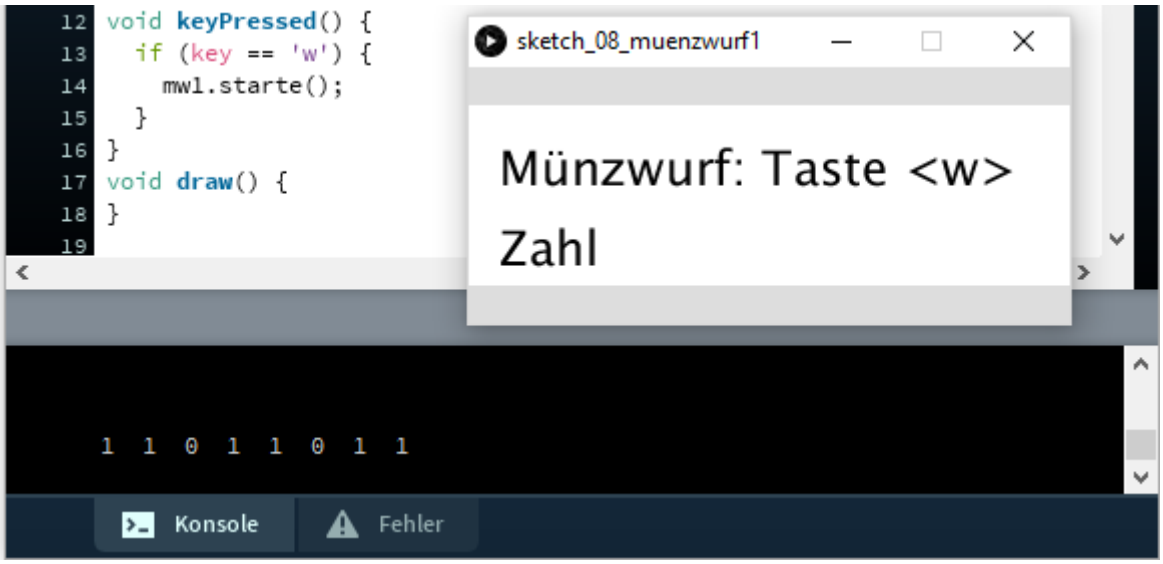
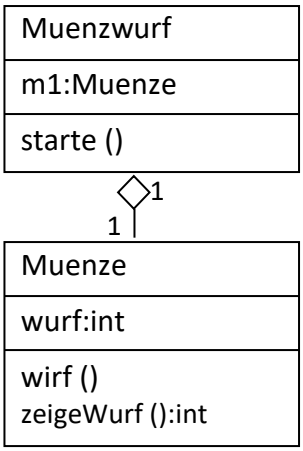
Auswahlstrukturen

Zweiseitige Auswahl

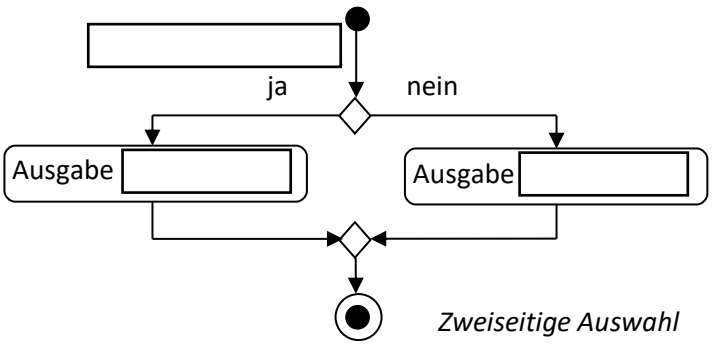
1. In einem Zufallsversuch kann der Wurf einer Münze simuliert werden. Die Würfe sollen mit Hilfe der Anweisung `print ()` in der Konsole angezeigt werden, die Ergebnisse als `text("Text",x,y)` (“Kopf” oder “Zahl”) im Zeichenfenster:



Hinweise:
- Die Abfrage der Taste <w> erfolgt mit der Anweisung `if (key == 'w') {` nicht `keycode`!).
- Für die Münze wird eine ganze Zufallszahl im Intervall [0;1] benötigt:
  `0` soll für „Kopf“ stehen und `1` für „Zahl“ (vgl. Arbeitsblatt 2.6.2 – 05, S. 3).
- In dem Klassendiagramm sind die Attribute und Methoden dargestellt.



