M411_Test 1_MirioEggmann_20160511

Drei Ebenen des Algorithmen Entwurfs:

- 1. Spezifikation: Präziese Fragen vor der Programmierung.

Wir:

Pre-conditions (Relevante Eigenschaften vor der Ausführen des Algo) z.B. ggt(a,b):

1. Welche Zahlen a,b sind zugelassen? -> Positive, negative Zahlen?

2. Welche Grundoperationen sind erlaubt? -> +, - oder auch mod?

Post-conditions (Relevante Eigenschaften, nach der Ausführung) z.B. ggt(a,b):

1. Was wird ausgegeben, falls m,n < 0?

2. Was wird ausgegeben, falls die Eingabe nicht den precon genügt?

Praxis:

Häufig weniger formale beschr. in Pflichtenheften, die umfangreich, mehrdeutig, inkonsistent sind.

- 2. Algorithmus: Genauer Ablauf muss bekannt sein. (z.B. strukto..)

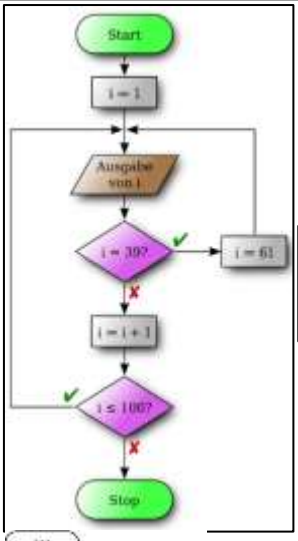
1. Pseudocode / 2. Flussdiagramm / 3. Struktogramm / 4. UML-Diagramm / 5. Programmcode

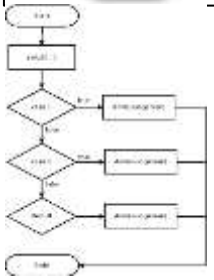
(Pseudocode: wenn, ansonsten, solange, setze, zeige, ersetze,..)

- 3. Programm: „Glue Code" – Konkrete Programmierung z.B. mit Java

```java
import java.io.*;
import java.util.Scanner;
```

```java
public class AB411_03_XOR {
    public static String encrypt(String text, int key) {
        char[] zeichen = text.toCharArray();
        for (int i=0; i<zeichen.length;i++) {
            zeichen[i] = (char)(zeichen[i]^key);
        }
        return new String(zeichen);
    }
    public static void main(String[] args) throws IOException {
        // Achtung KEINE Validierung
        if (args.length >= 3) {
            // int key = 23;
            int key = Integer.parseInt(args[0]);
            //String plain="/home/vmadmin/Dokumente/M411/NetBeansProjects/AB411_03_XOR/Gedicht.txt";
            String plain = args[1];
            //String crypt="/home/vmadmin/Dokumente/M411/NetBeansProjects/AB411_03_XOR/crypt.txt";
            String crypt = args[2];
            try (Scanner scanner = new Scanner(new File(plain), "UTF-8")) {
                BufferedWriter myWriter= new BufferedWriter(new FileWriter(crypt,false));
                int counter = 0;
                while (scanner.hasNextLine()) {
                    String line = scanner.nextLine();
                    myWriter.write(encrypt(line,key)+"\n");
                    counter++;
                }
                myWriter.close();
                scanner.close();
            }catch (FileNotFoundException eIO) {
                System.out.println("Folgender Fehler trat auf: "+eIO);
            }
        } else {
            System.out.println("Bitte richtige Anzahl Parameter mitgeben!");
        }
    }
```





```java
public class AB411_05_Matrix {
    public static void main(String[] args) {
        int[][] matrix = {{4, 5, 6},
                          {2,-9,-3}};
        for (int i=0; i<matrix.length; i++) {
            for (int j=0; j<matrix[i].length; j++) {
                System.out.print(matrix[i][j]+"\t");}
            System.out.println();}  }}
```

```java
public class AB411_03_ParameterEingabe {
    public static void main(String[] args) {
        for (int i=0; i < args.length; i++) {
            System.out.println("Eingabe "+i+": >"+args[i]+"<");
        }
```

