Praktikum Programmieren
Studiengang Angewandte Informatik
Prof. Dr. Bernd Kahlbrandt
Dipl.-Ing. Gerhard Oelker
Hochschule für Angewandte Wissenschaften Hamburg
Department für Informatik
9. Oktober 2019

Präsenzaufgabe A00 B-AI1 PTP WS 2019/20

Die folgende Aufgabe werden wir in den Praktikumsterminen der ersten beiden Wochen als Präsenzaufgabe behandeln. Als Beispiel für ein Praktikumsprotokoll werde ich dazu einen Lösungsvorschlag veröffentlichen.

Aufgabe A00

1. Schreiben Sie bitte eine Klasse, deren Objekte Würfel in einem Spiel repräsentieren! Ein ungezinkter Würfel liefert mit gleicher Wahrscheinlichkeit die Augenzahlen 1 bis 6.

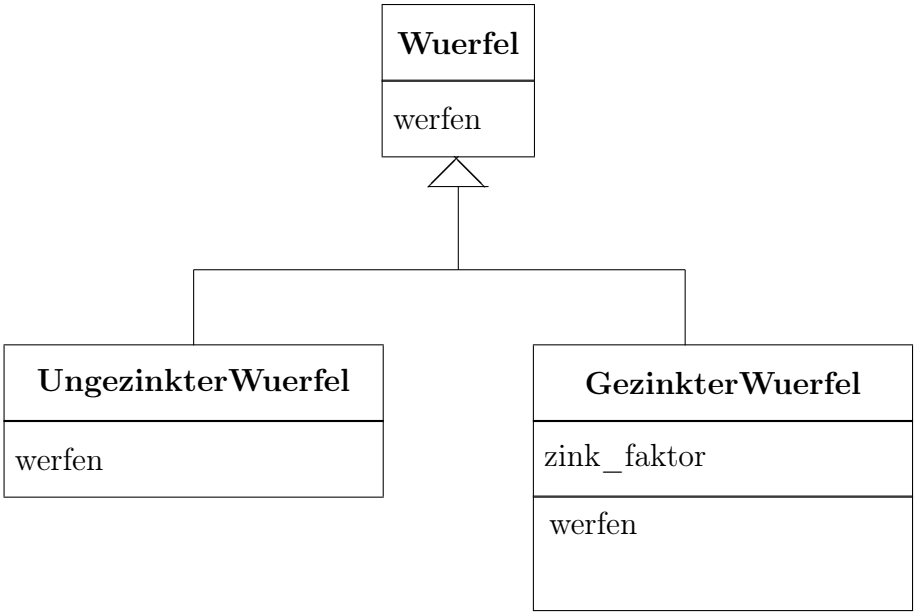
2. Ein gezinkter Würfel liefert mit einer vorgegebenen Wahrscheinlichkeit $0 \leq p < 1$ die Augenzahl 6 (Sinnvoll ist nur $\frac{1}{6} < p \leq 1$, aber das interessiert hier (noch) nicht). Ergänzen oder verändern Sie Ihren Entwurf, so dass auch gezinkte Würfel erzeugt werden können und illustrieren Sie die Nutzung wie oben. (**Tipp:** Wenn die 6 mit der Wahrscheinlichkeit p geworfen wird, bleibt für die anderen Zahlen eine Wahrscheinlichkeit von $1 - p$.)

3. Demonstrieren Sie bitte die Nutzung dieser Klassen durch Code, der z. B. $10, 100, \ldots$ „Würfe“ mit „Ihren“ Würfeln ausführt und die Häufigkeiten der Zahlen für einen ungezinkten und einige gezinkte Würfel ausgibt!

Beispiellösungsvorschlag A00

Ich habe einen sehr ausführlichen Lösungsvorschlag geschrieben. Von Ihnen erwarte ich nur die Diskussion einer Lösungsalternative. Auch der Teil über die „Würfelergebnisse“ ist nur ein Service für Sie, damit Sie sehen, wie ich mir die Ausgabe vorgestellt habe

Ein Würfel kann gezinkt sein oder nicht. An folgendem Klassendiagramm kann man die möglichen Implementierungen und ihre Vor- und Nachteile durchspielen.



