4 Arbeiten mit der Standardklasse Random

4.1. Erstellen Sie ein Programm zum Testen der Klasse Random nach folgenden Vorgaben:

- Der Benutzer wird aufgefordert einen Initialwert („seed“) für die Folge der Pseudo-Zufallszahlen einzugeben. Gibt der Benutzer einen Wert ungleich -1 ein, dann wird das Random-Objekt mit dem Initialwertwert erzeugt, sonst ohne.
- Der Benutzer hat dann noch die Möglichkeit einen oberen Grenzwert für die Zufallszahlen vorzugeben.
- Das Programm erzeugt dann eine Folge von 100 ganzzahligen Zufallszahlen und gibt diese am Bildschirm aus; jeweils 10 Zahlen in einer Zeile.

Testen Sie das Programm mit verschiedenen Eingaben. Was passiert bei Eingabe des selben Initialwertes und der selben oberen Grenze?

```
 7 import java.util.Random;
10 /**
11  * @author stk Kurzbeschreibung: Test der Klasse Random
12  */
13 public class A0401RandomTest
14 {
16      * @param args
19     public static void main(String[] args)
20     {
21         int iInit;
22         int iOGrenze;
23         Random zufall;
24         int i, j;
25
26         iInit = Eingabe.getInt("Initialwert (-1 für keinen) =");
27         iOGrenze = Eingabe.getInt("obere Grenze =");
28
29         if (iInit != -1)
30         {
31             zufall = new Random(iInit);
32         }
33         else
34         {
35             zufall = new Random();
36         }
37
38         for (i = 1; i <= 10; i++)
39         {
40             for (j = 1; j <= 10; j++)
41             {
42                 System.out.printf("%8d", zufall.nextInt(iOGrenze));
43             }
44             System.out.println();
45         }
46     }
47 }
```

4.2. Erstellen Sie eine Klasse `Zufall` mit folgenden **Klassen**methoden:

`double getZufallDouble()`	Liefert einen `double`-Zufallswert im Bereich zwischen 0.0 <= z < 1.0 Das Random Objekt wird ohne Initialwert erzeugt.
`double getZufallDouble(double dVon, double dBis)`	Liefert einen `double`-Zufallswert im Bereich zwischen `dVon` <= z < `dBis`
`int getZufallInt ()`	Liefert einen `int`-Zufallswert aus dem Zahlenbereich aller möglichen `int`-Zahlen
`int getZufallInt (int iVon, int iBis)`	Liefert einen `int`-Zufallswert im Bereich zwischen `iVon` <= z <= `iBis`

Testen Sie die Methoden in eine Main Methode.

```
54      * stellt einige Klassenmethoden zur Klasse Random zur Verfügung
55      */
56
57     /**
58      * @return double-Wert y:  0.0 <= y < 1.0
59      */
60     public static double getZufallDouble()
61     {
62         Random r = new Random();
63
64         return (r.nextDouble());
65     }
66
67     /**
68      *
69      * @param dVon
70      * Untergrenze
71      * @param dBis
72      * Obergrenze
73      * @return double-Wert y: dVon <= y < dBis
74      */
75     public static double getZufallDouble(double dVon, double dBis)
76     {
77         return A0402Zufall.getZufallDouble()*(dBis-dVon) + dVon;
78
79     }
80
81     /**
82      * @return int-Wert
83      */
84     public static int getZufallInt()
85     {
86         Random r = new Random();
87         return (r.nextInt());
88     }
89     /**
90      *
91      * @param iVon Untergrenze
92      * @param iBis Obergrenze
93      * @return int-Wert innerhalb von iVon (incl.) und iBis (incl.)
94      */
95     public static int getZufallInt (int iVon, int iBis)
96     {
97         int iZuf;
98         Random r = new Random();
99
100        iZuf = r.nextInt(iBis - iVon + 1)+iVon;
101        return (iZuf);
102    }
103 }
```

