SS2020

Hausarbeit SE1

Elektronische Variante des Spiels Kniffel

von Exoucia Mukubay (1925658) & Lilly Schumann (1924363)

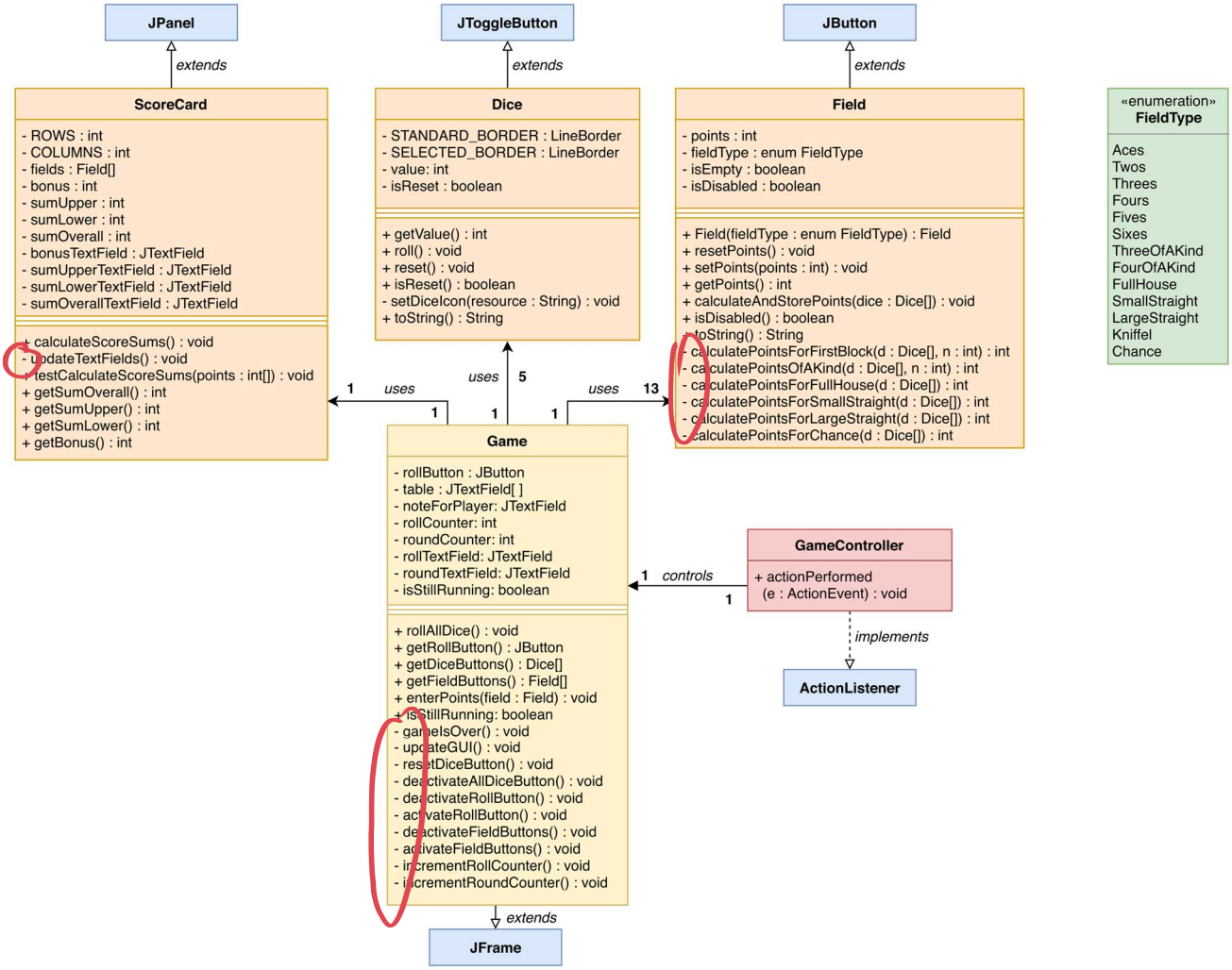
Inhaltsverzeichnis

Klassendiagramm

Wenn der Spieler ein Feld auswählt, um Punkte einzutragen oder auf den „Würfeln“-Knopf klickt, wird dieses Event von der Klasse „*GameController*“ registriert und entsprechende an die „*Game*“-Klasse weitergegeben. Die Klasse „*Game*“ umfasst das Graphic User Interface des Kniffel-Spiels und stellt die zentrale Einheit der Spiellogik dar.

Sie verwendet ein Objekt der Klasse „*ScoreCard*“, die für die Punkteberechnung verantwortlich ist. Die Felder sind von der Klasse „*Field*“ und beinhalten die eigentlichen Berechnungsalgorithmen für die verschiedenen Optionen. Um die verschiedenen Optionen zu unterscheiden, verwenden wir das Enum „*FieldType*“.

