * c.)
 */
 public double ReiheC(int n)
{
    double ergebnis = 0;
    for (double k = 1; k <= n; k++)
    {
        ergebnis = ergebnis + 1/(Math.pow(k,2));
    }

    return ergebnis;
}


/**
 * d.)
 */
public double ReiheD(int n)
{
    double ergebnis = 0;
    for (double k = 0; k <= n; k++)
    {
        ergebnis = ergebnis + 1/FakultätVon(k);
    }
    return ergebnis;
}
//gehört zu Aufgabe d
private double FakultätVon (double k)
{
    return k <= 1? 1 : k*FakultätVon(k-1);
}

/**
 * Test Methode für Aufgabe a
 */
public void TestA() {
    System.out.println("Aufgabe a\nn-te Iteration  ");
    System.out.println("      1: " + ReiheA(1));
    System.out.println("     50: " + ReiheA(50));
    System.out.println("    100: " + ReiheA(100));
    System.out.println("    200: " + ReiheA(200));
    System.out.println("    500: " + ReiheA(500));
    System.out.println("   1000: " + ReiheA(1000));
    System.out.println("  10000: " + ReiheA(10000));
    System.out.println("1000000: " + ReiheA(1000000));
}
```

```
   /**
    * Test Methode für Aufgabe b
    */
  public void TestingB(int n,int p)
   {
       double d = Math.log(2);
       for (int i = 2; i<=n; i++){

           if((i%2 == 0)  && (d - ReiheB(i) < (1/Math.pow(10,p))))
           {System.out.println ("Erste naheliegende Iteration B:" + i);
            break;
           }
           if((i%2 == 1)  && (d -ReiheB(i) > (1/Math.pow(10,p))))
           {System.out.println ("Erste naheliegende Iteration B:" + i);
            break;
           }
           if(i == n)
           {
               System.out.println("Es konnte kein Ergebnis gefunden werden, bitte versuchen
sie es mit einem größeren n- oder kleineren p- Wert nochmal.");
           }
       }
   }


   /**
    * Test Methode für Aufgabe c
    */

  public void TestingC(int n,int p)
   {
       double d = (Math.pow(Math.PI,2)/6);
       for (int i = 1; i<=n; i++){
           if(d - ReiheC(i) < (1/Math.pow(10,p)))
           {System.out.println ("Erste naheliegende Iteration:" + i);
            break;
           }
           if(i == n)
           {
               System.out.println("Es konnte kein Ergebnis gefunden werden, bitte versuchen
sie es mit einem größeren n- oder kleineren p- Wert nochmal.");
           }
       }
   }

   /**
    * Test Methode für Aufgabe d
    */
   public void TestingD(int n,int p)
```

```
   {
      double d = Math.E;
      for (int i = 1; i<=n; i++){
         if(d - ReiheD(i) < (1/Math.pow(10,p)))
         {System.out.println ("Erste naheliegende Iteration:" + i);
          break;
         }
         if(i == n)
         {
            System.out.println("Es konnte kein Ergebnis gefunden werden, bitte versuchen
sie es mit einem größeren n- oder kleineren p- Wert nochmal.");
         }
      }
   }

}
```