12.07.2020

Hausarbeit SE1
“Kniffel”

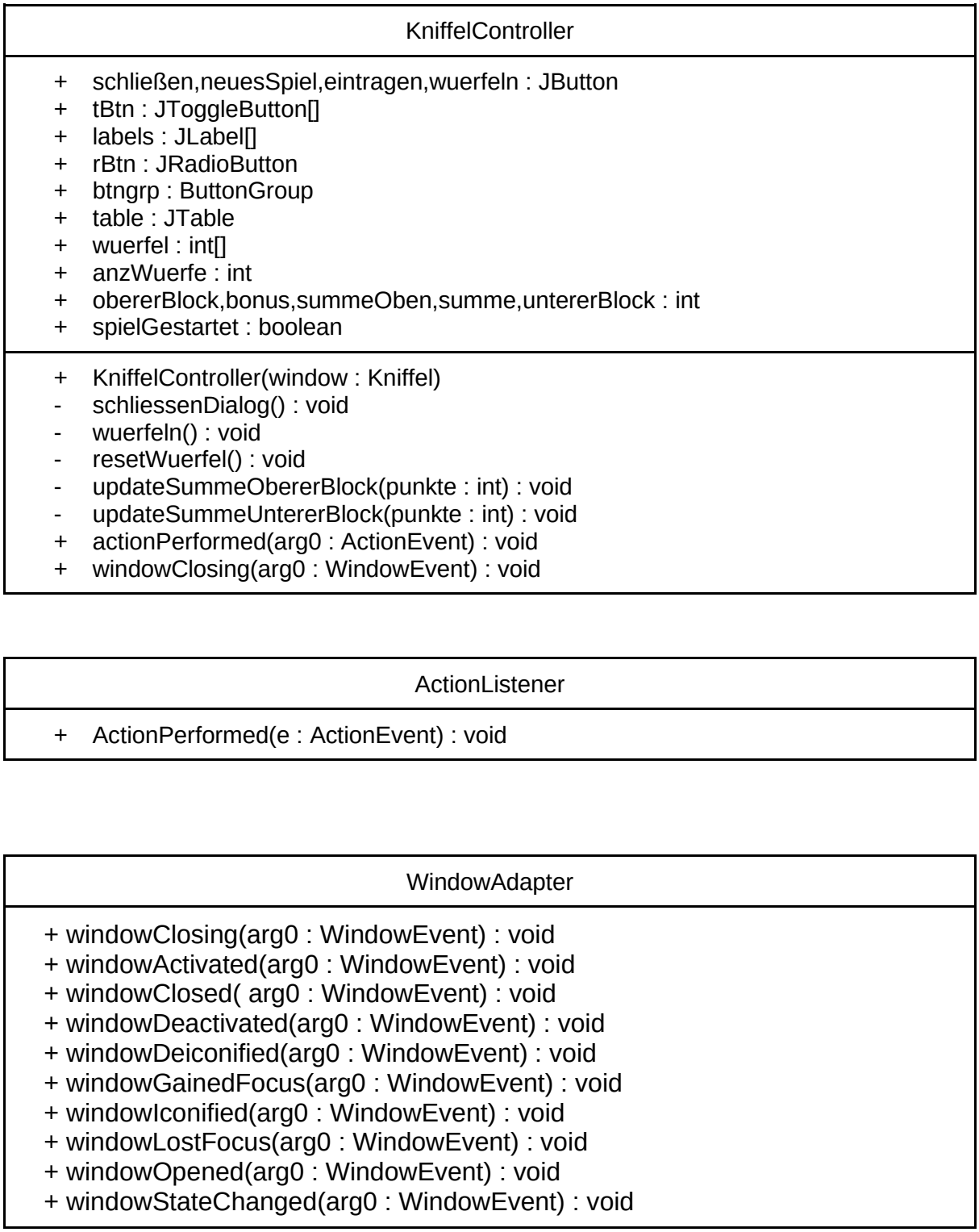
2IB – SS2020

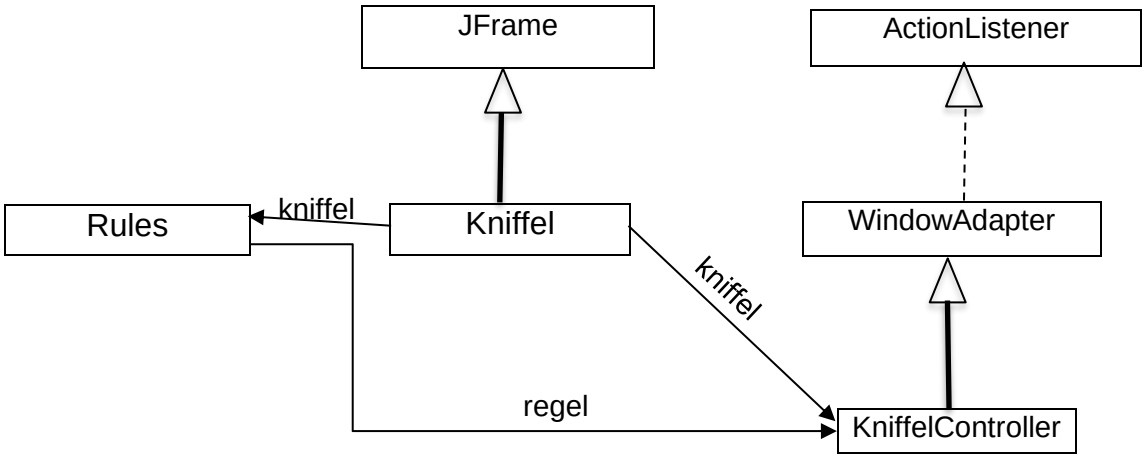
Atakan Ata – 1911877
Nils Rekus – 1826514
Frieder Widmann – 1911820

Die UML-Klassendiagramme wurden auf der 3.Seite der Übersichtlichkeit wegen, nur mit den Klassennamen dargestellt.

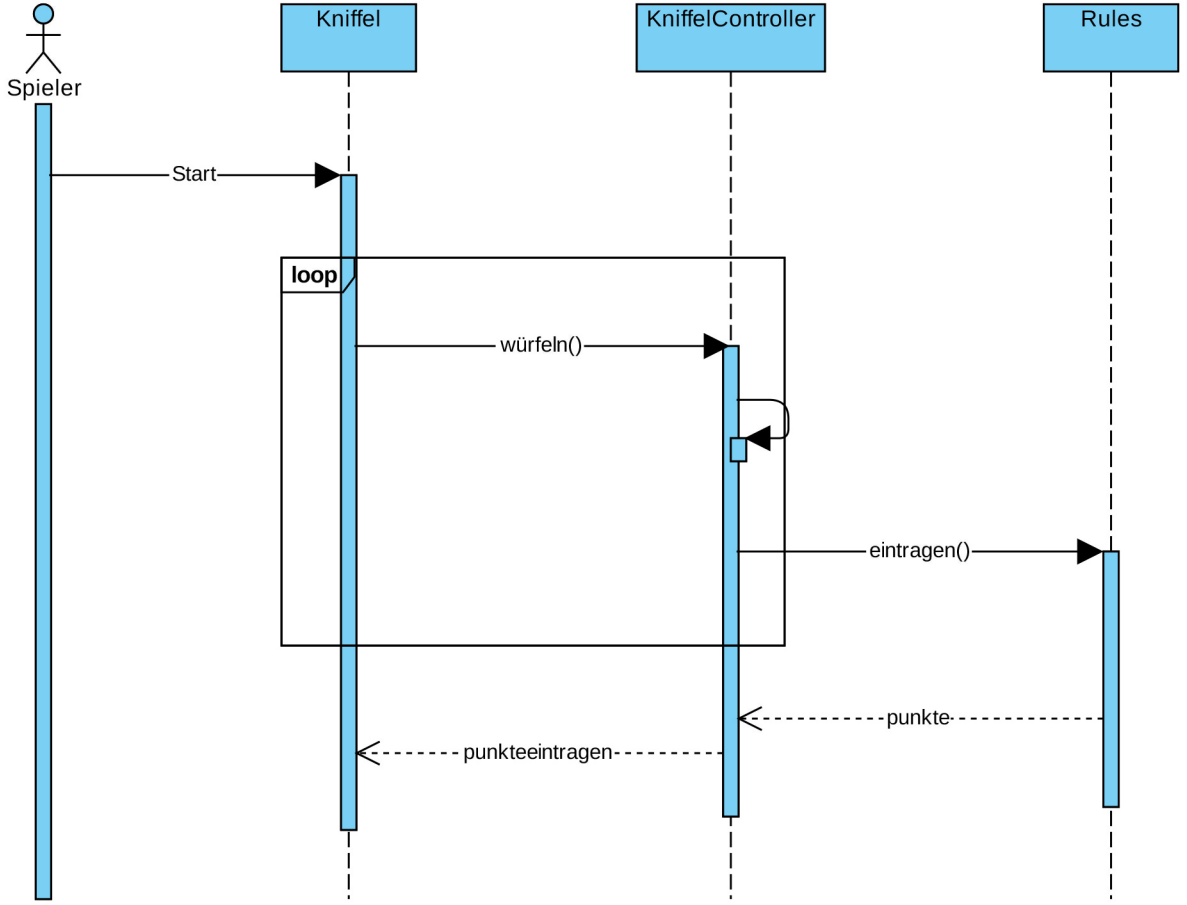
Kniffel
+ close,newGame,eintragen,roll : JButton + south,west,center,east,north,eastNumbers,eastButtons : JPanel + labels : JLabel[] + wuerfel : int[] + tBtn : JToggleButton[] + rBtn : JRadioButton[] + btnGrp : ButtonGroup + table : JTable + rBtnNames : String[] + fieldNames : String[] + fieldDescription : String[]
+ Kniffel() # erstelleButton():JToggleButton + get…(): … + set…(): ...