Ergänze das Aktivitätsdiagramm für die zweiseitige Auswahl:



Zweiseitige Auswahl

➢ Bei der **zweiseitigen Auswahl** wird *entweder* die eine *oder* die andere von zwei Sequenzen des Programms ausgeführt. Schreibweise für die zweiseitige Auswahl:

```
if (m1.zeigeWurf () == 0) {
  text("Kopf",15,80);
}
else {
  text("Zahl",15,80);
}
```

- Erstelle ein Programm dazu. Speichere die Datei unter `muenzwurf1`. Teste das Programm, indem du die Münze mehrfach wirfst und beobachtest, ob die Reihenfolge der Ereignisse zufällig ist.

Bilder

In Processing können auch Bilddateien eingefügt und verändert werden.

- Zuerst wird ein Bildobjekt des Datentyps `PImage` erstellt. Im Beispiel wird das Objekt `eins` erstellt und die Datei `eins.gif` geladen, die sich im Verzeichnis `wuerfelansichten` befindet:

```
PImage eins;
...
void setup()
{
  ...
  eins = loadImage("../wuerfelansichten/eins.gif");
  ...
}
```

- Dann wird das Bildobjekt an der Position (`x, y`) in das Zeichenfenster eingefügt:
  `image(eins,x,y);`

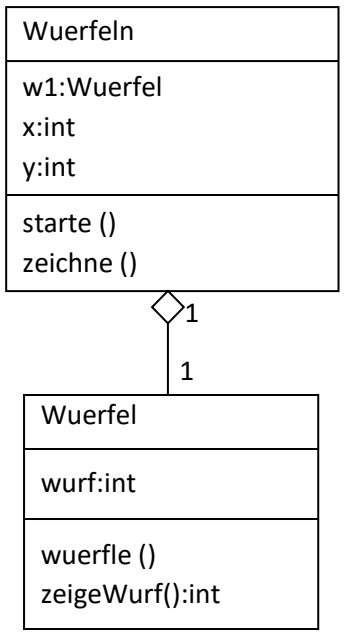
Der Attributwert für den Dateinamen des Bildes kann jederzeit geändert und das Bildobjekt neu gezeichnet und auch neu positioniert werden.

Mehrseitige Auswahl

Ein mögliches Zufallsgerät ist der Würfel.
In Processing kann man das Würfeln simulieren und die Ergebnisse der Versuche auswerten. Da bei dem Werfen eines Würfels mehr als zwei Ereignisse eintreten können, benötigt man die **mehrseitige Auswahl**.

2. Mit einem Processing-Programm soll der Wurf mit einem Würfel simuliert und das Ergebnis grafisch dargestellt werden (`wuerfeln1`).

   Hinweise:
   - Am einfachsten änderst du das Programm `muenzwurf1` ab. (Vorlagedatei v24_`muenzwurf1`).
   - Kopiere das Verzeichnis `wuerfelansichten` aus den Vorlagen in dein Programmverzeichnis.
   - Die Bildobjekte sollten global deklariert werden, also gleich zu Beginn des Programms.
   - Die Attribute `x` und `y` der Klasse `Wuerfeln` sind für die Koordinaten der Bilder vorgesehen, falls später mehrfach gewürfelt werden soll.



sketch_wuerfeln1
ws1:Wuerfeln eins:PImage zwei:PImage drei:PImage vier:PImage fuenf:PImage sechs:PImage

wuerfeln1:Zeichenfenster
Breite=600 Höhe=600 background=255

3. Zusatzaufgabe: Ergänze einen zweiten Würfel, der rechts neben dem ersten Würfel gezeichnet wird.

4. Erweitere das Programm `gluecksrad3` nach den unten stehenden Spielregeln zu einem Glücksspiel (Vorlagendatei `V25-gluecksrad3`). Speichere die Datei als Version `gluecksrad8`.

