ÜBERLADENE FUNTIONEN

```
int divide(int x, int y, int &Rest)
{
    Rest = x%y;
    return x/y;
}
double divide(int x, int y)
{
    return (double)x/(double)y;
}
```

Member Constructor

```
Person::Person (string nameStr, int year,
                        int month, int day)
    : bornDate (year, month, day),
      name (nameStr) {}
```

STRTOK BSP

```
#include <stdio.h>
#include <string.h>

int main ()
{
  char str[] ="- This, a sample string.";
  char * pch;
  printf ("Splitting string \"%s\" into tokens:\n",str);
  pch = strtok (str," ,.-");
  while (pch != NULL)
  {
    printf ("%s\n",pch);
    pch = strtok (NULL, " ,.-");
  }
  return 0;
}
```

VERERBUNG FIGUR

```
class Figur  {
public:
    virtual double area() { return 0; }
    virtual double scope() { return 0; }
    virtual void show() { std::cout << "Figur"
<< std::endl;}
    friend std::ostream& operator<<
(std::ostream&, Figur& f) { f.show();
f.draw();}
    virtual void draw() = 0;
};

class Square : public Figur {
private:
    double s;
public:
    Square(double side) { this->s = side; }
    double area() { return s*s; }
    double scope() { return 4*s; }
    void show() { std::cout << "s: " << s << "
area: " << area() << " scope: " << scope() <<
std::endl; }
    void draw() {...}
};
```

GETLINE BSP

```
#include <iostream>
#include <string>
int main ()
{
  std::string name;
  std::cout << "Please, enter your full name: ";
  std::getline (std::cin, name);
  std::cout << "Hello, " << name << "!\n";
  return 0;
}
```

ASCII TABLE IO MANIP

```
#include <iomanip>
#include <iostream>
using namespace std;
int main() {
    int i,j;
    cout << "|dec hex Char |dec hex Char
            <<  |dec hex Char |dec hex Char |"
    << endl;
    for(i=0;i<=31;i++) {
        cout << '|';
        for(j=0;j<=3*32;j+=32) {
            cout << right << dec << setw(4)
             << i+j << hex << setw(4) << i+j;
            if(isgraph(i+j)) {
                cout << setw(4) << (char) (j+i);
            } else {
                cout << setw(5) << '.';
            }
            cout << '|';
        }
        cout << endl;
    }
    return 0;
}
```

TEMPLATE

```
template <class T, int bufSizeMax>
class Buffer {
private:
    T vBuf[bufSizeMax];
    int iterator;
    short int sizeInput;
    int sizeMax;
    void iteratorNext () {
        iterator = (iterator+1)% bufSizeMax;
    }
public:
    Buffer() {
        iterator = 0;
        sizeMax = bufSizeMax;
    }
    void show() {
        for(int i = 0; i < sizeMax && i < sizeInput-1; i++) {
            std::cout << i << "]: " << vBuf[i] << std::endl;
        }
    }
    void store(T newElement) {
        vBuf[iterator] = newElement;
        iteratorNext();
        if(sizeInput < sizeMax)sizeInput++;
    }
    // >0 found, -1 no such element
    int find(T searchElement) {
        for(int i = 0; i < sizeMax; i++) {
            if(searchElement == vBuf[i])
                return i;
        }
        return -1;
    }

    T& operator[] (int elemNum) {
        T * Dummy = new T();
        if(elemNum > sizeMax) return *Dummy;
        return vBuf[elemNum];
    }
};
```