Wuerfel und *UngezinkterWuerfel* unterscheiden sich nicht. Von daher kann eine der Klassen entfallen. Das führt auf ein Modell, in dem ein *Wuerfel* Oberklasse von *GezinkterWuerfel* ist. Den Code für dieses Modell finden Sie in WS2019/a00 mit englischen Klassen- und Dateinamen in *ws2015a00/v1* ausarbeiten.

Sie können aber auch argumentieren, dass ein *Wuerfel* ein *GezinkterWuerfel* ist, der mit der Wahrscheinlichkeit $\frac{1}{6}$ „gezinkt“ wurde. Diese Überlegung führt auf ein Modell, in dem die Methoden und das Attribut aus *GezinketerWuerfel* in die Klasse *Wuerfel* verschoben werden. Dieses Modell werde ich in *ws2015a00/v2* ausarbeiten.

Die Klasse *Wuerfel* braucht nur eine Methode *werfen*, die eine Zufallszahl zwischen 1 und 6 zurückgibt. Für gezinkte Würfel kann ich eine weitere Klasse bilden. Ganzzahlige Zufallszahlen aus dem Intervall $[0, 6)$ liefert die Klassenmethode *rand(6)* des Modules *Random*. Sie existiert in *Kernel* auch unter dem gleichen Namen. Ich kann also auch einfach schreiben: *rand(6)*. So erhalte ich einen Wert aus dem Intervall $[0, 6)$ und muss nur noch 1 hinzuaddieren. Damit erhalte ich:

```
def werfen
  rand(6) + 1
end
```

Eine alternative Schreibweise ist

```
def werfen
  rand(1..6)
end
```

Für den gezinkten Würfel überschreibe ich die Methode werfen. Für den gezinkten Würfel überlege ich wie folgt: Die 6 soll mit einer Wahrscheinlichkeit von p geworfen werden. Mit dieser Wahrscheinlichkeit liefert *Random.rand(1.0)* einen Wert zwischen 0 einschließlich und 1.0 ausschließlich. Da 0 der default für den Parameter ist und *rand(0)* eine Zufallszahl aus $[0, 1)$ liefert, kann ich den Parameter auch weglassn. Ich erfülle diese Forderung also, wenn ich für den Fall $p \leq p$ die 6 in *werfen* zurückgebe. Die anderen Zahlen kommen mit gleicher Wahrscheinlichkeit $\frac{1-p}{5}$. Damit habe ich nun alles beisammen, um die Methode *toss* für gezinkte Würfel zu schreiben:

```
def werfen
  Random.rand <= @bias ? 6 : Random.rand(1..5)
end
```

oder

```
def werfen
  rand <= @bias ? 6 : Random.rand(5) + 1
end
```

Selbstverständlich geht das auch mit *if* statt des ternären Operators.

Nun muss ich mir noch überlegen, wie ich die Funktionsfähigkeit der beiden Klassen demonstriere. Dazu schreibe ich mir ein Ruby-Skript *wurfeln.rb*. In diesem Skript würfele ich mit jedem Würfel 100 mal, zähle die Resultate und gebe sie aus. Als *zink_faktor* wähle ich 0.1, 0.5 und 0.9. Ich erwarte als Ergebnisse:

Ungezinkter Würfel: Alle Zahlen mit ca. 1/6.

bias = 0.1 : 6 kommt in ca. 0.1 der Fälle, alle anderen mit ca. $0.18 = 0.9/5$.

bias = 0.5 : 6 kommt in ca. 0.5 der Fälle, alle anderen mit ca. $0.1 = 0.5/5$.

bias = 0.9 : 6 kommt in ca. 0.9 der Fälle, alle anderen mit ca. $0.02 = 0.1/5$.

Schon bei 100 Würfen erscheint das Ergebnis plausibel. Bei 1000 Würfen bin ich schon sehr nah am erwarteten Durchschnitt. Bei 10.000 erhalte ich z. B. :

	Häufigkeit bei 10.000 Würfen			
Zahl	fair	p=0,1	p=0,5	p=0,9
1	1646	1792	1081	212
2	1653	1823	1025	188
3	1702	1759	952	183
4	1630	1768	997	185
5	1643	1805	1024	215
6	1726	1053	4921	9017