   - Version 1: Gedreht wird durch Drücken der Taste <d>.
   - Version 2:
     - Die Farbfelder erzielen unterschiedliche Punktewerte (gelb: 20; blau: 100, grün:10, magenta: 30, cyan: 50; rot: 40)
     - Bei jedem Versuch soll eine entsprechende Meldung wie in der Abbildung rechts angezeigt werden.
   - Version 3: Durch Drücken der Taste <n> kann ein neues Spiel begonnen werden.

Hinweise:

- Die Summe der Drehvorgänge muss auf ganze Zahlen von 1 bis 6 umgerechnet werden:
  - Die Anzahl der Drehungen kann mit Hilfe der Funktion `floor()` auf 0-360 Drehungen zurückgerechnet werden: `AnzahlDrehungen=n-floor(n/360*360);`
    Das funktioniert so: Zuerst wird `n` durch `360` dividiert. Da `n` eine ganzzahlige Variable ist, wird die Kommastelle abgeschnitten, was die Anzahl der kompletten Umdrehungen ergibt. Dieses Ergebnis wird mit `360` multipliziert und ganzzahlig (`floor`) von `n` abgezogen.
  - Daraus kann der Kreissektor bestimmt werden, auf den die Markierung zeigt:
    `Sektor=floor(AnzahlDrehungen/60)+1;`
- Bei der *mehrseitigen Auswahl* wird genau eine aus mehreren Sequenzen des Programms ausgeführt. Das Struktogramm für die Punktewertung:

Falls Sektor ==						
1	2	3	4	5	6	sonst
gelb: 20 Punkte	blau: 100 Punkte	grün: 10 Punkte	magenta: 30 Punkte	cyan: 50 Punkte	rot: 40 Punkte	

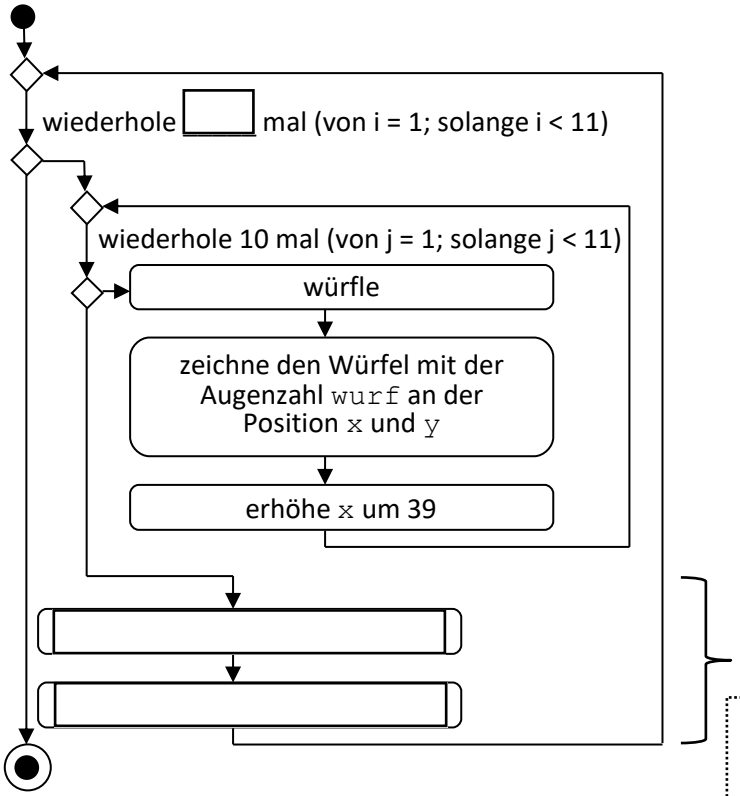
➢ Teste das Programm und versuche, eine möglichst hohe Punktzahl mit möglichst wenigen Versuchen zu erreichen.