STRING CLASS

```
class STRING {
    char * pBuf;
    int         Len;
public:
    STRING ();
    STRING (const char* pStr);
    STRING (const STRING& other);
    STRING (char C, int n);
    ~STRING();
    void show () const;
    char& CIdx(unsigned int i);
    int getLength();
    //Operatorüberladugen
    STRING& operator= (const STRING& other);

    STRING operator+ (const STRING& other);
    STRING operator+ (const char *other);

    friend std::ostream& operator<< (std::ostream& os, const
STRING& other);

    char& operator[] (unsigned int index);
};

STRING STRING::operator+ (const char *pOtherBuf) {
    Len += strlen(pOtherBuf);
    char *newPBuf = new char[this->Len];
    strcpy(newPBuf, this->pBuf);
    strcat(newPBuf, pOtherBuf);

    return STRING(newPBuf);
}

ostream& operator<< (ostream& os, const STRING& other)
{
    char *pTemp = other.pBuf;
    while(*pTemp!='\0') {
        os << *pTemp;
        pTemp++;
    }


    return os;
}

char& STRING::operator[] (unsigned int i) {
    if( i >= Len) {
        cout << "Indexzugriffsfehler" << endl;
        static char DUMMY = '0';
        return DUMMY;
    }
    return pBuf[i];
}
```

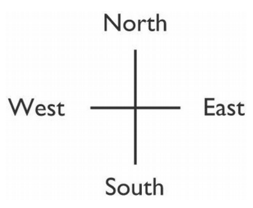
```
KeyListener enterListener = new KeyListener() {
  @Override
  public void keyPressed(KeyEvent e)
  {
    if(e.getKeyCode() == 10)
    {
      //..do
    }
  }
  public void keyReleased(KeyEvent e){}
  public void keyTyped(KeyEvent e){}
};

tf.addKeyListener(enterListener);
```

CALCULATOR BORDER + GRID LAYOUT

```
import java.awt.*;
import java.awt.event.*;

class Calculator extends Panel
{
  // hier Referenzen fuer Komponenten
  // (Buttons, Textfields, Panels) vereinbaren
  TextField tf;
  String buttons[] = { "M+", "7", "8", "9", "/", ... };

  Button tmpB;

  public Calculator()
  {
    // Komponenten erzeugen und zu Oberflaeche zusammenbauen,
    // Listener verbinden
    setFont(new Font("System", Font.PLAIN, 24));
    setLayout(new BorderLayout());
    tf = new TextField();
    Panel keys = new Panel(new GridLayout(4,5));
    add(tf, BorderLayout.NORTH);
    for(int i = 0; i < buttons.length; i++) {
            tmpB = new Button(buttons[i]);
            keys.add(tmpB);
    }
    add(keys, BorderLayout.CENTER);
  }

  public static void main(String args[])
  {
      Frame F=new Frame();
      F.addWindowListener(new WindowAdapter() {
        public void windowClosing(WindowEvent we)
            { System.exit(0); }
      });
      Calculator p = new Calculator();
      F.add(p);
      F.pack();
      F.setVisible(true);
  }
==============================================
  ActionListener nOp = new ActionListener()
   {
      @Override
       public void actionPerformed(ActionEvent e) {
            ...
            String newOperation = e.getActionCommand();
            ...
      }
  };
  ActionListener actionEvents [][] = {{ nL,  nL, nL, nL, nOp}, ...};
  GridBagConstraints C=new  GridBagConstraints();
  public CalculatorGB()
  {
    // Komponenten erzeugen und zu Oberflaeche zusammenbauen,
    // Listener verbinden
    setFont(new Font("System", Font.PLAIN, 24));
    setLayout(new GridBagLayout());
    C.fill=GridBagConstraints.BOTH;

    tf = new TextField(20);
    C.gridx=0; C.gridy=0;
    C.gridwidth=GridBagConstraints.REMAINDER;
    C.weightx=1.0;C.weighty=1.0;
    add(tf,C);
    C.gridwidth=1;

    C.weightx=1.0;C.weighty=1.0;
    for(int i = 0; i < buttons.length; i++) {
      for(int j = 0; j < buttons[i].length; j++) {
        C.gridx=j; C.gridy=i+1;
        tmpB = new Button(buttons[i][j]);
        tmpB.addActionListener(actionEvents[i][j]);
        add(tmpB,C);
      }
    }
  }
===============================================
}
```