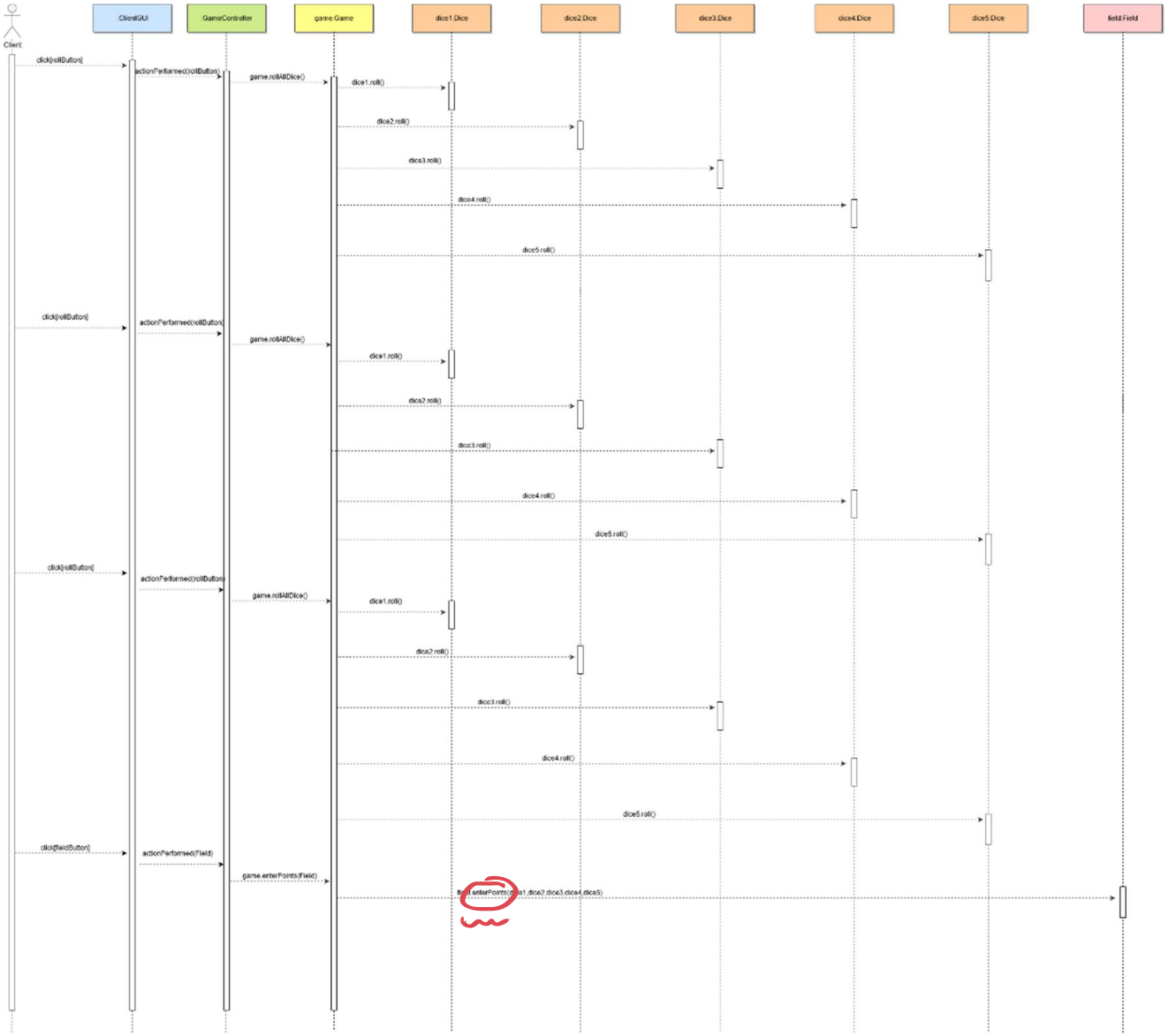
Die Klasse „*Dice*“ ist für die Würfel und deren Darstellung in der GUI verantwortlich.



Sequenzdiagramme

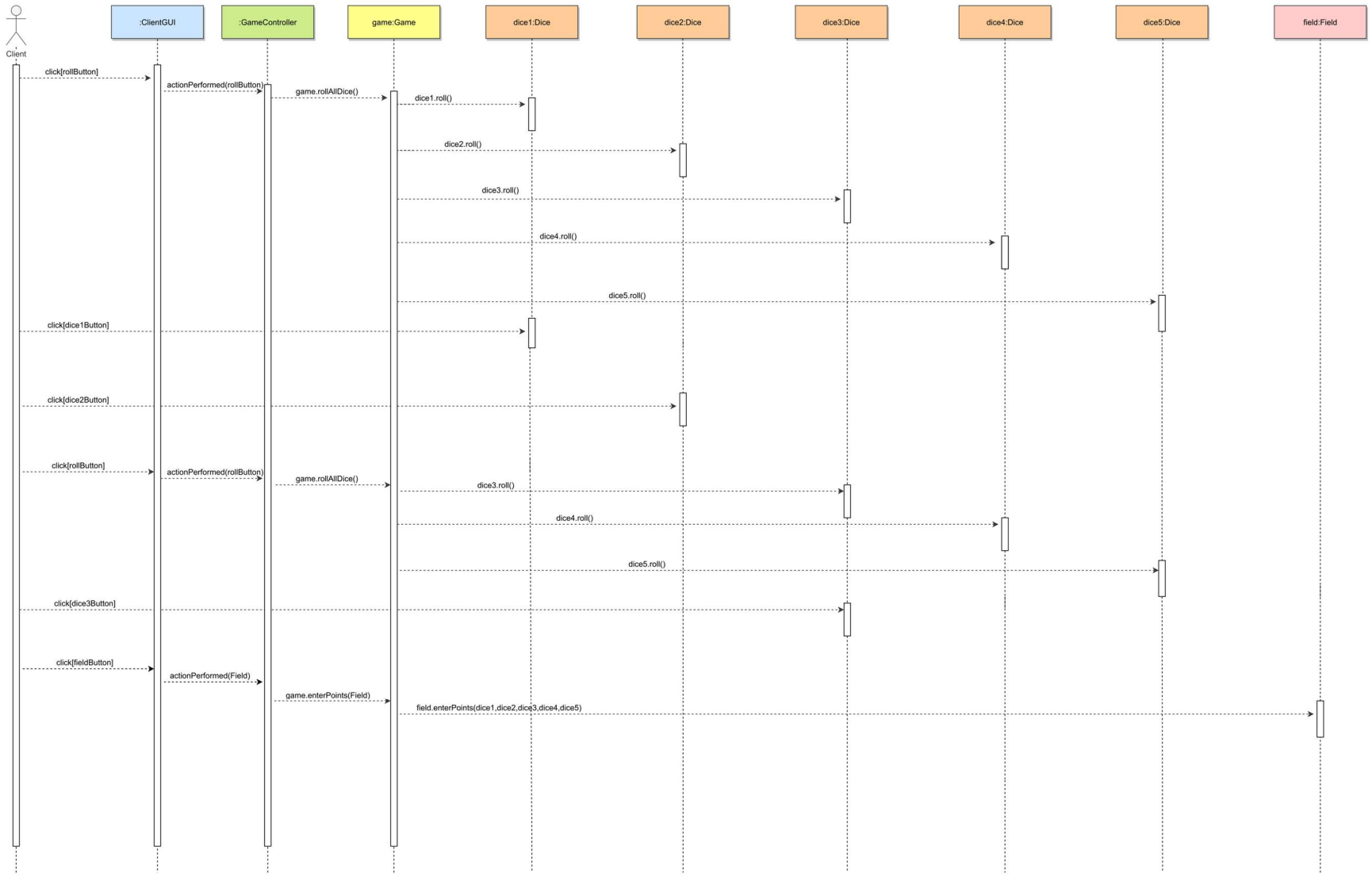
a) Dreimal hintereinander würfeln

Das Sequenzdiagramm beschreibt den Ablauf, wenn der Spieler dreimal hintereinander würfelt und nach dem letzten Wurf gezwungen wird ein Feld auszuwählen, um die Ergebnisse einzutragen.



