5.2 - Exceptions

Warum sind die Fehler aufgetreten?

1) Die for-Schleife in Zeile 70 ist eine Endlos-Schleife, im Endeffekt ist es das gleiche wie while(true) zu schreiben. Somit „denkt“ der Compilter, das Programm komme aus dieser Schleife nicht mehr heraus, wodurch das auf die Schleife folgende return-statement in Z. 80 niemals erreicht werden würde. Daher der Fehler „java: unreachable statement“. Das es eigentlich ein break-statement im try-Block gibt wird dabei vom Compiller missachtet. Da durch dieses sowieso im ersten Durchlauf des Loops abgebrochen wird, können wir hier die Endlos-Schleife durch eine beliebige andere Schleife ersetzen, z.B. eine Schleife über j von 1 bis 10.

2) In Zeile 157 wird im Try-Block eine IOException geworfen, im Catch-Block jedoch nur eine Runtime-Exception gefangen, und keine andere. Da die IO-Exception keine Subklasse der Runtime-Exception ist, tritt hier eine Exception auf die im Catch nicht gefangen werden kann.
   Dies lässt sich beheben, indem im Catch-Block statt einer Runtime-Exception eine IOException gefangen wird.

Zeilen der Ausgabe erklären:

Erstmal allgemein:
In der Main-Methode der Klasse UncertainException wird (mit Hilfe eines For-Loops und der Uncertain-Methode, die number als Parameter nimmt und dann uncertain_number() aufruft) jede uncertain_number() Methode von uncertain1() bis uncertain10() aufgerufen, in aufsteigender Reihenfolge.

Die Klasse hat auch einen Parameter i, der zu Beginn mit 0 initialisiert wird.

Daher bekommen wir erst Return-Wert von uncertain1(), dann von 2, dann 3 usw., bis wir bei uncertain10() sind.

Der Integer result wird bei jedem Aufruf von uncertain() mit 0 initialisiert.
Es wird nun zuerst ausgegeben, welche uncertain_number() Methode aufgerufen wird.
Dann wird in einem Try-Block uncertain_number() aufgerufen, und der Rückgabewert davon in result gespeichert.
Das result wird dann ausgegeben mit „result = “ + result + „, i = „+ uncertain.i, und anschließend wird die Klasse der möglicherweise im Try-Block geworfenen Exception ausgegeben mit „i = „ + uncertain.i + „ Exception = („ + e.getClass().getName() + „)“

Jetzt zu den einzelnen Ausgaben:

Zeile	Ausgabe	Erklärung
1	uncertain1()	Es ist der erste Durchlauf der For-Schleife in der Main-Methode, daher ist number=1, und

		uncertain1() wird aufgerufen, was in Zeile 12 des Codes ausgegeben wird, und zu dieser Ausgabe führt
2	i = 2 Exception (java.lang.NumberFormat Exception)	In uncertain1() wird der Klassenparameter i zwei mal um 1 erhöht, einmal im Try-Block (der auf jeden Fall aufgerufen wird), und einmal im finally, was auch auf jeden Fall aufgerufen wird. Im finally wird außerdem eine NumberFormatException geworfen, welche dann vom Catch der uncertain()-Methode aufgefangen wird. Dort wird dann das i = 2 ausgegeben, sowie der Name der Klasse der geworfenen Exception, daher java.lang.NumberFormatException
3	uncertain2()	Wir sind im nächsten Durchlauf des For-Loops in der Main-Methode, mit number=2, daher wird hier uncertain2() aufgerufen und dies ausgegeben
4	i = 1 Exception (java.lang.Runtime Exception)	i wurde erneut mit 0 initialisiert. Der Try-Block in uncertain2() wirft keine Exception, daher wird der Catch-Block nicht erreicht, und durch das break springen wir direkt aus der gesamten For-Schleife heraus, wodurch auch das i++ im finally nicht erreicht wird. Lediglich im return wird i um 1 erhöht, und ist somit 1. Desweiteren wird eine RuntimeException geworfen, welche im Catch-Block der uncertain() Methode aufgefangen wird, dort wird dann i und der Name der Exception ausgegeben, was zu unserer Ausgabe hier führt.
5	uncertain3()	Wir sind im nächsten Durchlauf des For-Loops in der Main-Methode, mit number=3, daher wird hier uncertain3() aufgerufen und dies ausgegeben *Die uncertain_number() Ausgaben erkläre ich ab hier nicht mehr, es ist immer der gleiche Grund, ich erwähne sie lediglich in dieser Tabelle*
6	result = 2, i = 3	Die Do-While Schleife von uncertain3() wird genau einmal ausgeführt, dabei wird eine Runtime Exception geworfen und gefangen. Im Catch, Finally und Return wird i jeweils um 1 erhöht, und ist somit 3. Jedoch wird i vor der letzen Erhöhung zurückgegeben, somit ist result = 2 und i = 3, was hier ausgegeben wird. Die Ausgabe erfolgt in Zeile 45 des Codes, welcher nach dem Switch-Case in dem Try-Block erfolgt, und erreicht wird, da uncertain3() keine Exception wirft. Der
7	uncertain4()	
8	result = 0, i = 2	Im Try-Block wird i erst zurückgegeben, solange es noch 0 ist, daher ist result = 0. Dann wird i um eins

