Programmieren 1
Selbststudium Semesterwoche 9

Maps und Sets

Kapitel 5.6
1. zu bearbeitende Aufgaben: 5.23 bis 5.30
2. Für welche Art von „Lookups“ sind Maps ideal?
→ Index-Lookups: Durch die eindeutige Zuordnung von Werten zu Indexen ist ein Auffinden eines Werts über einen Index vor allem bei grossen Datenmengen viel schneller als mit iterativen Methoden.

Kapitel 5.7
3. zu bearbeitende Aufgabe: 5.32
Kapitel 5.8
4. zu bearbeitende Aufgaben: 5.33, 5.35 und 5.36
Kapitel 5.9
Kapitel 5.10
5. zu bearbeitende Aufgaben 5.43 bis 5.45.
Hinweis zu Aufg. 5.43: Die Funktion lautet in Ihrer BlueJ Version Project Documentation.
6. Kommentieren Sie eines Ihrer eigenen Projekte (z.B. Balloon von SW 3) mit Hilfe von javadoc.

Kapitel 5.13
7. zu bearbeitende Aufgabe 5.58
→ `public static final float MEASURETOLERANCE = 0.001;`
→ `private static final int PASSMARK = 40;`
→ `public static final char HELPSHORTCUT = 'h';`

Algorithmen

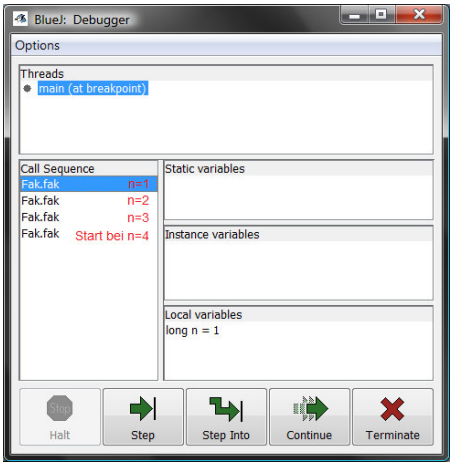
8. Implementieren Sie den rekursiven Algorithmus zur Berechnung der Fakultät (public long fak (long n), Folie 12) mit BlueJ. Beobachten Sie die Berechnung von fak(4) mit dem Debugger.

```
// Lösungsbeispiel zur rekursiven Berechnung der Fakultät von n
public class Fak
{
    public Fak()
    {

    }

    public long fak(long n)
    {
        if(n==0 || n==1)            //Rekursionsbedingung
        {
            return 1;               // Rekursionsbasis
        }
        else
        {
            return n * fak(n-1);    //Rekursionsvorschrift
        }
    }
}
```

→ Rekursiver Aufruf von fak(n) bis n der Bedingung n==1 entspricht. Danach werden die jeweiligen Resultate Schritt für Schritt an die zuletzt ausgeführte fak(n) Methode zurückgegeben und mit dem letzten n-Wert multipliziert.



→ Wichtig: Rekursion funktioniert nach dem Stack-Prinzip „First in – Last out“.

9. Implementieren Sie die rekursiven Methoden, um ein Array vorwärts und rückwärts auszugeben, in Java. (Folien 24 und 25)

//Lösungsbeispiel

```
import java.util.*;

public class RekArray
{
    public RekArray()
    {

    }

    public int[] printArray(int[] a, boolean direction)
    {
        if(direction)
        {
            return printArray(a, a.length-1, direction);
        }
        else
        {
            return printArray(a, 0, direction);
        }
    }

    private int[] printArray(int[] a, int index, boolean direction)
    {
        if(index<=a.length-1 && direction==false)   //Rekursionsvorschrift 1
        {
            System.out.println("Index "+index+"="+a[index]);
            index++;
            return(printArray(a, index, direction));    //Rekursionsbasis 1
        }
        else if(index>=0 && direction==true)        //Rekursionsvorschrift 2
        {
            System.out.println("Index "+index+"="+a[index]);
            index--;
            return(printArray(a, index, direction));    //Rekursionsbasis 2
        }
        else
        {
             return a;
        }
    }
}
```

10. Implementieren Sie eine rekursive Methode, welche die Summe aller Werte eines Arrays von Integern berechnet.

```
//Lösungsbeispiel

import java.util.*;

public class RekArraySum
{
    public RekArraySum()
    {

    }

    public int[] printArray(int[] a)
    {
        return(printArray(a, 0, 0));
    }

    private int[] printArray(int[] a, int index, int value)
    {
        if(index<=a.length-1)   //Rekursionsvorschrift
        {
            value = value + a[index];
            index++;
            return(printArray(a, index, value));    //Rekursionsbasis
        }
        else
        {
             System.out.println("Sum = "+value);
             return a;
        }
    }
}
```