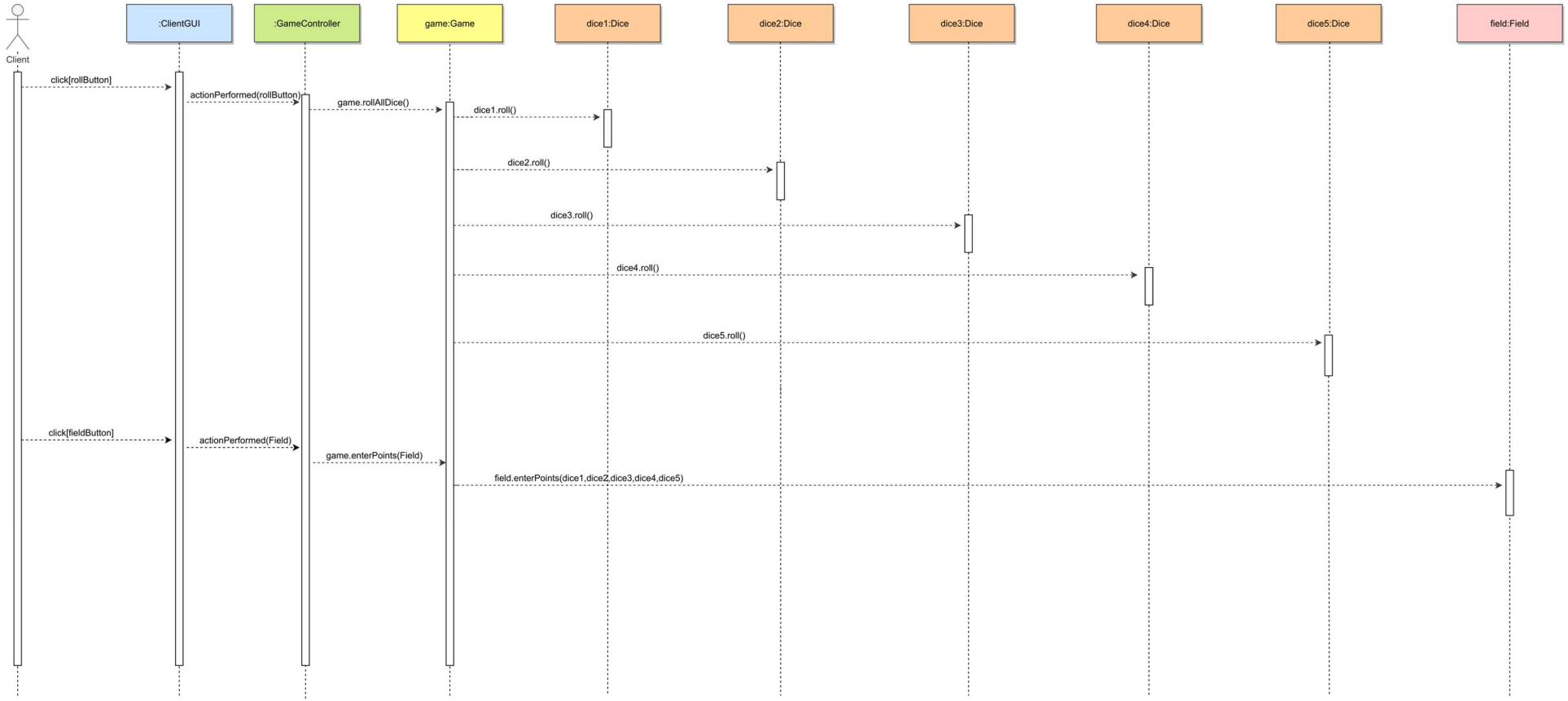
b) Würfel nach dem ersten und zweiten Wurf behalten

Das Sequenzdiagramm beschreibt den Ablauf, wenn der Spieler nach dem ersten Wurf zwei Würfel auswählt und behält und mit den verbleibenden weiter würfelt. Nach dem zweiten Wurf behält er noch einen Würfel und nach dem letzten Wurf wählt er ein freies Feld aus.



c) Einmal würfeln

Das Sequenzdiagramm beschreibt den Ablauf, wenn der Spieler nach dem ersten Wurf ein Feld auswählt, welches mit dem Ergebnis dieses Wurfs bewertet wird.



Testfall-Tabellen

a) Punkteberechnung testen

In den BlackBox-Tabellen wird die ungültige Äquivalenzklasse bei Feld, zum Beispiel "null", nicht getestet, da im Spiel die Felder(Einser, Zweier, etc.) vorgegeben sind und keine Möglichkeit besteht ein neues (ungültiges) Feld hinzuzufügen oder anzuklicken. Zudem gibt es keine ungültigen Testfälle bezüglich ungültigen Würfels, da wir in Abschnitt c) bereits testen, dass nur Zahlen zwischen eins und sechs gewürfelt werden können.

1. Einser

Die Methode "*calculateAndStorePoints*" muss beim ausgewählten Feld "Aces" gründlich getestet werden, damit nur die gleichen Würfelwerte, die die 1 haben, summiert werden. Somit wird ausgeschlossen, dass noch andere Würfelwerte mit dazugerechnet werden. Zum Schluss wird ein korrektes Ergebnis in das Feld eingetragen.

