S. 24/3 Bakterienwachstum

Rekursive Definition:

$$N(t) = \begin{cases} 1,2 \cdot N(t-1) & \text{falls } t>0 \\ 300 & \text{falls } t=0 \end{cases}$$

```
public class Bakterien
{
    private int anzahl0=300;
    private double wachstumsfaktor = 1.2;

    public double anzahlBerechnen(int n){
        if (n == 0) return anzahl0;
        else return anzahlBerechnen(n-1) * wachstumsfaktor;
    }
}
```

public int length():
Gibt die Länge des Strings zurück.

public char charAt(int index):
Gibt das Zeichen an der spezifizierten Stelle des Strings zurück. Index geht von 0 bis length()-1.

public String substring(int beginIndex, int endIndex):
Gibt einen Teil des Strings als neuen String zurück. Beginnt bei beginIndex, endet bei endIndex – 1. Damit ist die Länge endIndex-beginIndex.

Beispiel: "Wolnzach".substring(1,5); ergibt "olnz"

S. 24/4 Zeichenketten II

a) Rekursive Definition:

$$paltest(wort) = \begin{cases} paltest(teilwort) \,\&\& \\ (erstesZeichen == letztesZeichen) \end{cases}$$ falls wortlänge > 1

$$\text{true}$$ falls wortlänge ==0 oder wortlänge ==1;

teilwort = wort ohne erstes und letztes Zeichen

```
public boolean palindromtest(String s){
    if ((s.length() == 0) || (s.length() == 1)) { return true;}
    else {
        return palindromtest(s.substring(1, s.length()-1))
                && (s.charAt(0) == s.charAt(s.length()-1));
     }
  }
```

S. 24/4 Zeichenketten II

b) Rekursive Definition:

$$umdrehen(wort) = \begin{cases} letztesZeichen + umdrehen(restwort) & \text{falls wortlänge} > 1 \\ wort & \text{falls wortlänge} == 1; \end{cases}$$

restwort = wort ohne letztes Zeichen

```
public String umdrehen(String s) {
        if (s.length()==1) {return s;}
        else {
                return s.charAt(s.length()-1)+
                                umdrehen(s.substring(0, s.length()-1));
        }
}
```

c)

```
public boolean palindromtest2(String s){
        return s.equals(umdrehen(s));
}
```

S. 25/5 Dreiecksmuster

```
public class Dreiecksmuster
{
    public int anzahlGeben(int n) {
        if (n==1) {return 1;}
        else {return anzahlGeben(n-1)+4; }
    }

}
```

S. 25/7 Das Märchen vom Reiskorn

a) Abbruchbedingung fehlt

```
public long potenzVonZwei(int n){
        if (n==1) {return 2;}
        else { return 2*potenzVonZwei(n-1);  }
    }
```

b) potenzVonZwei(63)

Für Datentyp long gerade um 1 zu groß.

c)
```
public long anzahlKoerner(int n) {
    if (n==0) {return 1;}
    else{return anzahlKoerner(n-1)+potenzVonZwei(n); }
 }
```