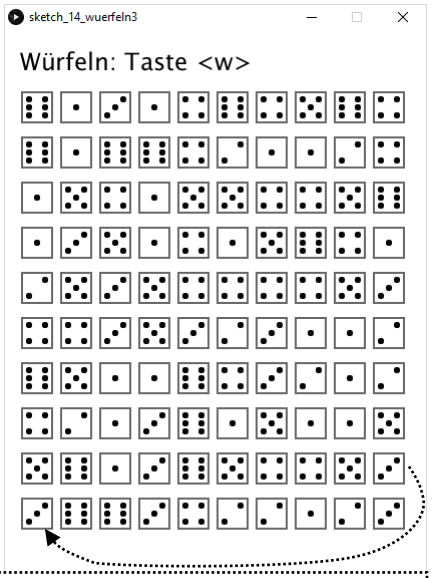
Wiederholungsstrukturen

Zählschleifen

Wuerfeln
w1:Wuerfel x:int y:int **i:int** **j:int**
starte () zeichne ()

5. In einem Zufallsversuch sollen nacheinander hundert Würfe mit einem Würfel durchgeführt werden.
   In dem Klassendiagramm rechts sind die Attribute und Methoden dargestellt.
   - Ergänze das Aktivitätsdiagramm unten für die Methode `start ()`.





Nach 20 Würfen wird `x` wieder auf 10 gesetzt und `y` um 45 erhöht, damit der nächste Wurf am linken Rand des Zeichenfensters weiter unten gezeichnet wird.

   - Ergänze dein Programm `wuerfeln1` bzw. die Vorlagendatei `v26_wuerfeln1`.
     Speichere das Programm unter `wuerfeln3`.

➢ Wenn die Anzahl auszuführender Wiederholungen von vorneherein feststeht, kann eine **Zählschleife** verwendet werden. In Processing lautet die Syntax für eine Zählschleife:

```
for (var i=(Startwert); (Endebedingung); i++) {
  Anweisungen; ...
}
```