Rules
+ Rules(window : Kniffel) + einerRegel(wuerfel : int[]) : int + zweierRegel(wuerfel : int[]) : int + dreierRegel(wuerfel : int[]) : int + viererRegel(wuerfel : int[]) : int + fuenferRegel(wuerfel : int[]) : int + sechserRegel(wuerfel : int[]) : int + dreierpaschRegel(wuerfel : int[]) : int + viererpaschRegel(wuerfel : int[]) : int + fullhouseRegel(wuerfel : int[]) : int + kstrasseRegel(wuerfel : int[]) : int + gstrasseRegel(wuerfel : int[]) : int + kniffelRegel(wuerfel : int[]) : int + chanceRegel(wuerfel : int[]) : int + gleicheWuerfel(wuerfel : int[]) : int[] + selection(wuerfel : int[], gleicheWuerfel : int[], index : int) : int + testWuerfelzahl(wuerfelzahl : int) : void + testPunktezahl(fixePunktzahl : boolean, minimum : int, maximum : int, punkte : int):void


KniffelController
+ schließen,neuesSpiel,eintragen,wuerfeln : JButton
+ tBtn : JToggleButton[]
+ labels : JLabel[]
+ rBtn : JRadioButton
+ btngrp : ButtonGroup
+ table : JTable
+ wuerfel : int[]
+ anzWuerfe : int
+ obererBlock,bonus,summeOben,summe,untererBlock : int
+ spielGestartet : boolean
+ KniffelController(window : Kniffel)
- schliessenDialog() : void
- wuerfeln() : void
- resetWuerfel() : void
- updateSummeObererBlock(punkte : int) : void
- updateSummeUntererBlock(punkte : int) : void
+ actionPerformed(arg0 : ActionEvent) : void
+ windowClosing(arg0 : WindowEvent) : void
ActionListener
+ ActionPerformed(e : ActionEvent) : void
WindowAdapter
+ windowClosing(arg0 : WindowEvent) : void
+ windowActivated(arg0 : WindowEvent) : void
+ windowClosed(arg0 : WindowEvent) : void
+ windowDeactivated(arg0 : WindowEvent) : void
+ windowDeiconified(arg0 : WindowEvent) : void
+ windowGainedFocus(arg0 : WindowEvent) : void
+ windowIconified(arg0 : WindowEvent) : void
+ windowLostFocus(arg0 : WindowEvent) : void
+ windowOpened(arg0 : WindowEvent) : void
+ windowStateChanged(arg0 : WindowEvent) : void


JFrame
ActionListener
Rules
kniffel
Kniffel
WindowAdapter
kniffel
regel
KniffelController

Sequenzdiagramm



Blackbox-Testtabellen

Kommentar: Testtabellen für die wuerfeln()-Methode und alle Regel-Methoden.
Die Testtabellen decken lediglich die Validität von Würfel-Zahlenfolgen ab und stellen die daraus resultierenden Punkte dar.
Auf die Validität von Punkten abhängig von der Würfel-Zahlenfolge wurde in den BB-Testfällen nicht eingegangen, da dies nicht innerhalb der gleichen Testfällen umsetzbar gewesen wäre. Es lassen sich keine konkreten Äquivalenzklassen definieren, wenn zwei unterschiedliche Parameter auf ihre Gültigkeit überprüft werden müssen, wobei die Gültigkeit (gültigen Wertebereiche) des Parameters "Punkte" zusätzlich vom Parameter "Zahlenfolge" abhängt. Es wäre also für jede Regel eine zweite Testtabelle nötig gewesen, was nicht der Aufgabenstellung entsprochen hätte.