Methode: calculateAndStorePoints											
		Äquivalenzklassen		**Randwerte, krit. Werte**	**TF 1, gültig**	**TF 2, gültig**	**TF 3, gültig**	**TF 4, gültig**	**TF 5, gültig**	**TF 6, gültig**
Startzustand	FieldType		g	Aces	Aces	Aces	Aces	Aces	Aces	Aces
	diceValue		g		{ 1, 1, 1, 1, 1 }	{ 1, 1, 1, 1, 3 }	{ 1, 1, 5, 1, 3 }	{ 6, 1, 5, 1, 3 }	{ 6, 4, 5, 1, 3 }	{ 6, 4, 5, 2, 3 }
Parameter	FieldType	existiert	g	Aces	Aces	Aces	Aces	Aces	Aces	Aces
	diceValue	>= {1,1,1,1,1}	g	"{1,1,1,1,1}"	{ 1, 1, 1, 1, 1 }	{ 1, 1, 1, 1, 3 }	{ 1, 1, 5, 1, 3 }	{ 6, 1, 5, 1, 3 }	{ 6, 4, 5, 1, 3 }	{ 6, 4, 5, 2, 3 }
	Points	< 0	u	-1						
		>= 0	g	0	5	4	3	2	1	0
		> 5	u	6						
Endzustand	Points				5	4	3	2	1	0

2. Zweier

Die Methode "*calculateAndStorePoints*" muss beim ausgewählten Feld "Twos" gründlich getestet werden, damit nur die gleichen Würfelwerte, die die 2 haben, summiert werden. Somit wird ausgeschlossen, dass noch andere Würfelwerte mit dazugerechnet werden. Zum Schluss wird ein korrektes Ergebnis in das Feld eingetragen.

Methode: calculateAndStorePoints										
		Äquivalenzklassen		**Randwerte, krit. Werte**	**TF 1, gültig**	**TF 2, gültig**	**TF 3, gültig**	**TF 4, gültig**	**TF 5, gültig**	**TF 6, gültig**
Startzustand	FieldType		g	Twos	Twos	Twos	Twos	Twos	Twos	Twos
	diceValue		g		{ 2, 2, 2, 2, 2 }	{ 1, 2, 2, 2, 2 }	{ 2, 2, 5, 2, 3 }	{ 6, 2, 5, 2, 3 }	{ 6, 4, 5, 2, 3 }	{ 6, 4, 5, 1, 3 }
Parameter	FieldType	existiert	g	Twos	Twos	Twos	Twos	Twos	Twos	Twos
	diceValue	>= {1,1,1,1,1}	g	"{1,1,1,1,1}"	{ 2, 2, 2, 2, 2 }	{ 1, 2, 2, 2, 2 }	{ 2, 2, 5, 2, 3 }	{ 6, 2, 5, 2, 3 }	{ 6, 4, 5, 2, 3 }	{ 6, 4, 5, 1, 3 }
	Points	< 0	u	-1						
		>= 0	g	0	10	8	6	4	2	0
		> 10	u	11						
Endzustand	Points				10	8	6	4	2	0

3. Dreier

Die Methode "*calculateAndStorePoints*" muss beim ausgewählten Feld "Threes" gründlich getestet werden, damit nur die gleichen Würfelwerte, die die 3 haben, summiert werden. Somit wird ausgeschlossen, dass noch andere Würfelwerte mit dazugerechnet werden. Zum Schluss wird ein korrektes Ergebnis in das Feld eingetragen.

Methode: calculateAndStorePoints											
		Äquivalenzklassen		**Randwerte, krit. Werte**	**TF 1, gültig**	**TF 2, gültig**	**TF 3, gültig**	**TF 4, gültig**	**TF 5, gültig**	**TF 6, gültig**
Startzustand	FieldType		g	Threes	Threes	Threes	Threes	Threes	Threes	Threes
	diceValue		g		{ 3, 3, 3, 3, 3 }	{ 1, 3, 3, 3, 3 }	{ 3, 3, 5, 3, 2 }	{ 6, 3, 5, 3, 2 }	{ 6, 4, 5, 3, 2 }	{ 6, 4, 5, 1, 2 }
Parameter	FieldType	existiert	g	Threes	Threes	Threes	Threes	Threes	Threes	Threes
	diceValue	>= {1,1,1,1,1}	g	"{1,1,1,1,1}"	{ 3, 3, 3, 3, 3 }	{ 1, 3, 3, 3, 3 }	{ 3, 3, 5, 3, 2 }	{ 6, 3, 5, 3, 2 }	{ 6, 4, 5, 3, 2 }	{ 6, 4, 5, 1, 2 }
	Points	< 0	u	-1						
		>= 0	g	0	15	12	9	6	3	0
		> 15	u	16						
Endzustand	Points				15	12	9	6	3	0

4. Vierer

Die Methode "*calculateAndStorePoints*" muss beim ausgewählten Feld "Fours" gründlich getestet werden, damit nur die gleichen Würfelwerte, die die 4 haben, summiert werden. Somit wird ausgeschlossen, dass noch andere Würfelwerte mit dazugerechnet werden. Zum Schluss wird ein korrektes Ergebnis in das Feld .

Methode: calculateAndStorePoints										
		Äquivalenzklassen		**Randwerte, krit. Werte**	**TF 1, gültig**	**TF 2, gültig**	**TF 3, gültig**	**TF 4, gültig**	**TF 5, gültig**	**TF 6, gültig**
Startzustand	FieldType		g	Fours	Fours	Fours	Fours	Fours	Fours	Fours
	diceValue		g		{ 4, 4, 4, 4, 4 }	{ 4, 4, 4, 5, 4 }	{ 1, 4, 4, 5, 4 }	{ 1, 2, 4, 6, 4 }	{ 2, 3, 4, 5, 6 }	{ 1, 3, 2, 5, 6 }
Parameter	FieldType	existiert	g	Fours	Fours	Fours	Fours	Fours	Fours	Fours
	diceValue	>= {1,1,1,1,1}	g	"{1,1,1,1,1}"	{ 4, 4, 4, 4, 4 }	{ 4, 4, 4, 5, 4 }	{ 1, 4, 4, 5, 4 }	{ 1, 2, 4, 6, 4 }	{ 2, 3, 4, 5, 6 }	{ 1, 3, 2, 5, 6 }
	Points	< 0	u	-1						
		>= 0	g	0	20	16	12	8	4	0
		> 16	u	17						
Endzustand	Points				20	16	12	8	4	0

5. Fünfer

Die Methode "*calculateAndStorePoints*" muss beim ausgewählten Feld "Fives" gründlich getestet werden, damit nur die gleichen Würfelwerte, die die 5 haben, summiert werden. Somit wird ausgeschlossen, dass noch andere Würfelwerte mit dazugerechnet werden. Zum Schluss wird ein korrektes Ergebnis in das Feld eingetragen.