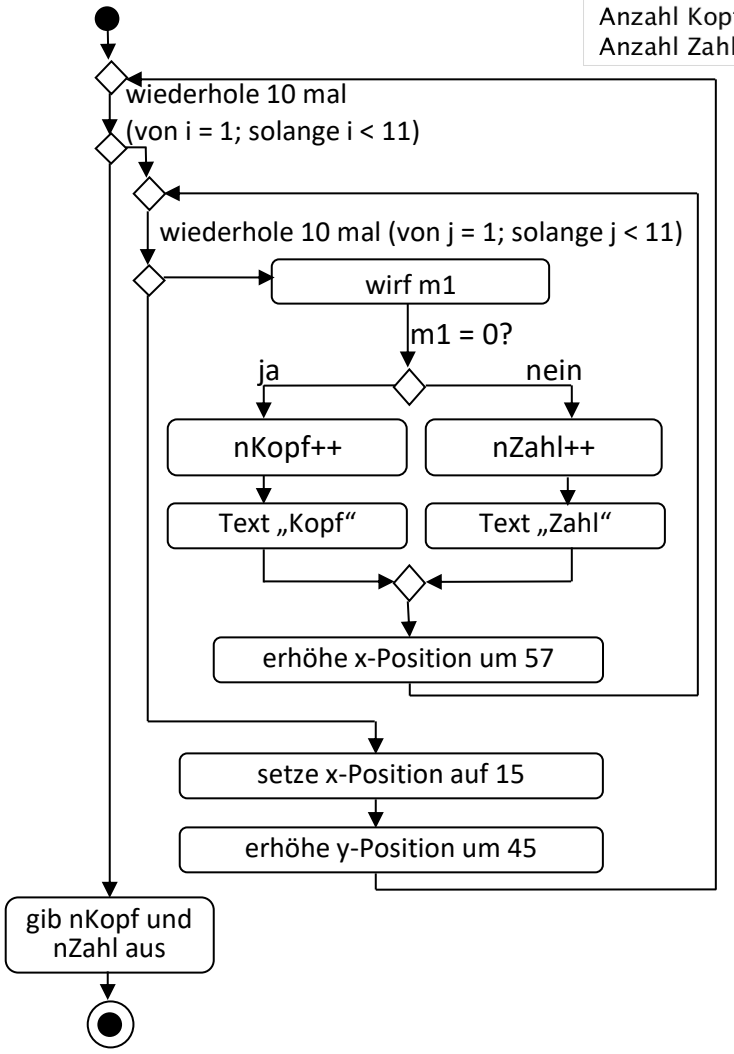
Hinweise:
- In der Methode `wuerfle ()` des Objekts `w1` wird eine Zufallszahl von 1 bis 6 ermittelt.
- Die aktuelle x- bzw. y-Position des Würfels wird in den Variablen `x` und `y` gespeichert.
- Die Anzeige des Würfels mit der jeweiligen „Augenzahl“ erfolgt in der Methode `zeichne ()`.
- Danach wird der Wert der Variablen `x` um 39 erhöht, damit der nächste Wurf weiter rechts angezeigt wird.
- Nach 20 Würfen wird `x` wieder auf 10 gesetzt und `y` um 45 erhöht, damit der nächste Wurf am linken Rand des Zeichenfensters weiter unten gezeichnet wird.
- Da in einer Zeile des Zeichenfensters nur zehn Würfel gezeichnet werden können, müssen die 100 Würfe in 10 mal 10 Wiederholungen aufgeteilt werden.

6. Die Anzahl der Würfe und das arithmetische Mittel aller „gewürfelten“ Werte sind unterhalb der Würfel in einem Textobjekt anzuzeigen.
   Ergänze dein Programm `wuerfeln3`, speichere als `wuerfeln4`.

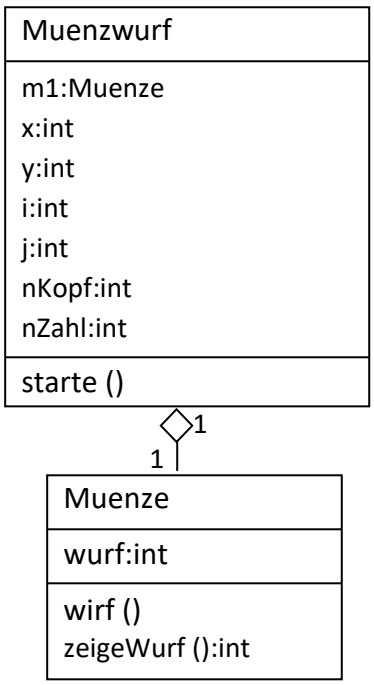
   Hinweis: Die Funktion `float ()` wandelt ganzzahlige Werte in Kommazahlen um.

7. In einem Zufallsversuch sollen nacheinander hundert Würfe mit einer Münze durchgeführt werden.
   Die Anzahl der jeweiligen Ereignisse („Kopf“ oder „Zahl“) soll angezeigt werden.
   - Beachte das Klassendiagramm und das Aktivitätsdiagramm unten.
   - Ergänze die Anweisungen in deinem Programm `muenzwurf1` oder verwende die Vorlagendatei `v27_muenzwurf1`. Speichere die Datei unter `muenzwurf2`.

