Aufgabenblatt: Methoden - rekursiv

(1.) Wird noch erstellt!

Lösen Sie die folgenden Aufgaben indem Sie:
- Basis und Rekursive Bedingung formulieren!
- die vorgegebene Methode programmieren
- ein Rahmenprogramm erstellen, um die Funktion zu testen, falls dies nicht in der Aufgabe formuliert ist!
- das fertige Programm mit F7 mit kleinen(!) Aufrufewerte der rekursiven Funktion durchlaufen lassen!
VERBINDLICH: Einrückungen und sinnvolle Namensgebung(mindestens 3 Buchstaben!) der Variablen!

(2.) (a.) Kopieren Sie das Beispiel aus dem Unterricht und lassen Sie es mit F7 durchlaufen! Wie groß ist die maximale Anzahl der Stackgröße(unten rechts!)?
(b.) 7 Personen können mit 7! Möglichkeiten auf 7 Plätze verteilt werden! Bestimmen Sie, wie viele verschiedene Möglichkeiten es gibt, sich in ihrer Klasse auf Stühle zu setzen!
(3.) (a.) Schreiben Sie eine rekursive Methode double fibo(int n) . fibo soll die n-te Fibonaccizahl berechnen. Da die Fibonaccizahlen sehr schnell sehr groß werden, soll für Argumente größer 92 der Wert -1 zurückgegeben werden. Der rekursive Algorithmus lautet

Basisbedingung: fibo(0) = 0 , fibo(1) = 1
Rekursive Bedingung: fibo(n) = fibo(n-1) + fibo(n-2)

(b.) Bestimmen Sie die Quotienten der aufeinanderfolgenden Fibonaccizahlen für n=2,.....30 und damit den genauen Wert für den Goldenen Schnitt!
(4.) Schreiben Sie eine rekursive Methode summeN zur Bildung der Summe der Zahlen 1 bis n! Bestimmen Sie damit die Summe der Zahlen 1 bis 100!
(5.) Schreiben Sie eine rekursive Methode (8.) zur Berechnung der Potenz a^n für beliebige Potenzen mit positiven Exponenten! Bestimmen Sie damit die Potenz 2 hoch 10! Warum ist dies eine besondere Zahl?
(6.)Schreiben Sie eine rekursive Methode double nÜberK(int n, int v) . nÜberK ermittelt den Binomialkoeffizienten zu n und v, also n über v. Dazu verwenden Sie die folgende Rekursionsformel:

$$\binom{n+1}{v} = \binom{n}{v} + \binom{n}{v-1} \qquad \binom{n}{0} = 1 \qquad \binom{n}{n} = 1 \qquad n \geq 1 \quad v \geq 0 \quad n \geq v$$

Basisbedingung: v= 0: 1 ; v=n: 1
Rekursive Bedingung: binko(n,v)= binko(n-1,v)+ binko(n-1,v-1)
Berechnen Sie damit 49 über 6 und die Anzahl der Möglichkeiten der Ausspielung 6 aus 49 im Wochenendlotto!

(7.) Schreiben Sie eine rekursive Methode int ggt(int a, int b) . ggT ermittelt den größten gemeinsamen Teiler der Zahlen a und b. Für eine rekursive Methode ziehen Sie die folgende von Euklid (325 - 265 v.Chr.) entdeckte Eigenschaft heran:
Wenn a und b durch t teilbar sind, so auch a%b!
Basisbedingung: a%b = = 0 : b
Rekursive Bedingung: a%b ! =0: ggT(b, a%b)

(8.) (a.) Schreiben Sie eine rekursive Methode static boolean isPalindrome(String PalindromeStr), welche überprüft, ob eine Zeichenkette ein Palindrom ist z. B. anna!
Basisbedingung: Zeichenkette ist nur ein Zeichen oder leer
Rekursive Bedingung: erster Buchstabe =letzterBuchstabe && isPalindrome(2ter bis vorletzter Buchstabe)
(b.) Erweitern Sie obiges Programm, dass auch Palindrome mit Groß-/Kleinschreibung erkannt werden z.B. Anna!