Methode: calculateAndStorePoints										
		Äquivalenzklassen		**Randwerte, krit. Werte**	**TF 1, gültig**	**TF 2, gültig**	**TF 3, gültig**	**TF 4, gültig**	**TF 5, gültig**	**TF 6, gültig**
Startzustand	FieldType		g	Fives	Fives	Fives	Fives	Fives	Fives	Fives
	diceValue		g		{ 5, 5, 5, 5, 5 }	{ 1, 5, 5, 5, 5 }	{ 1, 3, 5, 5, 5 }	{ 1, 2, 4, 5, 5 }	{ 2, 6, 1, 3, 5 }	{ 1, 4, 3, 6, 2 }
Parameter	FieldType	existiert	g	Fives	Fives	Fives	Fives	Fives	Fives	Fives
	diceValue	>= {1,1,1,1,1}	g	"{1,1,1,1,1}"	{ 5, 5, 5, 5, 5 }	{ 1, 5, 5, 5, 5 }	{ 1, 3, 5, 5, 5 }	{ 1, 2, 4, 5, 5 }	{ 2, 6, 1, 3, 5 }	{ 1, 4, 3, 6, 2 }
	Points	< 0	u	-1						
		>= 0	g	0	25	20	15	10	5	0
		> 25	u	26						
Endzustand	Points				25	20	15	10	5	0

6. Sechser

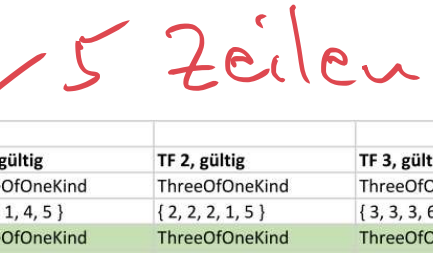
Die Methode "*calculateAndStorePoints*" muss beim ausgewählten Feld "Sixes" gründlich getestet werden, damit nur die gleichen Würfelwerte, die die 6 haben, summiert werden. Somit wird ausgeschlossen, dass noch andere Würfelwerte mit dazugerechnet werden. Zum Schluss wird ein korrektes Ergebnis in das Feld eingetragen.

Methode: calculateAndStorePoints										
		Äquivalenzklassen		**Randwerte, krit. Werte**	**TF 1, gültig**	**TF 2, gültig**	**TF 3, gültig**	**TF 4, gültig**	**TF 5, gültig**	**TF 6, gültig**
Startzustand	FieldType		g	Sixes	Sixes	Sixes	Sixes	Sixes	Sixes	Sixes
	diceValue		g		{ 6, 6, 6, 6, 6 }	{ 1, 6, 6, 6, 6 }	{ 3, 5, 6, 6, 6 }	{ 1, 2, 6, 5, 6 }	{ 1, 2, 4, 6, 5 }	{ 1, 2, 3, 5, 4 }
Parameter	FieldType	existiert	g	Sixes	Sixes	Sixes	Sixes	Sixes	Sixes	Sixes
	diceValue	>= {1,1,1,1,1}	g	"{1,1,1,1,1}"	{ 6, 6, 6, 6, 6 }	{ 1, 6, 6, 6, 6 }	{ 3, 5, 6, 6, 6 }	{ 1, 2, 6, 5, 6 }	{ 1, 2, 4, 6, 5 }	{ 1, 2, 3, 5, 4 }
	Points	< 0	u	-1						
		>= 0	g	0	30	24	18	12	6	0
		> 30	u	31						
Endzustand	Points				30	24	18	12	6	0

7. Dreierpasch

Die Methode "*calculateAndStorePoints*" muss beim ausgewählten Feld "ThreeOfOneKind" gründlich getestet werden, um zu prüfen, ob die Bedingung für einen Dreierpasch erfüllt ist, um alle Würfelwerte zu summieren. Zum Schluss wird ein korrektes Ergebnis in das Feld eingetragen. Wenn die Bedingung nicht erfüllt ist, wird eine 0 ins Feld eingetragen.

(Der übersichtshalber haben wir hier die Testfälle untereinander aufgeführt)



Methode: calculateAndStorePoints									
		Äquivalenzklassen		**Randwerte, krit. Werte**	**TF 1, gültig**	**TF 2, gültig**	**TF 3, gültig**	**TF 4, gültig**	**TF 5, gültig**
Startzustand	FieldType		g	ThreeOfOneKind	ThreeOfOneKind	ThreeOfOneKind	ThreeOfOneKind	ThreeOfOneKind	ThreeOfOneKind
	diceValue		g		{ 1, 1, 1, 4, 5 }	{ 2, 2, 2, 1, 5 }	{ 3, 3, 3, 6, 2 }	{ 1, 4, 4, 4, 5 }	{ 1, 5, 2, 5, 5 }
Parameter	FieldType	existiert	g	ThreeOfOneKind	ThreeOfOneKind	ThreeOfOneKind	ThreeOfOneKind	ThreeOfOneKind	ThreeOfOneKind
	diceValue	>= {1,1,1,1,1}	g	"{1,1,1,1,1}"	{ 1, 1, 1, 4, 5 }	{ 2, 2, 2, 1, 5 }	{ 3, 3, 3, 6, 2 }	{ 1, 4, 4, 4, 5 }	{ 1, 5, 2, 5, 5 }
	Points	< 0	u	-1					
		>= 0	g	0	12	12	17	18	18
		> 30	u	31					
Endzustand	Points				12	12	17	18	18
		Äquivalenzklassen		**Randwerte, krit. Werte**	**TF 6, gültig**	**TF 7, gültig**	**TF 8, gültig**	**TF 9, gültig**	**TF 10, gültig**
			g	ThreeOfOneKind	ThreeOfOneKind	ThreeOfOneKind	ThreeOfOneKind	ThreeOfOneKind	ThreeOfOneKind
			g		{ 6, 1, 6, 3, 6 }	{ 1, 1, 1, 1, 5 }	{ 1, 1, 1, 1, 1 }	{ 1, 1, 3, 4, 5 }	{ 1, 2, 3, 4, 5 }
		existiert	g	ThreeOfOneKind	ThreeOfOneKind	ThreeOfOneKind	ThreeOfOneKind	ThreeOfOneKind	ThreeOfOneKind
		>= {1,1,1,1,1}	g	"{1,1,1,1,1}"	{ 6, 1, 6, 3, 6 }	{ 1, 1, 1, 1, 5 }	{ 1, 1, 1, 1, 1 }	{ 1, 1, 3, 4, 5 }	{ 1, 2, 3, 4, 5 }
		< 0	u	-1					
		>= 0	g	0	22	9	5	0	0
		> 30	u	31					
Endzustand	Points				22	9	5	0	0

8. Viererpasch

Die Methode "*calculateAndStorePoints*" muss beim ausgewählten Feld "FourOfOneKind" gründlich getestet werden, um zu prüfen, ob die Bedingung für einen Viererpasch erfüllt ist, um alle Würfelwerte zu summieren. Zum Schluss wird ein korrektes Ergebnis in das Feld eingetragen. Wenn die Bedingung nicht erfüllt ist, wird eine 0 ins Feld eingetragen.