5 Arbeiten mit den Standardklassen Formatter und DecimalFormat

5.1. Erzeugen Sie einen "Zahlenpfeil" aus Zufallszahlen. z.B.

```
9
82
155
6225
96585
768535
44145
4077
675
33
3
```

Die Grundidee dabei ist, dass aus beliebig vielen Zufallszahlen zuerst eine einstellige, dann eine zweistellige usw. bis 6-stellig, dann rückwärts ermittelt und ausgegeben werden. Zur Ermittlung der Stellenzahl und zur Ausgabe auf dem Bildschirm wird die Zufallszahl in einen String umgewandelt.

```
7  import java.util.Random;
8
9  /**
10  * @author stk Kurzbeschreibung: * gibt Zahlenpfeil aus, z.B.
11  * 9
12  * 82
13  * 155
14  * 6225
15  * 96585
16  * 768535
17  * 44145
18  * 4077
19  * 675
20  * 33
21  * 3
22  */
23 public class A0501Zahlenpfeil
24 {
26     * @param args
29     public static void main(String[] args)
30     {
31         Random r = new Random();
32         String sInt;
33         int iAnz = 1;
34         while (iAnz < 6)
35         {
36             sInt = Integer.toString(r.nextInt(100000));
37             if (sInt.length() == iAnz)
38             {
39                 System.out.println(sInt);
40                 iAnz++;
41             }
42         }
43         while (iAnz > 0)
44         {
45             sInt = Integer.toString(r.nextInt(1000000));
46             if (sInt.length() == iAnz)
47             {
48                 System.out.println(sInt);
49                 iAnz--;
50             }
51         }
52     }
53 }
```

5.2. Es sollen bis zu 10 double-Zufallszahlen ermittelt werden. Die genaue Anzahl wird zufällig ermittelt. Diese sollen in einem String verkettet werden, jeweils durch ; getrennt. Anschließend soll dieser String wieder in einzelne doubles getrennt werden, die dann mit 3 Vor- und 5 Nachkommastellen ausgegeben werden.
Beispiel:
Kette: `72.13726205311184 ;413.98616499178144 ;8.1345711326591 ;`
`Nr. 1 072,13726`
`Nr. 2 413,98616`
`Nr. 3 008,13457`

Hinweise:
Zur Erzeugung der Zufallszahen können die Methoden aus Aufgabe 4.2 aufgerufen werden.
Um die Zahlenkette wieder zu zerlegen ein StringTokenizer Objekt verwenden.
Um die Zahl in einer Stringvariablen wieder in einen int Umzuwandeln kann man die Klassenmethode Double.parseDouble() verwenden..
Die Ausgabe von Fließkommazahlen mit führenden Nullen funktioniert nicht mit printf(), deshalb eine DecimalFormat Objekt verwenden.

```
 7 import java.text.DecimalFormat;
10
11 /**
12  * @author stk
13  * Kurzbeschreibung: Umformatierung von Zufallszahlen
14  */
15 public class A0502RandomKette
16 {
18      * @param args
21     public static void main(String[] args)
22     {
23         //Random r = new Random();
24         DecimalFormat df = new DecimalFormat("000.00000");
25         StringTokenizer st;
26         double dZahl;
27         String sKette="";
28         int iAnz;
29         int iZaehler;
30         int iCnt = 1;
31
32         iAnz = A0402Zufall.getZufallInt(1, 10);
33
34         for (iZaehler = 1; iZaehler <= iAnz; iZaehler++)
35         {
36             sKette = sKette + A0402Zufall.getZufallDouble(1, 1000) + " ;" ;
37         }
38
39         System.out.println("Kette: "+sKette);
40         st = new StringTokenizer(sKette, ";");
41         while (st.hasMoreTokens() == true)
42         {
43           dZahl = Double.parseDouble(st.nextToken());
44
45           System.out.println("Nr. "+ iCnt + " "+df.format(dZahl));
46           iCnt++;
47         }
48     }
49 }
```