KniffelController.java: wuerfeln()

Je Testfall werden Zahlenfolgen mit fünf ganzen Zahlen getestet. Einzelne Zahlen können in einer Zahlenfolge mehrmals auftreten.

		Äquivalenzklassen		Randwerte, krit. Werte	TF 1, gültig	TF 2, gültig	TF 3, gültig	TF 3, ungültig	TF 4, ungültig	TF 5, ungültig
Parameter	**Array mit fünf Zahlen**	Enthält die Zahlen 1,2,3,4,5 oder 6	g	1,6	{1,2,3,4,6}	{1,1,1,1,1}	{6,6,6,6,6}			
		Enthält Zahl <0	u	-1				{-1,5,4,3,2}		
		Enthält Zahl 0	u	0					{1,0,6,6,4}	
		Enthält Zahlen >6	u	7						{5,3,5,1,7}
Exception								Würfelzahl <1 oder >6	Würfelzahl <1 oder >6	Würfelzahl <1 oder >6

Rules.java: einerRegel()

Je Testfall werden Zahlenfolgen mit fünf ganzen Zahlen getestet. Einzelne Zahlen können in einer Zahlenfolge mehrmals auftreten.
Regel für Punkteberechnung: Nur Einer zählen

		Äquivalenzklassen		Randwerte, krit. Werte	TF 1, gültig	TF 2, gültig	TF 3, gültig	TF 3, ungültig	TF 4, ungültig	TF 5, ungültig
Ausgangszustand	**Punktzahl als ganze Zahl**		g	0	0	0	0	0	0	0
Parameter	**Array mit fünf Zahlen**	Enthält die Zahlen 1,2,3,4,5 oder 6	g	1,6	{1,1,1,1,1}	{1,2,3,4,5}	{6,6,6,6,6}			
		Enthält Zahl <0	u	-1				{-1,1,1,1,1}		
		Enthält Zahl 0	u	0					{1,0,6,5,3}	
		Enthält Zahlen >6	u	7						{5,1,3,4,7}
Exception								Würfelzahl <1 oder >6	Würfelzahl <1 oder >6	Würfelzahl <1 oder >6
Endzustand	**Punktzahl als ganze Zahl**				5	1	0			

Rules.java: zweierRegel()

Je Testfall werden Zahlenfolgen mit fünf ganzen Zahlen getestet. Einzelne Zahlen können in einer Zahlenfolge mehrmals auftreten.
Regel für Punkteberechnung: Nur Zweier zählen

		Äquivalenzklassen		Randwerte, krit. Werte	TF 1, gültig	TF 2, gültig	TF 3, gültig	TF 3, ungültig	TF 4, ungültig	TF 5, ungültig
Ausgangszustand	**Punktzahl als ganze Zahl**		g	0	0	0	0	0	0	0
Parameter	**Array mit fünf Zahlen**	Enthält die Zahlen 1,2,3,4,5 oder 6	g	1,6	{2,2,2,2,2}	{2,1,3,4,5}	{6,6,6,6,6}			
		Enthält Zahl <0	u	-1				{-2,2,2,2,2}		
		Enthält Zahl 0	u	0					{2,0,6,5,3}	
		Enthält Zahlen >6	u	7						{5,2,3,4,7}
Exception								Würfelzahl <1 oder >6	Würfelzahl <1 oder >6	Würfelzahl <1 oder >6
Endzustand	**Punktzahl als ganze Zahl**				10	2	0			

Rules.java: dreierRegel()

Je Testfall werden Zahlenfolgen mit fünf ganzen Zahlen getestet. Einzelne Zahlen können in einer Zahlenfolge mehrmals auftreten.
Regel für Punkteberechnung: Nur Dreier zählen