(Der übersichtshalber haben wir hier die Testfälle untereinander aufgeführt)

Methode: calculateAndStorePoints									
		Äquivalenzklassen		**Randwerte, krit. Werte**	**TF 1, gültig**	**TF 2, gültig**	**TF 3, gültig**	**TF 4, gültig**	**TF 5, gültig**
Startzustand	FieldType		g	FourOfOneKind	FourOfOneKind	FourOfOneKind	FourOfOneKind	FourOfOneKind	FourOfOneKind
	diceValue		g		{ 1, 1, 1, 1, 5 }	{ 2, 2, 2, 2, 1 }	{ 3, 3, 3, 3, 2 }	{ 4, 4, 4, 4, 5 }	{ 1, 5, 5, 5, 5 }
Parameter	FieldType	existiert	g	FourOfOneKind	FourOfOneKind	FourOfOneKind	FourOfOneKind	FourOfOneKind	FourOfOneKind
	diceValue	>= {1,1,1,1,1}	g	"{1,1,1,1,1}"	{ 1, 1, 1, 1, 5 }	{ 2, 2, 2, 2, 1 }	{ 3, 3, 3, 3, 2 }	{ 4, 4, 4, 4, 5 }	{ 1, 5, 5, 5, 5 }
	Points	< 0	u	-1					
		>= 0	g	0	9	9	14	21	21
		> 30	u	31					
Endzustand	Points				9	9	14	21	21
		Äquivalenzklassen		**Randwerte, krit. Werte**	**TF 6, gültig**	**TF 7, gültig**	**TF 8, gültig**	**TF 9, gültig**	
			g	FourOfOneKind	FourOfOneKind	FourOfOneKind	FourOfOneKind	FourOfOneKind	
			g		{ 6, 6, 6, 6, 2 }	{ 1, 1, 1, 4, 5 }	{ 1, 1, 3, 4, 5 }	{ 1, 2, 3, 4, 5 }	
		existiert	g	FourOfOneKind	FourOfOneKind	FourOfOneKind	FourOfOneKind	FourOfOneKind	
		>= {1,1,1,1,1}	g	"{1,1,1,1,1}"	{ 6, 6, 6, 6, 2 }	{ 1, 1, 1, 4, 5 }	{ 1, 1, 3, 4, 5 }	{ 1, 2, 3, 4, 5 }	
		< 0	u	-1					
		>= 0	g	0	26	0	0	0	
		> 30	u	31					
Endzustand	Points				26	0	0	0	

9. Full-House

Die Methode "*calculateAndStorePoints*" muss beim ausgewählten Feld "FullHouse" gründlich getestet werden, um zu prüfen, ob die Bedingung für einen FullHouse erfüllt ist, um alle Würfelwerte zu summieren. Zum Schluss wird ein korrektes Ergebnis in das Feld eingetragen. Wenn die Bedingung nicht erfüllt ist, wird eine 0 ins Feld eingetragen.

Methode: calculateAndStorePoints										
		Äquivalenzklassen		**Randwerte, krit. Werte**	**TF 1, gültig**	**TF 2, gültig**	**TF 3, gültig**	**TF 4, gültig**	**TF 5, gültig**	**TF 6, gültig**
Startzustand	FieldType		g	FullHouse	FullHouse	FullHouse	FullHouse	FullHouse	FullHouse	FullHouse
	diceValue		g		{ 1, 1, 1, 2, 2 }	{ 2, 2, 3, 3, 3 }	{ 1, 1, 2, 3, 3 }	{ 1, 1, 1, 2, 3 }	{ 1, 1, 1, 1, 3 }	{ 1, 1, 1, 1, 1 }
Parameter	FieldType	existiert	g	FullHouse	FullHouse	FullHouse	FullHouse	FullHouse	FullHouse	FullHouse
	diceValue	>= {1,1,1,1,1}	g	"{1,1,1,1,1}"	{ 1, 1, 1, 2, 2 }	{ 2, 2, 3, 3, 3 }	{ 1, 1, 2, 3, 3 }	{ 1, 1, 1, 2, 3 }	{ 1, 1, 1, 1, 3 }	{ 1, 1, 1, 1, 1 }
	Points	< 0	u	-1						
		>= 0	g	0	25	25	0	0	0	0
		> 25	u	26						
Endzustand	Points				25	25	0	0	0	0

10. Kleine Straße

Die Methode "*calculateAndStorePoints*" muss beim ausgewählten Feld "Small Straight" gründlich getestet werden, um zu prüfen, ob die Bedingung für einen SmallStraight erfüllt ist, um alle Würfelwerte zu summieren. Zum Schluss wird ein korrektes Ergebnis in das Feld eingetragen. Wenn die Bedingung nicht erfüllt ist, wird eine 0 ins Feld eingetragen.

(Der übersichtshalber haben wir hier die Testfälle untereinander aufgeführt)