		erhöht, und im finally ebenfalls. Der Catch-Block wird nicht erreicht, da keine Exception auftritt. Die Erhöhung von i im letzten Return wird ebenfalls nicht erreicht, da im Try-Block schon ein Wert returned wurde, und das zweite return somit nicht aufgerufen wird.
9	uncertain5()	
10	i = 0 Exception (java.lang. RuntimeException)	i wird keinmal erhöht, und bleibt somit bei 0. Im Finally wird eine RuntimeException geworfen, dadurch wird von uncertain() der Catch-Block erreicht und dort i und der Name der geworfenen Exception ausgegeben.
11	uncertain6()	
12	result = 1, i = 1	Im Try-Block von uncertain6() wird zwar eine Exception geworfen, diese jedoch nicht aufgefangen sondern ignoriert. Es wird jedoch außerhalb des Try-Blocks keine Exception geworfen, daher wirft die gesamte Methode keine Exception zurück an die aufrufende Methode, wodurch dort das Ende des Try-Blocks erreicht wird, wo result und i ausgegeben wird, der Catch-Block wird aber nicht mehr erreicht. Uncertain6() erhöht i um 1 und gibt es dann zurück, sodass result und i beides 1 ist.
13	uncertain7()	
14	result = 0, i = 1	Im Try-Block wird eine IOException geworfen, der Catch-Block fängt aber lediglich eine RuntimeException, weshalb der Catch-Block nicht erreicht wird. Lediglich das finally wird erreicht, wo i erst zurückgegeben wird (daher ist return = 0), und dann um 1 erhöht wird (daher ist i = 1), und da die gesamte Methode keine Exception wirft wird die Ausgabe am Ende vom Try von uncertain() erreicht und gibt hier unser result und i aus.
15	uncertain8()	
16	i = 2 Exception (java.lang.Runtime Exception)	Im Try wird eine NumberFormatException geworfen, und im Catch wird eine RuntimeException gefangen. Da RuntimeException eine Superklasse von NumberFormatException ist, wird die geworfene Exception auch gefangen und der Catch-Block erreicht. Dort wird i um 1 erhöht. Dann wird eine neue RuntimeException geworfen (welche an die aufrufende Methode zurückgeworfen wird). Im Finally wird i nochmal erhöht, und ist somit 2. Da eine Exception zurückkommt, wird der Catch-Block von uncertain() erreicht, wo i und der Name der geworfenen RuntimeException ausgegeben wird

17	uncertain9()	
18	result = 1, i = 2	Im Try wird eine ClassCastException geworfen, eine Subklasse von RuntimeException, welche im Catch-Block daher gefangen wird. Dort wird i zwar um 1 erhöht, jedoch trotz des returns nicht zurückgegeben, da dieses vom return im Finally „überschrieben wird“. Somit ist i nun 1, wird so im Finally zurückgegeben, daher return = 1, und dann wird i nochmals um 1 erhöht, daher i = 2. Es wird keine Exception außerhalb des Try-Blocks geworfen, daher findet die Ausgabe wieder am Ende des Try-Blocks von uncertain() statt.
19	uncertain10()	
20	result = 1, i = 0	Es wird eine IOException im Try geworfen und im Catch gefangen, ansonsten passiert dort jedoch nichts. I wird nicht erhöht und ist somit 0. Es wird 1 zurückgegeben, daher ist return = 1. Da außerhalb des Try-Blocks keine weitere Exception geworfen wird, findet die Ausgabe wieder am Ende des Try-Blocks von uncertain() statt.