Hausaufgabe 11

Tim Wende

December 20, 2021

Äquivalenzklassen Abschranzbudenbesuch

a. Bestimmen Sie die Äquivalenzklassen zur Berechnung folgender Rabatt-Funktion. Beachten Sie dabei auch Grenzwerte, und Sonderfälle. Geben Sie tabellarisch die Eingaben der drei Parameter, das Ergebnis und eine kurze Beschreibung der Äquivalenzklasse an. Gliedern Sie Ihre Äquivalenzklassen wie folgt:

- Regulären Situationen
- Fehlerfälle
- Grenzfälle
- Sonderfälle

Extremwerte können Sie hier außer Acht lassen.

Grenzfälle können sich auf reguläre Situationen und Fehlerfälle beziehen. Hier darf es Überschneidungen (oder Dopplungen) geben. Bitte geben Sie diese auch unbedingt in den verschiedenen Gruppen an.

Äquivalenzklassen

ÄK	Äquivalenzklasse	Typ	Repräsentant		
			geschlecht	alter	istBeamter
0	regulär: geschlecht $\in \{m, w, d\}$ $\wedge$ $18 \leq$ alter ≤ 79	Zulässig	m, w, d	18, 20, 79	true, false
1	fehler: geschlecht == null $\vee$ $17 \geq$ alter $\vee$ alter ≥ 80	Unzulässig	null, egal	$[-\infty..\infty]$	egal

Unterklassen

$$\forall \mathrm{AK}_j : \mathrm{UK}_{j_i} \subseteq \mathrm{AK}_j \quad \wedge \quad \bigcup_{u \in \mathrm{UK}_j} u = \mathrm{AK_j} \quad \wedge \quad \bigcup_{a \in \mathrm{AK}} a = \Omega$$

Regulärfälle (`@return double` $\in \{0.0, 0.05, 0.07, 0.1, 0.12\}$)

ÄK	RF	Unterklasse	Typ	Repräsentant		
				geschlecht	alter	istBeamter
0	0	kein Rabatt	(Zulässig, 0.0)	m, d	[18..79], 20	false
0	1	5% Rabatt	(Zulässig, 0.05)	m, d	[18..79], 20	true
0	1	5% Rabatt	(Zulässig, 0.05)	w	[30..79], 79	false
0	2	7% Rabatt	(Zulässig, 0.07)	w	[18..29], 20	false
0	3	10% Rabatt	(Zulässig, 0.1)	w	[30..79], 79	true
0	4	12% Rabatt	(Zulässig, 0.12)	w	[18..29], 20	true

Fehlerfälle (`@throws IllegalArgumentException()`)

ÄK	FF	Unterklasse	Typ	Repräsentant		
				geschlecht	alter	istBeamter
1	0	alter ≤ 17	Unzulässig	egal	-7, 0, 7, 17	egal
1	1	alter ≥ 80	Unzulässig	egal	80, INT_MAX	egal
1	2	geschlecht == null	Unzulässig	null	egal	egal

Grenzfälle

Da es sich bei `alter` als einziges um eine geordnete sortierbare Menge an Werten handelt, folgt nun die Grenzwertbetrachtung dieses Attributs. Wenn man eine unsortierte Liste von 3, respektive 2, Elementen vorliegen hat, ist eine GW-Betrachtung sinnlos bis hin zu nicht möglich (bzgl. `geschlecht`). Selbiges gilt bei `istBeamter`; Hier kann man sich jedoch streiten, ob es sich um zwei Zeichenketten ("true", "false") oder um Zahlwerte (0, 1) handelt. Im zweiten Fall wäre eine GW-Betrachtung möglich, jedoch auf solch einem Intervall nicht aussagekräftig, da jeder vorhandene Wert ein GW wäre.