READ FROM FILE

```
import java.io.*;

class Stream {
    public static void main (String [] args) {
        try {
// FileInputStream instanzieren
    FileInputStream myStream;
    myStream = new FileInputStream("Taxigeschichten.txt");
//und via available die Anzahl bereitstehender Bytes ermitteln
    int byteAvailable = myStream.available();
    byte [] buff = new byte[byteAvailable];
// von myStream available ablesen
    myStream.read(buff, 0, byteAvailable);
    System.out.println(new String(buff) + " //bytes: " + byteAvailable);
 ============================================================
//erzeugen Sie ein Objekt des Typs File
    File myFile = new File("Taxigeschichten.txt");
// ermitteln Sie die exakte Länge der Datei
    int size = (int)myFile.length(); // get available bit count
// FileInputStream instanzieren
    FileInputStream in = new FileInputStream(myFile);
    byte [] buff = new byte[size];
// Mit Schleife ablesen
    int bytesRead = 0;
    while (bytesRead < size)
        bytesRead+=in.read(buff, bytesRead, size-bytesRead);
    System.out.println(new String(buff));
 ===========================================================
    File myFile = new File("Taxigeschichten.txt");
    FileInputStream in = new FileInputStream(myFile);
    ByteArrayOutputStream bytesOS = new ByteArrayOutputStream();
    byte [] buff = new byte[1024];
    int bytesRead = 0;
    while ((bytesRead=in.read(buff)) > -1) { // schreiben in Zwischenspeicher buff
        bytesOS.write(buff, 0, bytesRead); // schreiben von buf in bytesOutputStream
    }
    System.out.println(bytesOS.toString());
 ============================================================
    File myFile = new File("Taxigeschichten.txt");
    FileReader in = new FileReader(myFile);
    CharArrayWriter charOS = new CharArrayWriter();
    char [] buff = new char[1024];
    int bytesRead = 0;
// schreiben in Zwischenspeicher buff
    while ((bytesRead=in.read(buff, 0, 1024)) > -1) {
        charOS.write(buff, 0, bytesRead); // schreiben von buf in charOS
    }
    System.out.println(charOS.toString());

    } catch (Exception e) {
        System.out.println(e);
        return;
    }
  }
}
```



TIPPS TO FIELDS

```
ta.setEditable(false);

TextField tf = new TextField(40);
```

URL BEISPIEL

```
try
{
  URL url = new URL(tf.getText());
  InputStream i= url.openStream();
  HexDump h = new HexDump(i);
  ta.setText(h.getHexString());
}
catch (MalformedURLException ex)
  {System.out.println(ex);System.exit(1);}
catch(IOException ex)
  {System.out.println(ex);System.exit(1);}

public class HexDump
{

  byte data[];

  HexDump(InputStream fis)
  {
     try
     {
       ByteArrayOutputStream bos=new
            ByteArrayOutputStream(1024);
       byte buf[]=new byte[1024];
       int lenr;
       while ((lenr=fis.read(buf))>-1)
         bos.write(buf,0,lenr);
       data=bos.toByteArray();
     }catch(Exception e)
{System.out.println(e);}
  }
```

URL FORT

```
private char[] hexByte(int z, int len)
  {
    char[] x=new char[len];
    int i,hx;
    for(i=len-1;i>=0;i--)
    {
      hx=z;
      z>>>=4;
      hx&=0xf;
      x[i]=(char)(hx<=9?hx+'0':hx+'A'-10);
    }
    return x;
  }

 public String getHexString()
  {

   String s ="";

   for(int i = 0; i<data.length; i+=16) {

     s+="\n" + new String(hexByte(i,4)) + ": ";
     for(int j = 0; j<16 && (i+j)<data.length; j++)
     {
       s+=new String(hexByte(data[i+j],2)) + ((((j+1)%4
== 0) && (j<15))? " | ":" ");
     }
     for(int j = 0; j<16 && (i+j)<data.length; j++)
     {
       s+= (data[i+j] > ' ') ? (char)(data[i+j]): ".";
     }
   }

    return s;
  }
}
```