Zeichen	Bedeutung
\b	Rückschritt (Backspace)
\t	Horizontaler Tabulator
\n	Zeilenschaltung (Newline)
\r	Seitenumbruch (Formfeed)
\f	Wagenrücklauf (Carriage return)
\"	Doppeltes Anführungszeichen
\'	Einfaches Anführungszeichen
\\	Backslash
\nnn	Oktalzahl nnn (kann auch kürzer

```java
public class AB411_05_GameOfLife {
// global definierte Konstanten für die beiden Dimensionen
final static int DIM1 = 12;
final static int DIM2 = 12;
// liefert eine zufällig initialisierte Welt
public static boolean[][] initWelt() {
    boolean[][] welt = new boolean[DIM1][DIM2];
    for (int y=1; y<DIM2-1; y++)
        for (int x=1; x<DIM1-1; x++)
            welt[x][y] = Math.random() > 0.4; // 60% Lebendig
    return welt;}
// liefert Anzahl Nachbarn einer Zelle
public static int anzNachbarn(boolean[][] welt, int x, int y) {
    int ret = 0;
    for (int i=x-1; i<=x+1; ++i)
        for (int j=y-1; j<=y+1; ++j)
            if (welt[i][j])
                ret += 1;
    // einen Nachbarn zuviel mitgezählt?
    if (welt[x][y])
        ret -= 1;
    return ret;}
// gibt die aktuelle Welt aus
public static void zeigeWelt(boolean[][] welt) {
    for (int y=1; y<DIM2-1; y++) {
        for (int x=1; x<DIM1-1; x++) {
            if (welt[x][y])
                System.out.print("X");
            else
                System.out.print(" ");    }
        System.out.println();  }
    System.out.println();}
// wendet die 4 Regeln an und gibt die
// Folgegeneration wieder zurück
public static boolean[][] wendeRegelnAn(boolean[][] welt){
    boolean[][] welt_neu = new boolean[DIM1][DIM2];
    int nachbarn;
    for (int y=1; y<DIM2-1; y++)
        for (int x=1; x<DIM1-1; x++) {
            nachbarn = anzNachbarn(welt, x, y);
            if (welt[x][y]) {
                if ((nachbarn < 2) || (nachbarn > 3)) // Regel 1, 2
                    welt_neu[x][y] = false;
                if ((nachbarn == 2) || (nachbarn == 3)) // Regel 3
                    welt_neu[x][y] = true;
            } else {
                if (nachbarn == 3) // Regel 4
                    welt_neu[x][y] = true;  }  }
    return welt_neu;}
public static void main(String[] args) {
    // Welt initialisieren
    boolean[][] welt = initWelt();
    System.out.println("Startkonstellation");
    zeigeWelt(welt);
    for (int i=1; i<=100; i++){
        welt = wendeRegelnAn(welt);
        System.out.println("Generation "+i);
        zeigeWelt(welt);}}}
```

```java
public class AB411_04_PerformanceTesting {
    public static void main(String[] args) {
        int [] liste = new int[100000];
        for (int i=0; i<liste.length; i++)
            liste[i] = (int)(100*Math.random());
        long startCloneTime = System.currentTimeMillis();
        for (int j=0; j<10000;j++) {
            int[] cloneListe = (int[]) liste.clone();}
        long stopCloneTime = System.currentTimeMillis();
        long startCopyTime = System.currentTimeMillis();
        for (int j=0; j<10000;j++) {
            int[] copyListe = new int[liste.length];
            for (int n=0; n<liste.length;n++) {
                copyListe[n] = liste[n];  }  }
        long stopCopyTime = System.currentTimeMillis();
        long cloneTime = stopCloneTime - startCloneTime;
        long copyTime = stopCopyTime - startCopyTime;
        System.out.println("Dauer (clone): " + cloneTime + "ms");
        System.out.println("Dauer (elementweise Kopieren): " + copyTime + "ms");
    }
}
```

```java
public class AB411_01_collatz {
    public static void main(String[] args) {
        try {
            long number = collatzUserInput();
            System.out.println("Collatz-calculation for: " + number);
            ArrayList collatzProcedureList = collatz_procedure(number);
            System.out.println("Prodecdure calculation:");
            for (int i = 0; i < collatzProcedureList.size(); i++) {
                System.out.print(collatzProcedureList.get(i) + " ");}
            System.out.println("Functional calculation:");
            System.out.print(collatz_function(number));
        } catch (IOException ex) {
            Logger.getLogger(AB411_01_collatz.class.getName()).log(Level.SEVERE, null, ex);}}
    public static long collatzUserInput() throws IOException {
        long number = 1;
        BufferedReader input = new BufferedReader(new InputStreamReader(System.in));
        System.out.println("Entry: ");
        while (number % 2 != 0) {
            number = Long.parseLong(input.readLine());}
        return number;}
    public static ArrayList collatz_procedure(long n) {
        ArrayList<Long> collatzList = new ArrayList<Long>();
        while (n > 1) {
            if(n==1){
            } else if (n % 2 == 0) {
                n = n / 2;
            } else {
                n = 3*n + 1;
            }
            collatzList.add(n);}
        return collatzList;}
    public static long collatz_function(long n) {
        if(n==1){
        } else if (n % 2 == 0) {
            collatz_function(n / 2);
        } else {
            collatz_function(3*n + 1);}
        return n;}
}
```

```java
public class AB411_01_collatz {
    public static int collatz_function(long n){
        if (n == 1)
        {
            return 1;
        }
        else if (n % 2 == 0)
        {
            System.out.print((n / 2) + " ");
            return collatz_function(n / 2);
        }
        else
        {
            System.out.print((3*n + 1) + " ");
            return collatz_function(3 * n + 1);
        }
    }

    public static void collatz_procedure(long n){
        while (n > 1)
        { if (n % 2 == 0)
            {
                System.out.print((n / 2) + " ");
                n = n / 2;
            }
        else
            {
                System.out.print((3*n + 1) + " ");
               n = 3 * n + 1;
            }
        }
    }
    public static void main(String[] args) throws java.io.IOException {
        String eingabe;
        BufferedReader input = new BufferedReader(new InputStreamReader(System.in));
        System.out.println("Eingabe:");
        eingabe = input.readLine();
        long n = Long.parseLong(eingabe);
        System.out.println("Collatz-Berechnung für: " + eingabe);
        System.out.println("Prozedurale Berechnung:");
        collatz_procedure(n);
        System.out.println();
        System.out.println("Funktionale Berechnung:");
        collatz_function(n);
```

```java
public class AB411_03_WriteFile {
    public static void main(String[] args) throws IOException {
        String datei, in;
        datei="/home/vmadmin/Dokumente/M411/NetBeansProjects/AB411_03_WriteFile/Ausgabe.txt";
        BufferedReader input= new BufferedReader(new InputStreamReader(System.in));
        try {
            BufferedWriter myWriter= new BufferedWriter(new FileWriter(datei,false));
            do {
                System.out.println("Eingabe");
                in = input.readLine();
                if (!in.equals("q")) {
                    myWriter.write(in+"\n");
                } else {
                    myWriter.close();
                    break;  }
            } while (true);
        } catch (IOException eIO) {
            System.out.println("Folgender Fehler trat auf: "+eIO);
        }}}
```

```java
public class AB411_03_ReadFile {
    public static void main(String[] args) {
        String datei="/home/vmadmin/Dokumente/M411/NetBeansProjects/AB411_03_ReadFile/Liste.txt";
        try (Scanner scanner = new Scanner(new File(datei), "UTF-8")) {
            int counter = 0;
            while (scanner.hasNextLine()) {
                String line = scanner.nextLine();
                System.out.println("Zeile "+counter+": "+line);
                counter++;}
            scanner.close();
        } catch (FileNotFoundException e) {
                e.printStackTrace();}}}
```

