## Welche Ziele möchte ich in den drei Stunden erreichen?

Primär möchte ich mich heute Nachmittag nur noch damit auseinandersetzen wie die Math.random()-Methode funktioniert und richtig verwendet wird. Ich fand die Verwendung damals bei Einführung schon umständlich und habe mich auf die java.lang.utils.Random-Klasse versteift, weil diese ihren Zweck wunderbar bisher erfüllt.

Wenn ich bis 16 Uhr damit fertig werden sollte oder absehbar ist, dass ich das noch bis Feierabend schaffe, will ich heute zusätzlich (Möglichkeit 2) die Quizzes der anderen Aufgaben machen, aber dann nur mit Laptop und Maus, wie es in der Prüfung sein wird.

## Wie möchte ich meine Ziele erreichen?

Ich werde mich nochmal kurz in die Sache einlesen oder in unsere alten Unterlagen und Aufgaben schauen und ansonsten nochmal eine Testklasse erstellen, in der ich die Testszenarien durchprobiere, die mir wichtig erscheinen und schauen, ob dieses Vorgehen auch Vorteile bietet, die die Random-Klasse nicht hat.

Den Laptop nehme ich von der Dockingstation und schließe nur Ladekabel und Maus an.

## Reflexion: Habe ich meine Ziele erreicht? Und was habe ich gelernt?

Ich habe mein Ziel erreicht. Es ist jetzt ca. 16:30 Uhr. Wird knapp mit den Quizzes.

Ich mach jetzt nur noch Kleinkram und später noch ein wenig quizzen, sowie übers Wochenende, soweit ich Zeit und Lust habe. Am Montag geht’s es dann genauso weiter.

Jetzt weiß ich wieder, wieso ich die Klasse Random mit seiner nextInt(x,y) bevorzuge; es ist weitaus komfortabler und weniger fehleranfällig. Aber ich konnte mich meiner selbstgestellten Herausforderung stellen, ohne mir diesmal helfen lassen zu müssen und so konnte ich diese Denksportaufgabe wenigstens nutzen mir selbst zu beweisen, dass ich es kann, wenn man mir nur genügend Zeit lässt. (Siehe gerne wieder Arbeitstagebuch).

## Arbeitstagebuch

Bisher habe ich mich etwas in die API eingelesen. Tatsächlich nutzt die Math.random()-Methode die nextDouble()-Methode aus der Klasse java.lang.util.Random. Ich hätte nicht gedacht, dass Utils, die weniger elementar anmuten als java.lang. davon Gebrauch machen. Eher hätte ich umgekehrt erwartet, dass alle anderen darauf aufbauen. Abhängigkeiten innerhalb der Java-API sind also wechselseitig und nicht eingleisig.

Ab dem Abstieg zur nextDouble-Methode finden sich weitere wechselseitige Abhängigkeiten zwischen dem Double-Wrapper, Math und Random, welche hier jetzt den Rahmen sprengen. Wichtig ist nur zu wissen: **Math.random() = random.nextDouble(),** wobei random hier eine Instanz und nicht die Klasse darstellt und in beiden Fällen ein Double zurückgeliefert wird der größer gleich 0 und kleiner 1 ist.

In unseren Szenarien hingegen waren aber stets ganzzahlige Werte vonnöten, welche einfach leichter mit der Methode random.nextInt(x,y) zu erzeugen sind, weil sie einem direkt ermöglicht einen inklusiven Startwert und einen exklusiven Endwert festzulegen.

Neben all den interessanten Methoden, die Random und Math sonst noch zu bietet haben, drängt sich mir aus der Erinnerung noch eine Frage wieder auf:

Wie ersetze ich **random.nextInt(x,y)** mit **Math.random()** gleich nochmal?

Zwischenstand: Den einfachen Bound (von 1 bis n-1) habe schonmal herausgefunden. Übrigens habe ich bewusst noch nicht in unsere alten Lösungen geschaut, sondern versuche es mir ganz bewusst nochmal selbst herzuleiten.

int gegenprobe0 = (int) (0\*10)+1;  
int gegenprobe099 = (int) (0.99\*10)+1;  
System.*out*.println(gegenprobe0); //1  
System.*out*.println(gegenprobe099); //10  
  
for(;;){  
 int number = (int) (Math.*random*()\*10)+1;  
 System.*out*.println(number); //1-10  
 UserInputScanner.*pressReturnToContinue*(new Scanner(System.*in*));  
}

Hier passiert bisher folgendes:

Ich erzeuge einen Wert zwischen 0.00 und 0.99

Diesen kann man nun nutzen, um einen Bruchteil des gewünschten Maximalwertes zu erzeugen, denn wir wollen ja Zahlen bis 10.

Danach wird mit +1 wird sichergestellt, dass die Zahl nicht 0 ist, wenn wir mindestens 1 wollen, also ein Intervall von 1-10. Erzeugt random() 0, haben wir so mindestens 1 und erzeugt er 9 (weil 0,99\*10 nie 10 ist), wird so auch die 10 noch erreicht.

Das int-Donwcasting muss hier stattfinden, damit wir in einen int speichern dürfen.

Aus all dem entnehme ich, dass ich das Problem fast gelöst habe. Bisher habe ich die Variante random.nextInt(int n) nachgestellt, also von 1 bis n-1, nun glaube ich aber, dass ich durch den Versuch sicherzustellen, dass wir nicht unter 1 und 10 landen, bereits den zweiten Ansatz mit drinnen habe. Ich muss die Untergrenze dem Rest hinzuaddieren.

Weil erzeuge ich einen Wert von 1 bis 10 passt das ja. Will ich jetzt aber beispielsweise zwischen 10 und 30 eine Zahl erzeugen, so muss ich bedenken:

Zwischen dem Intervall 10 bis 30, liegt also die Spannweite (Range) 20.

ABER diese Differenz bildet nicht den gewünschten Umfang ab. Hier tritt ein off-by-One Error ein, wenn man nicht daran denkt, dem max-Wert eins aufzuaddieren, da die Range um eins größer sein muss. (Es gibt immer eine Lücke weniger als Zaunpfähle).

Nochmal von oben aufgegriffen:

(Math.*random*()\*10)+1; hier wäre max(10)-min(1) = 9 und nicht 10, wie es dasteht. Also muss man anpassen range = max - min + 1;

Dem wird 10 aufaddiert, um bei 0 auf 10 zu kommen und bei 20 auf 30.

Final ergibt sich für mich jetzt also folgende Syntax:

int min = 10;  
int max = 30;  
int range = max - min + 1;  
int gegenprobe0 = (int) (0 \* (max - min + 1)) + min; //  
int gegenprobe099 = (int) (0.99 \* range) + min;  
System.*out*.println(gegenprobe0); //10  
System.*out*.println(gegenprobe099); //30  
  
for (; ; ) {  
 int number = (int) (Math.*random*() \* range) + min;  
 int number1 = (int) (Math.*random*() \* (max - min + 1)) + min;  
 int number2 = (int) (Math.*random*() \* (30 - 10 + 1)) + 10;  
 System.*out*.println(number); //10-30  
 UserInputScanner.*pressReturnToContinue*(new Scanner(System.*in*));  
}

Ein Bild, das Text, Screenshot, Schrift enthält.

Automatisch generierte Beschreibung

Also halten wir fest, eine Zahl zwischen x und y mit Math.random() erhält man durch:

int z = (int) (Math.random()\*(y–x+1))+x bzw. int z = (int) (Math.random()\*(max-min+1))+min