(9.) Gegeben ist ein Haufen mit n>1 Chips. Zerlege den Haufen zufällig in zwei Haufen mit k und n-k Chips und berechne das Produkt. Dann wird jeder der beiden Haufen mit mehr als einem Chip ausgewählt und zufällig in zwei Haufen zerlegt. Das Produkt seiner Teile wird zum vorangehenden Produkt addiert. Das Verfahren wird so lange wiederholt, bis alle Haufen auf 1 reduziert sind.
(a.) Schreiben Sie eine rekursive Methode, die den Wert der Zerlegung bei gegebener Chipanzahl berechnet!
(b.) Überprüfen Sie mit einer Wiederholungsanweisung, ob bei 1000000-facher Berechnung die Ergebnisse für die Zahlen 5 und 8 immer identisch sind!
(c.) Wenn f(n) der Wert der Zerlegung ist, bestimmen Sie eine Wertetabelle von 1 bis 20!
(d.) Erkennen Sie ein Muster und bestimmen Sie den Term für f(n), z.B. $f(n)= n^2+n$. Dies kann nicht mit Hilfe von Java gelöst werden, ist also eine mathematische Aufgabe.

*Hinweis zur Erstellung von **ganzen** Zufallszahlen:*
```
import java.util.Random;
//Allererste Zeile vor public class
Random randomGenerator = new Random();
// Instanzen randomGenerator der Klasse Random wird erzeugt…zu Beginn des Programms oder Methode!
int randomInt = randomGenerator.nextInt(8);
// ganze Zahl zwischen 0 und 7 wird erzeugt(NICHT 8!)
```

*(10.) Schreibe eine rekursive Methode **void reverse(int n)**, die für einen positiven Integerwert x die Ziffern in umgekehrter Reihenfolge ausgibt. Aufruf: reverse(1234); Ausgabe: 4321.*

(11.) Jeep Fahrt
http://www.saar.de/~awa/jrekursion.html

Aufgabenblatt: Methoden - rekursiv – Lösungen

(3.)

```
public class FibonacciRekursiv
{
  public static void main(String args[])
  {
    for(int i=0 ; i<10 ; i++)
       System.out.println("fibo("+i+") = " + fibo(i) ) ;
  }

  public static double fibo(int n)
  {
    if (n<=1) return n;
    if (n>92) return -1;
    return fibo(n-1) + fibo(n-2) ;
  }
} // end class
```

(4.)

```
 public static double summeN(int n)
 {
  if (n==0) return 0 ;
  return n+ summeN(n-1);
 }
```

(5.)

```
 public static double PotenzN (double a,int n)
 {
  if (n==0) return 1 ;
  return a* PotenzN (a,n-1);
 }
```

(6.)

```
public class Binomial
{
  public static void main(String args[])
  {
    for(int n=0 ; n<10 ; n++)
    {
      System.out.print(n + " : ") ;
      for(int v=0; v<=n; v++)
         System.out.print(binkoff(n,v) + "\t" ) ;
      System.out.println() ;
    }
    System.out.println(binkoff(4,5));
  }

  public static double binkoff(int n, int v)
  {
    if (v==0) return 1 ;
    if (v==n) return 1 ;
    int tmp1= binkoff(n-1,v);
    int tmp2= binkoff(n-1,v-1);
    return tmp1+ tmp2;
  }
} // end class
```

(7.)

```
public class GGT
{
  public static void main(String args[])
  {
    System.out.println( ggt(6,8));
  }

  public static int ggT(int a, int b)
  {
    //return (a%b==0) ? b : ggT(b, a%b);
   if (a%b==0)
          return b
   else
        return ggT(b, a%b)
  }
} // end class
```

(8.)

```
 public static boolean isPalindrome (String PalindromeStr)
 {
   if (PalindromeStr.length()<2)
      return true ;
   String firstChar=PalindromeStr.substring(0,1);
   String lastChar=PalindromeStr.substring(PalindromeStr.length()-1,PalindromeStr.length());
   String middleString=PalindromeStr.substring(1,PalindromeStr.length()-1);
   return
          firstChar.equals(lastChar) && isPalindrome(middleString);
 }
```