		Äquivalenzklassen		Randwerte, krit. Werte	TF 1, gültig	TF 2, gültig	TF 3, gültig	TF 3, ungültig	TF 4, ungültig	TF 5, ungültig
Ausgangszustand	**Punktzahl als ganze Zahl**		g	0	0	0	0	0	0	0
Parameter	**Array mit fünf Zahlen**	Enthält die Zahlen 1,2,3,4,5 oder 6	g	1,6	{3,3,3,3,3}	{2,1,3,4,5}	{6,6,6,6,6}			
		Enthält Zahl <0	u	-1				{-3,3,3,3,3}		
		Enthält Zahl 0	u	0					{2,0,6,5,3}	
		Enthält Zahlen >6	u	7						{5,2,3,4,7}
Exception								Würfelzahl <1 oder >6	Würfelzahl <1 oder >6	Würfelzahl <1 oder >6
Endzustand	**Punktzahl als ganze Zahl**				15	3	0			

Rules.java: viererRegel()

Je Testfall werden Zahlenfolgen mit fünf ganzen Zahlen getestet. Einzelne Zahlen können in einer Zahlenfolge mehrmals auftreten.
Regel für Punkteberechnung: Nur Vierer zählen

		Äquivalenzklassen		Randwerte, krit. Werte	TF 1, gültig	TF 2, gültig	TF 3, gültig	TF 3, ungültig	TF 4, ungültig	TF 5, ungültig
Ausgangszustand	**Punktzahl als ganze Zahl**		g	0	0	0	0	0	0	0
Parameter	**Array mit fünf Zahlen**	Enthält die Zahlen 1,2,3,4,5 oder 6	g	1,6	{4,4,4,4,4}	{2,1,3,4,5}	{6,6,6,6,6}			
		Enthält Zahl <0	u	-1				{-4,4,4,4,4}		
		Enthält Zahl 0	u	0					{2,0,6,4,3}	
		Enthält Zahlen >6	u	7						{5,2,3,4,7}
Exception								Würfelzahl <1 oder >6	Würfelzahl <1 oder >6	Würfelzahl <1 oder >6
Endzustand	**Punktzahl als ganze Zahl**				20	4	0			

Rules.java: fuenferRegel()

Je Testfall werden Zahlenfolgen mit fünf ganzen Zahlen getestet. Einzelne Zahlen können in einer Zahlenfolge mehrmals auftreten.
Regel für Punkteberechnung: Nur Fünfer zählen

		Äquivalenzklassen		Randwerte, krit. Werte	TF 1, gültig	TF 2, gültig	TF 3, gültig	TF 3, ungültig	TF 4, ungültig	TF 5, ungültig
Ausgangszustand	**Punktzahl als ganze Zahl**		g	0	0	0	0	0	0	0
Parameter	**Array mit fünf Zahlen**	Enthält die Zahlen 1,2,3,4,5 oder 6	g	1,6	{5,5,5,5,5}	{2,1,3,4,5}	{6,6,6,6,6}			
		Enthält Zahl <0	u	-1				{-5,5,5,5,5}		
		Enthält Zahl 0	u	0					{2,0,6,5,3}	
		Enthält Zahlen >6	u	7						{5,2,3,4,7}
Exception								Würfelzahl <1 oder >6	Würfelzahl <1 oder >6	Würfelzahl <1 oder >6
Endzustand	**Punktzahl als ganze Zahl**				25	5	0			

Rules.java: sechserRegel()

Je Testfall werden Zahlenfolgen mit fünf ganzen Zahlen getestet. Einzelne Zahlen können in einer Zahlenfolge mehrmals auftreten.
Regel für Punkteberechnung: Nur Sechser zählen

		Äquivalenzklassen		Randwerte, krit. Werte	TF 1, gültig	TF 2, gültig	TF 3, gültig	TF 3, ungültig	TF 4, ungültig	TF 5, ungültig
Ausgangszustand	**Punktzahl als ganze Zahl**		g	0	0	0	0	0	0	0
Parameter	**Array mit fünf Zahlen**	Enthält die Zahlen 1,2,3,4,5 oder 6	g	1,6	{6,6,6,6,6}	{2,1,3,4,6}	{1,1,1,1,1}			
		Enthält Zahl <0	u	-1				{-6,6,6,6,6}		
		Enthält Zahl 0	u	0					{2,0,6,5,3}	
		Enthält Zahlen >6	u	7						{6,2,3,4,7}
Exception								Würfelzahl <1 oder >6	Würfelzahl <1 oder >6	Würfelzahl <1 oder >6
Endzustand	**Punktzahl als ganze Zahl**				30	6	0			

Rules.java: dreierpaschRegel()