ÄK	FF	Typ	Randwerte		
			geschlecht	alter	istBeamter
1	0_o	Unzulässig	egal	17	egal
0_u		Zulässig	egal	18	egal
0_o		Zulässig	egal	79	egal
1	1_u	Unzulässig	egal	80	egal

Sonderfälle

Da wir in unserem Beispiel keine wirklichen Sonderfälle haben [1] , habe ich die Randwerte der Ausgabe gewählt. Argumentativ könnte man dies als Abfrage begründen, da man als Disco-Besitzender zu hohe Rabatte verhindern möchte, bzw. die Menge an KundInnen erfahren will, welche diesen Höchstsatz an Rabatten bekommen. Hier könnte man ebenfalls einen festen Grenzwert wählen; hier wäre beispielsweise 10% mMn. ebenfalls möglich. Zusätzlich möchte man keine KundInnen frustrieren, indem man ihnen keine/ zu geringe Rabatte zuordnet.

ÄK	RF	Typ	Sonderfall		
			geschlecht	alter	istBeamter
0	0	(Zulässig, 0.0)	m, d	[18..79], 20	false
0	4	(Zulässig, 0.12)	w	[18..29], 20	true

b. Erklären Sie allgemein was Extremwerte sind und geben Sie ein Beispiel dazu.

Vorab möchte ich gerne die umfangreichen Informationen aus der Vorlesung darstellen:

- Minimal zulässige Zahlenwerte
- Maximal zulässige Zahlenwerte (z.Bsp.: INT_MAX)

So gehören zu Extremwerten im generellen Fälle:

a. mit einem normalen Ausgang und Extremeingangswerten
(Extreme im Sinne von zu hoch oder zu niedrig, zu kurz oder zu lang, zu viel oder zu wenig)

b. mit einem abnormalen Ausgang, resultierend aus falschen/ ungültigen/ ungünstigen Eingabewerten

Die der einzige Parameter, welcher ein wenig künstlerische Freiheit ermöglicht `alter` ist, welcher zusätzlich in seinem Gültigkeitsintervall ([18..79]) eingeschränkt ist, hat unser Beispiel keine Extremeingangswerte. Zusätzlich kennen wir nicht die genaue Implementierung[2] der Methode (was ja auch Sinn dieses Black-Box-Tests ist), kann ich keine abnormalen Ausgänge erraten.

Um jedoch ein gutes, von unserer Miniwelt losgelöstes Beispiel zu geben, habe ich mich auf Methoden mit Strings konzentriert. Hier wären Extremeingangswerte der leere String: „“, extrem lange Strings (z.B. aus eingelesenen Dateien). Bei Zahlwerten ist dies noch einfacher. Hier wären Extremeingangswerte: `Integer.MAX_VALUE`, `-Integer.MAX_VALUE` oder `0.0000000001`.

Ein ausgearbeitet detailliertes Beispiel wäre demnach:

```
(new Scanner(new File("pi.txt"))).readLine().subString(Integer.MAX_VALUE);
```

[1] Nach meinem Verständnis sind Sonderfälle stark verwandt mit Extremfällen. (Böse Zungen behaupten sogar, diese seien Synonyme) Als Sonderfall verstehe ich Äquivalenzklassen, welche in ihrer Häufigkeit des Auftretens oder besonderheit der Rückgabe auffallen. So würde man besonders seltene oder häufig auftretende Klassen hier besonders erwähnen. Da dies jedoch nicht der Fall ist, da die Parameter ungefähr gleichverteillt aufteten und das daraus resultierende Ergebnis meist in einem sehr überschaubaren Rahmen liegt, versuche ich mich an dieser Betrachtung unter erschwerten Umständen. Man könnte einen Zusammenhang aus Höhe des Rabattes und größe der Menschengruppe herstellen, die diesen Rabatt erhält, jedoch ist mir hier die Datenlage zu ungenau bzw. zu sporadisch. Sollte dies jedoch das erwünschte Ergebnis sein, unterstelle ich dem Disco-Betreibenden, dass er $\approx 50\%$ aller KundInnen keinen Rabatt und nur einer minimalen Menge an Kunden den Höchstsatz von 12% Rabatt ermöglicht. Wer hätte dies nur erwartet ...

