1 Aufgabe 1

2 Aufgabe 2

a) Die äquivalente While-Struktur lautet

```
while(true)
  statement
```

b) So etwas kann nützlich sein, wenn man z.B. auf ein Ereignis von außen in einer Schleife wartet und man den Zeitpunkt nicht vorhersehen kann. Z.B. Netzwerktraffic, Eingaben über Peripherie etc.

3 Aufgabe 3

In Java benötigt man Ausnahmebehandlungen (Exceptions) um Ausnahmebehandlungen (z.B. Fehler) zu verarbeiten. Zu jeder Ausnahme kann eine eigene Klasse definiert werden, welche von Exception erbt. Hier können dann die spezifischen Vorgehensweise beim Auftreten dieser Ausnahme eingebettet werden. Damit der Programmablauf auch bei einer Ausnhame nicht unterbrochen wird, wird der Block, in dem eine Ausnahme erwartet wird mit try-catch bearbeitet. So wird das Programm stabiler und ist nicht mehr so anfällig für z.B. falsche Eingaben etc.

4 Aufgabe 4

Aufgaben vom 5. Zettel mit Exceptions ...

```
 public static double newton(double xi) throws
    IllegalArgumentException
{
        if(xi <= 1)
                throw new IllegalArgumentException("Wert␣
                   muss␣größer␣als␣1␣sein!");
        double xiAlt = xi;
        xi = xiAlt - ((xiAlt * xiAlt - 3) / (2*xiAlt));
        if(Math.abs(xi - xiAlt) < 0.0001) // Was für eine
           Genauigkeit
                return xi;
        else return newton(xi);
}
```

```
public static int gewicht(int[] feld)
{
        int summe = 0;
        try
        {
                for(int i = 0;i<feld.length;i++)
                {
                        summe = summe + feld[i];
                }
        }
        catch(NullPointerException e)
        {
                System.out.println(e + " Leeres Array");
                e.printStackTrace();
        }
        return summe;
}
```

Aufgabaen vom 6. Zettel mit Exceptions ...

```
public static int linSuche(double[] a, double w)
{
        try
        {
                for(int i = 0; i< a.length; i++)
                {
                        if(a[i] == w)
                                return i;
                }
        }
        catch(NullPointerException e)
        {
                System.out.println(e + " Leeres Array!");
                e.printStackTrace();
        }
        return -1;
}

public static int[] laufSum(int[] a)
{
  try
  {
```

```
    int summe =0;
    int[] b = new int[a.length];
    for(int i = 0; i< a.length; i++)
    {
      summe = summe + a[i];
      b[i] = summe;
    }
    return b;
  }
  catch(NullPointerException e)
  {
          System.out.println(e + " Leeres Array!");
          e.printStackTrace();
  }
  return null;
}
```

Aufgaben vom 7. Zettel mit Exceptions ...

```
static int binSearch(int[] a, int l, int r, int e) throws
   SearchException
{
        for(int i = l; i<r;i++)
        {
                if(a[i] > a[i+1])
                        throw new SearchException();
        }
        if(l<=r)
        {
                int m = ((l+r) / 2);
                System.out.println("m: " + m + " a[m]: " +
                   a[m] + " l: " + l + " r: " +r + " e: " +
                    e);
                if(a[m] > e)
                        return binSearch(a,l,m-1,e);
                else if(a[m] < e)
                        return binSearch(a,m+1,r,e);
                else if(a[m] == e)
                        return m;
        }
        return -1;
}
```

```
class SearchException extends Exception
{
        SearchException()
        {
                super();
        }
        SearchException(String fehler)
        {
                super(fehler);
        }
        public String getLocalizedMessage()
        {
                return "Array␣nicht␣sortiert,␣Binäre␣Suche␣
                   geht␣also␣nicht!";
        }
}

public static int skalarProdukt(int[] a, int[] b) throws
   ArrayIndexOutOfBoundsException
{
        int skalar = 0;
        if(a.length != b.length)
        {
                throw new ArrayIndexOutOfBoundsException("
                   Vektoren␣ungleich␣lang!");
        }
        for(int i = 0;i<a.length; i++)
        {
                skalar += a[i] * b[i];
        }
        return skalar;
}

public static int[][] negZahl(int[][] a)
{
  try
  {
     boolean neg = false;
         int counter = 0;
         int[][] b = new int[a.length][2];
```

```
            marke:
            for(int i=0;i<a.length;i++)
            {
                    neg = false;
                    for(int j = 0;j<a[i].length;j++)
                    {
                      if(!neg)
                      {
                            if(a[i][j] < 0)
                            neg  = true; // erste negative Zahl
                                gefunden
                      }
                      else
                      {
                            if(a[i][j] < 0)
                            {
                                    b[counter][0] = i;
                                    b[counter][1] = j;
                                    neg = false;
                                    counter++;
                                    continue marke;
                            }
                      }
                }
            }
            return b;
    }
    catch(NullPointerException e)
    {
            System.out.println(e);
            e.printStackTrace();
    }
    return null;
}
```