Aktivitätsdiagramm:

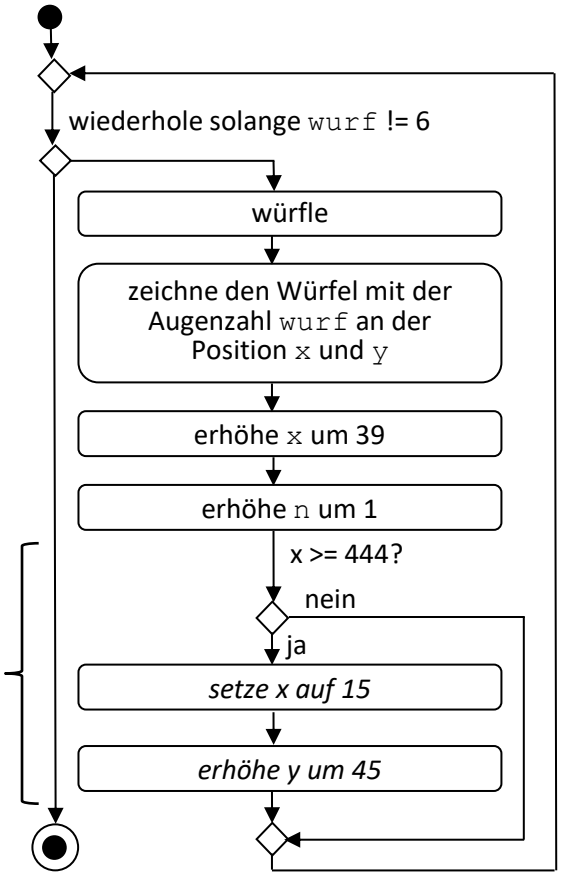


Klassendiagramm:



Wiederholung mit Ausführungsprüfung

8. In dem Programm `sechserwuerfeln1` soll so lange gewürfelt werden, bis die Augenzahl Sechs beträgt.

   ➢ Jetzt steht die Anzahl der Wiederholungen **nicht** von vorneherein fest. Dafür kann man eine *Wiederholung mit Ausführungsprüfung* verwenden.

   Schreibweise in Processing:

   ```
   while (Anfangsbedingung) {
        (Anweisungen) ...
   }
   ```

   Hinweise:
   - Verwende das Programm `wuerfeln3` als Vorlage (bzw. `v28_wuerfeln3`).
   - Da jetzt keine Zählschleifen mehr vorhanden sind, müssen die Werte der Attribute `x` und `y` mit Hilfe einer Auswahlstruktur zugewiesen werden.

9. Bei dem Programmtest tritt ein Problem auf:
   Würfelt man ein zweites Mal, wird die Wiederholung gar nicht durchlaufen. Das liegt daran, dass der Würfel zuletzt den Wert Sechs hatte.
   Dieses Problem kann gelöst werden, indem in der Klasse `Wuerfel` eine Methode `neu ()` ergänzt wird, um das Attribut `wert` wieder auf 0 zu setzen (`sechserwuerfeln2`).

10. Bei weiteren Programmtests fällt ein anderer Mangel auf: Wird auf Anhieb eine Sechs gewürfelt, stimmt die Meldung grammatikalisch nicht („Du hast 1 Würfe benötigt, um eine Sechs zu würfeln.“)
    Dieser Mangel kann mit Hilfe einer zweiseitigen Auswahl behoben werden.
    Ergänze das Programm und speichere die Datei unter `sechserwuerfeln3`.

11. In dem Programm `paschwuerfeln1` soll mit zwei Würfeln gewürfelt werden, bis ein Pasch vorliegt, also die Augenzahl beider Würfel gleich ist.

    - Codiere die Klassen und den Algorithmus. Ergänze dazu deine Programmversion `sechserwuerfeln3` oder die Vorlagedatei `v29_sechserwuerfeln3`.

    Hinweis:
    Zu Beginn ist darf die Augenzahl der beiden Würfel nicht gleich sein, weil sonst die Wiederholungsstruktur nicht durchlaufen wird.