[2] Anbei mal meine Version dieser Methode, zum errechnen der Rückgabewerte von Äquivalenzklassen

Hier die gegebene Javadoc:

```java
/**
* Berechnet den Rabatt auf den Eintrittspreis eines Diskobesuchs
* Frauen erhalten 5% Rabatt. Sind diese zusätzlich unter 30, werden zusätzliche 2% Rabatt gewährt.
* Beamte erhalten nochmals 5% Rabatt (unabh. vom Geschlecht).
*
* @param geschlecht Wert eines enums mit den Ausprägungen "männlich" und "weiblich" (m/w)
* @param alter      Das Alter (17 < alter < 80) des Diskobesuchers,
                   ansonsten wird eine Exception geworfen
* @param istBeamter Flag, um anzuzeigen, dass der Diskobesucher ein Beamter ist
* @return           Rabattfaktor (1.0 entsprechen 100% = freier Eintritt,
                   0.5 entsprechen 50% = halber Eintrittspreis)
* @throws           Exception in allen weiteren nicht hier spezifizierten Fällen
*/
public double berechneRabatt(GESCHLECHT geschlecht, int alter, boolean istBeamter){}
```

Hier der dazugehörende Output:

```
(männlich, 20) ist kein Beamter; Rabatt: 0.0
(männlich, 20) ist Beamter;      Rabatt: 0.05
(weiblich, 20) ist Beamter;      Rabatt: 0.12
(weiblich, 20) ist kein Beamter; Rabatt: 0.07
(weiblich, 79) ist Beamter;      Rabatt: 0.1
(weiblich, 79) ist kein Beamter; Rabatt: 0.05
(männlich, 79) ist Beamter;      Rabatt: 0.05
(männlich, 79) ist kein Beamter; Rabatt: 0.0
```

Und hier meine (dank Black-Box-Test bis zum Schluss verschlossene) Implementierung:

```
public class Diskobesuch {
	private static double berechneRabatt(GESCHLECHT geschlecht, int alter, boolean istVerbeamteter) {
		if (alter <= 17)
			throw new IllegalArgumentException("FF 0: alter muss >= 18 sein!");
		if (alter >= 80)
			throw new IllegalArgumentException("FF 1: alter muss <= 79 sein!");

		if (geschlecht == null)
			throw new IllegalArgumentException("F 2: geschlecht darf nicht null sein");

		System.out.print("(" + geschlecht + ", " + alter + ") ist "
			+ ((istVerbeamteter) ? "" : "kein ") + "Beamter; "
			+ ((istVerbeamteter) ? " ".repeat(5) : "") + "Rabatt: ");
		return (((geschlecht == GESCHLECHT.weiblich) ? 5 + ((alter < 30) ? 2 : 0) : 0)
			+ ((istVerbeamteter) ? 5 : 0)) / 100.;
	}

	public static void main(String[] args) {
		// AK 0
		System.out.println(berechneRabatt(GESCHLECHT.maennlich, 20, false)); // UK 0
		System.out.println(berechneRabatt(GESCHLECHT.maennlich, 20, true)); // UK 1
		System.out.println(berechneRabatt(GESCHLECHT.weiblich, 20, true)); // UK 4
		System.out.println(berechneRabatt(GESCHLECHT.weiblich, 20, false)); // UK 2
		System.out.println(berechneRabatt(GESCHLECHT.weiblich, 79, true)); // UK 3
		System.out.println(berechneRabatt(GESCHLECHT.weiblich, 79, false)); // UK 1
		System.out.println(berechneRabatt(GESCHLECHT.maennlich, 79, true)); // UK 1
		System.out.println(berechneRabatt(GESCHLECHT.maennlich, 79, false)); // UK 0

		// AK 1; FF 0
//		  System.out.println(berechneRabatt(GESCHLECHT.maennlich, -7, false));
//		  System.out.println(berechneRabatt(GESCHLECHT.maennlich, 17, false));

		// AK 1; FF 1
//		  System.out.println(berechneRabatt(GESCHLECHT.maennlich, 80, false));
//		  System.out.println(berechneRabatt(GESCHLECHT.maennlich, Integer.MAX_VALUE, false));

		// AK 1; FF 2
//		  System.out.println(berechneRabatt(null, 20, false));
	}

	private enum GESCHLECHT {
		maennlich, weiblich, divers
	}
}
```