Je Testfall werden Zahlenfolgen mit fünf ganzen Zahlen getestet. Einzelne Zahlen können in einer Zahlenfolge mehrmals auftreten.
Regel für Punkteberechnung: Drei gleiche Zahlen – Alle Augen zählen

		Äquivalenzklassen		Randwerte, krit. Werte	TF 1, gültig	TF 2, gültig	TF 3, gültig	TF 3, ungültig	TF 4, ungültig	TF 5, ungültig
Ausgangszustand	**Punktzahl als ganze Zahl**		g	0	0	0	0	0	0	0
Parameter	**Array mit fünf Zahlen**	Enthält die Zahlen 1,2,3,4,5 oder 6	g	1,6	{6,6,6,6,6}	{3,3,3,1,6}	{1,1,1,1,1}			
		Enthält Zahl <0	u	-1				{-6,6,6,6,6}		
		Enthält Zahl 0	u	0					{2,0,1,1,1}	
		Enthält Zahlen >6	u	7						{6,6,6,4,7}
Exception								Würfelzahl <1 oder >6	Würfelzahl <1 oder >6	Würfelzahl <1 oder >6
Endzustand	**Punktzahl als ganze Zahl**				30	16	5			

Rules.java: viererpaschRegel()

Je Testfall werden Zahlenfolgen mit fünf ganzen Zahlen getestet. Einzelne Zahlen können in einer Zahlenfolge mehrmals auftreten.
Regel für Punkteberechnung: Vier gleiche Zahlen – Alle Augen zählen

		Äquivalenzklassen		Randwerte, krit. Werte	TF 1, gültig	TF 2, gültig	TF 3, gültig	TF 3, ungültig	TF 4, ungültig	TF 5, ungültig
Ausgangszustand	**Punktzahl als ganze Zahl**		g	0	0	0	0	0	0	0
Parameter	**Array mit fünf Zahlen**	Enthält die Zahlen 1,2,3,4,5 oder 6	g	1,6	{6,6,6,6,6}	{3,3,3,3,6}	{1,1,1,1,1}			
		Enthält Zahl <0	u	-1				{-6,6,6,6,6}		
		Enthält Zahl 0	u	0					{1,0,1,1,1}	
		Enthält Zahlen >6	u	7						{6,6,6,6,7}
Exception								Würfelzahl <1 oder >6	Würfelzahl <1 oder >6	Würfelzahl <1 oder >6
Endzustand	**Punktzahl als ganze Zahl**				30	18	5			

Rules.java: fullhouseRegel()

Je Testfall werden Zahlenfolgen mit fünf ganzen Zahlen getestet. Einzelne Zahlen können in einer Zahlenfolge mehrmals auftreten.
Regel für Punkteberechnung: Drei gleiche und zwei gleiche, andere Zahlen – 25 Punkte

		Äquivalenzklassen		Randwerte, krit. Werte	TF 1, gültig	TF 2, gültig	TF 3, gültig	TF 3, ungültig	TF 4, ungültig	TF 5, ungültig
Ausgangszustand	**Punktzahl als ganze Zahl**		g	0	0	0	0	0	0	0
Parameter	**Array mit fünf Zahlen**	Enthält die Zahlen 1,2,3,4,5 oder 6	g	1,6	{6,6,6,1,1}	{1,2,3,4,5}	{1,1,1,1,1}			
		Enthält Zahl <0	u	-1				{-5,5,6,6,6}		
		Enthält Zahl 0	u	0					{0,0,1,1,1}	
		Enthält Zahlen >6	u	7						{6,6,6,7,7}
Exception								Würfelzahl <1 oder >6	Würfelzahl <1 oder >6	Würfelzahl <1 oder >6
Endzustand	**Punktzahl als ganze Zahl**				25	0	0			

Rules.java: kstrasseRegel()

Je Testfall werden Zahlenfolgen mit fünf ganzen Zahlen getestet. Einzelne Zahlen können in einer Zahlenfolge mehrmals auftreten.
Regel für Punkteberechnung: 1-2-3-4, 2-3-4-5, oder 3-4-5-6 – 30 Punkte