Methode: calculateAndStorePoints									
		Äquivalenzklassen		**Randwerte, krit. Werte**	**TF 1, gültig**	**TF 2, gültig**	**TF 3, gültig**	**TF 4, gültig**	**TF 5, gültig**
Startzustand	FieldType		g	Small Straight	Small Straight	Small Straight	Small Straight	Small Straight	Small Straight
	diceValue		g		{ 1, 2, 3, 4, 1 }	{ 2, 3, 4, 5, 1 }	{ 3, 4, 5, 6, 2 }	{ 3, 2, 4, 4, 1 }	{ 5, 6, 3, 4, 1 }
Parameter	FieldType	existiert	g	Small Straight	Small Straight	Small Straight	Small Straight	Small Straight	Small Straight
	diceValue	>= {1,1,1,1,1}	g	"{1,1,1,1,1}"	{ 1, 2, 3, 4, 1 }	{ 2, 3, 4, 5, 1 }	{ 3, 4, 5, 6, 2 }	{ 3, 2, 4, 4, 1 }	{ 5, 6, 3, 4, 1 }
	Points	< 0	u	-1					
		>= 0	g	0	30	30	30	30	30
		> 30	u	31					
Endzustand	Points				30	30	30	30	30
		Äquivalenzklassen		**Randwerte, krit. Werte**	**TF 6, gültig**	**TF 7, gültig**	**TF 8, gültig**	**TF 9, gültig**	**TF 10, gültig**
			g	Small Straight	Small Straight	Small Straight	Small Straight	Small Straight	Small Straight
			g		{ 3, 2, 3, 4, 5 }	{ 1, 1, 3, 4, 1 }	{ 1, 1, 1, 4, 1 }	{ 1, 2, 3, 5, 6 }	{ 1, 4, 4, 2, 2 }
		existiert	g	Small Straight	Small Straight	Small Straight	Small Straight	Small Straight	Small Straight
		>= {1,1,1,1,1}	g	"{1,1,1,1,1}"	{ 3, 2, 3, 4, 5 }	{ 1, 1, 3, 4, 1 }	{ 1, 1, 1, 4, 1 }	{ 1, 2, 3, 5, 6 }	{ 1, 4, 4, 2, 2 }
		< 0	u	-1					
		>= 0	g	0	30	0	0	0	0
		> 30	u	31					
Endzustand	Points				30	0	0	0	0

11. Große Straße

Die Methode "*calculateAndStorePoints*" muss beim ausgewählten Feld "Large Straight" gründlich getestet werden, um zu prüfen, ob die Bedingung für einen LargeStraight erfüllt ist, um alle Würfelwerte zu summieren. Zum Schluss wird ein korrektes Ergebnis in das Feld eingetragen. Wenn die Bedingung nicht erfüllt ist, wird eine 0 ins Feld eingetragen.

(Der übersichtshalber haben wir hier die Testfälle untereinander aufgeführt)

Methode: calculateAndStorePoints										
		Äquivalenzklassen		**Randwerte, krit. Werte**	**TF 1, gültig**	**TF 2, gültig**	**TF 3, gültig**	**TF 4, gültig**	**TF 5, gültig**	
Startzustand	FieldType		g	Large Straight	Large Straight	Large Straight	Large Straight	Large Straight	Large Straight	
	diceValue		g		{ 1, 2, 3, 4, 5 }	{ 2, 3, 4, 5, 6 }	{ 1, 4, 2, 3, 5 }	{ 3, 4, 2, 5, 6 }	{2, 3, 4, 1, 5 }	
Parameter	FieldType	existiert	g	Large Straight	Large Straight	Large Straight	Large Straight	Large Straight	Large Straight	
	diceValue	>= {1,1,1,1,1}	g	"{1,1,1,1,1}"	{ 1, 2, 3, 4, 5 }	{ 2, 3, 4, 5, 6 }	{ 1, 4, 2, 3, 5 }	{ 3, 4, 2, 5, 6 }	{2, 3, 4, 1, 5 }	
	Points	< 0	u	-1						
		>= 0	g	0	40	40	40	40	40	
		> 40	u	41						
Endzustand	Points				40	40	40	40	40	
		Äquivalenzklassen		**Randwerte, krit. Werte**	**TF 6, gültig**	**TF 7, gültig**	**TF 8, gültig**	**TF 9, gültig**	**TF 10, gültig**	**TF 11, gültig**
			g	Large Straight	Large Straight	Large Straight	Large Straight	Large Straight	Large Straight	Large Straight
			g		{ 1, 2, 3, 1, 1 }	{ 1, 2, 4, 2, 1 }	{ 2, 3, 4, 3, 3 }	{ 2, 3, 2, 2, 2 }	{ 1, 6, 1, 6, 1 }	{ 1, 1, 1, 1, 1 }
		existiert	g	Large Straight	Large Straight	Large Straight	Large Straight	Large Straight	Large Straight	Large Straight
		>= {1,1,1,1,1}	g	"{1,1,1,1,1}"	{ 1, 2, 3, 1, 1 }	{ 1, 2, 4, 2, 1 }	{ 2, 3, 4, 3, 3 }	{ 2, 3, 2, 2, 2 }	{ 1, 6, 1, 6, 1 }	{ 1, 1, 1, 1, 1 }
		< 0	u	-1						
		>= 0	g	0	0	0	0	0	0	0
		> 40	u	41						
Endzustand	Points				0	0	0	0	0	0

12. Kniffel

Die Methode "*calculateAndStorePoints*" muss beim ausgewählten Feld "Kniffel" gründlich getestet werden, um zu prüfen, ob die Bedingung für einen Kniffel erfüllt ist, um alle Würfelwerte zu summieren. Zum Schluss wird ein korrektes Ergebnis in das Feld eingetragen. Wenn die Bedingung nicht erfüllt ist, wird eine 0 ins Feld eingetragen.

(Der übersichtshalber haben wir hier die Testfälle untereinander aufgeführt)