Typname	Länge	Wertebereich	Standardwert
boolean	1	true, false	false
char	2	Alle Unicode-Zeichen	\u0000
byte	1	$-2^7 \ldots 2^7-1$	0
short	2	$-2^{15} \ldots 2^{15}-1$	0
int	4	$-2^{31} \ldots 2^{31}-1$	0
long	8	$-2^{63} \ldots 2^{63}-1$	0
float	4	$+/-3.40282347 * 10^{38}$	0.0
double	8	$+/-1.79769313486231570 * 10^{308}$	0.0

Einlesen von duz als String	
Eingabeprüfung - duz korrekt?	
j — duz korrekt — n	
dez := 0	Ausgabe Fehlermeldung
faktor := 1	Löschen des Eingabefeldes
Für i von Länge(duz) rückwärts_bis 1	Eingabefeld aktivieren
dez := dez + faktor * zahlwert(duz[i])	
faktor := faktor * 2	
Ausgabe von Textwert(dez)	

Notizen:
- Caesar, Xor (1^1 = 0, 1^0 = 1, 0^1 = 1, 0^0 = 0)
z.B: ArrayIndexOutofBoundsExeption
- pertesting: 10000 elementweise schneller 100000 clone schneller
Algorithums

begriffe:
geordnete menge: sequenzell wenn 1,2,3 oder nicht unbedingt nacheinander schritte oder auch verteilt.
Eindeutigkeit: gleiche eingabe=gleiches ergebnis,
muss bekannt sein was getan werden muss, aussert bei random
Effektivität: rechnerisch / mechanisch umsetzbar sein (!=Effizienz)
Terminierung: Hält nach endlichen schritten an, nicht loop
Korrektheit: Ein und ausgabeparameter