		Äquivalenzklassen		Randwerte, krit. Werte	TF 1, gültig	TF 2, gültig	TF 3, gültig	TF 4, ungültig	TF 5, ungültig	TF 6, ungültig
Ausgangszustand	**Punktzahl als ganze Zahl**		g	0	0	0	0	0	0	0
Parameter	**Array mit fünf Zahlen**	Enthält die Zahlen 1,2,3,4,5 oder 6	g	1,6	{1,2,3,4,5}	{6,5,4,3,1}	{1,3,6,4,6}			
		Enthält Zahl <0	u	-1				{-1,2,3,4,5}		
		Enthält Zahl 0	u	0					{1,0,2,3,4}	
		Enthält Zahlen >6	u	7						{3,4,5,6,7}
Exception								Würfelzahl <1 oder >6	Würfelzahl <1 oder >6	Würfelzahl <1 oder >6
Endzustand	**Punktzahl als ganze Zahl**				30	30	0			

Rules.java: gstrasseRegel()

Je Testfall werden Zahlenfolgen mit fünf ganzen Zahlen getestet. Einzelne Zahlen können in einer Zahlenfolge mehrmals auftreten.
Regel für Punkteberechnung: 1-2-3-4-5 oder 2-3-4-5-6 – 40 Punkte

		Äquivalenzklassen		Randwerte, krit. Werte	TF 1, gültig	TF 2, gültig	TF 3, gültig	TF 4, ungültig	TF 5, ungültig	TF 6, ungültig
Ausgangszustand	**Punktzahl als ganze Zahl**		g	0	0	0	0	0	0	0
Parameter	**Array mit fünf Zahlen**	Enthält die Zahlen 1,2,3,4,5 oder 6	g	1,6	{1,2,3,4,5}	{6,5,4,3,2}	{1,2,3,4,6}			
		Enthält Zahl <0	u	-1				{-1,2,3,4,5}		
		Enthält Zahl 0	u	0					{1,0,2,3,4}	
		Enthält Zahlen >6	u	7						{3,4,5,6,7}
Exception								Würfelzahl <1 oder >6	Würfelzahl <1 oder >6	Würfelzahl <1 oder >6
Endzustand	**Punktzahl als ganze Zahl**				40	40	0			

Rules.java: kniffelRegel()

Je Testfall werden Zahlenfolgen mit fünf ganzen Zahlen getestet. Einzelne Zahlen können in einer Zahlenfolge mehrmals auftreten.
Regel für Punkteberechnung: Fünf gleiche Zahlen – 50 Punkte

		Äquivalenzklassen		Randwerte, krit. Werte	TF 1, gültig	TF 2, gültig	TF 3, gültig	TF 4, ungültig	TF 5, ungültig	TF 6, ungültig
Ausgangszustand	**Punktzahl als ganze Zahl**		g	0	0	0	0	0	0	0
Parameter	**Array mit fünf Zahlen**	Enthält die Zahlen 1,2,3,4,5 oder 6	g	1,6	{1,1,1,1,1}	{6,6,6,6,6}	{1,6,6,6,6}			
		Enthält Zahl <0	u	-1				{-1,-1,-1,-1,-1}		
		Enthält Zahl 0	u	0					{0,0,0,0,0}	
		Enthält Zahlen >6	u	7						{7,7,7,7,7}
Exception								Würfelzahl <1 oder >6	Würfelzahl <1 oder >6	Würfelzahl <1 oder >6
Endzustand	**Punktzahl als ganze Zahl**				50	50	0			

Rules.java: chanceRegel()

Je Testfall werden Zahlenfolgen mit fünf ganzen Zahlen getestet. Einzelne Zahlen können in einer Zahlenfolge mehrmals auftreten.
Regel für Punkteberechnung: Alle Augen zählen

		Äquivalenzklassen		Randwerte, krit. Werte	TF 1, gültig	TF 2, gültig	TF 3, gültig	TF 4, ungültig	TF 5, ungültig	TF 6, ungültig
Ausgangszustand	**Punktzahl als ganze Zahl**		g	0	0	0	0	0	0	0
Parameter	**Array mit fünf Zahlen**	Enthält die Zahlen 1,2,3,4,5 oder 6	g	1,6	{1,1,1,1,1}	{6,6,6,6,6}	{1,2,3,5,5}			
		Enthält Zahl <0	u	-1				{-1,1,2,3,6}		
		Enthält Zahl 0	u	0					{2,0,4,4,2}	
		Enthält Zahlen >6	u	7						{6,2,1,4,7}
Exception								Würfelzahl <1 oder >6	Würfelzahl <1 oder >6	Würfelzahl <1 oder >6
Endzustand	**Punktzahl als ganze Zahl**				5	30	16			