Methode: calculateAndStorePoints											
		Äquivalenzklassen		**Randwerte, krit. Werte**	**TF 1, gültig**	**TF 2, gültig**	**TF 3, gültig**	**TF 4, gültig**	**TF 5, gültig**	
Startzustand	FieldType		g	Kniffel	Kniffel	Kniffel	Kniffel	Kniffel	Kniffel	
	diceValue		g		{ 1, 1, 1, 1, 1 }	{ 2, 2, 2, 2, 2 }	{ 3, 3, 3, 3, 3 }	{ 4, 4, 4, 4, 4 }	{ 5, 5, 5, 5, 5 }	
Parameter	FieldType	existiert	g	Kniffel	Kniffel	Kniffel	Kniffel	Kniffel	Kniffel	
	diceValue	>= {1,1,1,1,1}	g	"{1,1,1,1,1}"	{ 1, 1, 1, 1, 1 }	{ 2, 2, 2, 2, 2 }	{ 3, 3, 3, 3, 3 }	{ 4, 4, 4, 4, 4 }	{ 5, 5, 5, 5, 5 }	
		> {6,6,6,6,6}	u	"{7,7,7,7,7}"						
	Points	< 0	u	-1						
		>= 0	g	0	50	50	50	50	50	
		> 50	u	51						
Endzustand	Points				50	50	50	50	50	
		Äquivalenzklassen		**Randwerte, krit. Werte**	**TF 6, gültig**	**TF 7, gültig**	**TF 8, gültig**	**TF 9, gültig**	**TF 10, gültig**	**TF 11, gültig**
			g	Kniffel	Kniffel	Kniffel	Kniffel	Kniffel	Kniffel	Kniffel
			g		{ 6, 6, 6, 6, 6 }	{ 1, 1, 1, 1, 4 }	{ 1, 1, 1, 2, 4 }	{ 1, 1, 3, 2, 4 }	{ 1, 5, 3, 2, 4 }	{ 1, 2, 3, 5, 6 }
		existiert	g	Kniffel	Kniffel	Kniffel	Kniffel	Kniffel	Kniffel	Kniffel
		>= {1,1,1,1,1}	g	"{1,1,1,1,1}"	{ 6, 6, 6, 6, 6 }	{ 1, 1, 1, 1, 4 }	{ 1, 1, 1, 2, 4 }	{ 1, 1, 3, 2, 4 }	{ 1, 5, 3, 2, 4 }	{ 1, 2, 3, 5, 6 }
		> {6,6,6,6,6}	u	"{7,7,7,7,7}"						
		< 0	u	-1						
		>= 0	g	0	50	0	0	0	0	0
		> 50	u	51						
Endzustand	Points				50	0	0	0		0

13. Chance

Die Methode "*calculateAndStorePoints*" muss beim ausgewählten Feld "Chance" gründlich getestet werden, um zu prüfen, ob die Bedingung für einen Chance erfüllt ist, um alle Würfelwerte zu summieren. Zum Schluss wird ein korrektes Ergebnis in das Feld eingetragen.

(Der übersichtshalber haben wir hier die Testfälle untereinander aufgeführt)

Methode: calculateAndStorePoints								
		Äquivalenzklassen		**Randwerte, krit. Werte**	**TF 1, gültig**	**TF 2, gültig**	**TF 3, gültig**	**TF 4, gültig**
Startzustand	FieldType		g	Chance	Chance	Chance	Chance	Chance
	diceValue		g		{ 1, 2, 3, 4, 1 }	{ 1, 2, 3, 4, 6 }	{ 5, 6, 3, 4, 1 }	{ 1, 2, 3, 4, 5 }
Parameter	FieldType	existiert	g	Chance	Chance	Chance	Chance	Chance
	diceValue	>= {1,1,1,1,1}	g	"{1,1,1,1,1}"	{ 1, 2, 3, 4, 1 }	{ 1, 2, 3, 4, 6 }	{ 5, 6, 3, 4, 1 }	{ 1, 2, 3, 4, 5 }
	Points	< 5	u	4				
		>= 5	g	5	11	16	19	15
		> 30	u	31				
Endzustand	Points				11	16	19	15
		Äquivalenzklassen		**Randwerte, krit. Werte**	**TF 5, gültig**	**TF 6, gültig**	**TF 7, gültig**	
			g	Chance	Chance	Chance	Chance	
			g		{ 6, 6, 5, 5, 5 }	{ 1, 2, 3, 3, 3 }	{ 1, 2, 2, 4, 1 }	
		existiert	g	Chance	Chance	Chance	Chance	
		>= {1,1,1,1,1}	g	"{1,1,1,1,1}"	{ 6, 6, 5, 5, 5 }	{ 1, 2, 3, 3, 3 }	{ 1, 2, 2, 4, 1 }	
		< 5	u	4				
		>= 5	g	5	27	12	10	
		> 30	u	31				
Endzustand	Points				27	12	10	

b) Summen- und Bonusberechnung testen

Die Methode "*calculateScoreSums*" in der Klasse ScoreCard muss gründlich getestet werden. Dazu nutzen wir in den Junit-Tests die Methode "testCalculateScoreSums(int[] points)", wo wir die Möglichkeit haben, Punkte manuell als Parameter mit zuliefern.

Methode: testCalculateScoreSums(int[] points)											
			JUnit-Methode:		*testNoPoints()*	*testLowerNoBonus()*	*testUpperAndLowerNoBonus()*	*testUpperAndLowerWithBonus()*	*testUpperNoBonus()*	*testUpperWithBonus()*
Parameter:			**Äquivalenzklassen**		**TF 1, gültig**	**TF 2, gültig**	**TF 3, gültig**	**TF 4, gültig**	**TF 5, gültig**	**TF 6, gültig**
int[] Punkte					{0,0,0,0,0,0, 0,0,0,0,0,0,0}	{0,0,0,0,0,0, 5,5,25,0,0,0,15}	{1,8,3,20,0,30, 5,5,25,0,0,0,15}	{1,10,3,20,0,30, 5,5,25,0,0,0,15}	{2,10,0,20,0,30, 0,0,0,0,0,0,0}	{1,10,3,20,0,30, 0,0,0,0,0,0,0}
Endzustand		bonus	0 Punkte	g	0	0	0		0	
			35 Punkte	g				35		35
		sumUpper	0 Punkte	g	0	0				
			> 0 Punkte	g			62	99	62	99
		sumLower	0 Punkte	g	0				0	0
			> 0 Punkte	g		50	50	50		
		sumOverall	0 Punkte	g	0					
			> 0 Punkte	g		50	112	149	99	99

c) Würfel testen

Wir benutzen für das Spiel einen Würfel mit 6 Seiten. Um sicherzugehen, dass keine Zahl kleiner als 1 oder größer als 6 gewürfelt werden kann, haben wir eine Testmethode geschrieben, die 1000 Mal würfelt und prüft, ob gültig oder ungültig gewürfelt wurde. Da Zufallsgeneratoren nur schwierig getestet werden können, sagt das Testergebnis streng genommen nur etwas über die statistische Wahrscheinlichkeit, außerhalb von dem gültigen Bereich zu würfeln, aus.

Methode: roll()				**Randwerte, krit. Werte**	TF1, gültig	TF2, gültig	TF3, gültig	TF4, gültig	TF5, gültig	TF6, gültig	TF7, ungültig	TF8, ungültig
		Äquivalenzklassen										
		>= 1 und <=6	g	ist 6-seitiger Würfel	1	2	3	4	5	6		
		< 1	u	ist kein 6-seitiger Würfel							0	
		> 6	u	ist kein 6-seitiger Würfel								7