(9.) (a.)

```
import java.util.Random;
public class ChipsZerlegung
{
  public static void main(String args[])
  {
    System.out.println("Result for n=5:" + ChipChipHurra(5));
  }

  public static int ChipChipHurra(int n)
  {
    Random randomGenerator = new Random();
    if (n==1)
      return 0;
    int k = 1+randomGenerator.nextInt(n-1);
    int tmp1=ChipChipHurra(k);
    int tmp2=ChipChipHurra(n-k);
    return k*(n-k)+tmp1+tmp2;
  }
} // end class
```

Optimierte Methode:

```
public static int ChipChipHurra(int n)
{
   Random randomGenerator = new Random();
   if (n==1)
      return 0;
   int k = 1+randomGenerator.nextInt(n-1);
   return k*(n-k)+ChipChipHurra(k)+ChipChipHurra(n-k);
}
```

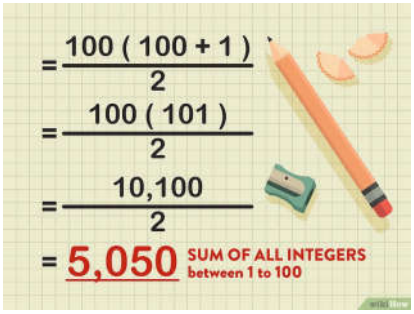
(b.)

```
import java.util.Random;
public class ChipsZerlegung
{
  public static void main(String args[])
  {
    int oldValue=ChipChipHurra(5);
    int newValue;
    for (int i=1; i<1000;i++){
        newValue=ChipChipHurra(5);
        if (newValue!=oldValue){
          System.out.println("Werte sind unterschiedlich:!"));
          break;
        }
    }
    if (newValue==oldValue){
      System.out.println("Werte sind immer gleich!");
  }
  public static int ChipChipHurra(int n)
  {
    Random randomGenerator = new Random();
    if (n==1)
      return 0;
    int k = 1+randomGenerator.nextInt(n-1);
    int tmp1=ChipChipHurra(k);
    int tmp2=ChipChipHurra(n-k);
    return k*(n-k)+tmp1+tmp2;
  }
}  // end class
```

(c.)

```
import java.util.Random;
public class ChipsZerlegung
{
  public static void main(String args[])
  {
    for (int i=1; i<=20;i++)
        System.out.println("n="+i+" "+ChipChipHurra(i));
  }
  public static int ChipChipHurra(int n)
  {
    Random randomGenerator = new Random();
    if (n==1)
      return 0;
    int k = 1+randomGenerator.nextInt(n-1);
    int tmp1=ChipChipHurra(k);
    int tmp2=ChipChipHurra(n-k);
    return k*(n-k)+tmp1+tmp2;
  }
}  // end class
```

(d.) Es wird die Summe der Zahlen bis zur Zahl vorher gebildet!
Bei n Zahlen werden also die Zahlen 1 bis n-1 addiert!
Wenn Sie „Summe 1 bis 100“ googlen finden Sie z.B.



Die Formel für die Summe der Zahlen 1 bis n lautet: n(n+1)/2
Wir summieren aber NICHT von 1 bis n, sondern nur bis n-1. Also ist n in der Formel durch n-1 zu ersetzen:
*(n-1)((n-1)+1)/2 = (n-1)*n/2*
Dies wäre das Ergebnis!

$$(n-1)n/2 = \frac{n*(n-1)}{2} = \frac{n*(n-1)*(n-2).....*3*2*1}{2!*(n-2).....*3*2*1} = \frac{n!}{2!(n-2)!} = \binom{n}{2}$$

(10.)

```
 static void reverse( int number){
   if (number>0) {
       int digit=number%10;
       System.out.print(digit);
       reverse(number/10);
   } // end of if
 }
```