Einsatz von *Processing* für *Java*-Programme

Inhaltsverzeichnis

1 Einsatzmöglichkeiten

Processing ist eine auf die Einsatzbereiche Grafik, Simulation und Animation spezialisierte objektorientierte, stark typisierte Programmiersprache und ermöglicht, mit einfachen Mitteln Interaktionen und visuelle Elemente zu programmieren. Die Klassen sind in der Programmiersprache *Java* entwickelt; die Bibliotheken berücksichtigen vor allem die Themen Video, Grafik, Grafikformate, Sound, Animation, Typografie, 3-D, Simulation, Datenzugriff und Datentransfer.
Die Klassenbibliotheken der Programmiersprache lassen sich z. B. gut zur Spieleentwicklung sowie für die Entwicklung von Android-Apps und Netzwerkanwendungen einsetzen.

2 Download und Installation

Das Werkzeug *Processing* ist kostenlos (Open Source) per Download erhältlich, gut dokumentiert und kann unter Windows-, Linux- und MacOS-Betriebssystemen installiert werden.
Bei Windows-Systemen erhält man nach dem Download eine zip-Datei, die nur entpackt werden muss. Prinzipiell könnte man sofort die mitgelieferte eigene Entwicklungsumgebung von *Processing* zur Entwicklung interaktiver Grafikprogramme einsetzen.
Möchte man jedoch die Bibliotheksklassen in *BlueJ* nutzen, so muss lediglich das jar-Paket `core.jar` (`..\processing-x.y.z\core\library\core.jar`) in den Bibliotheksordner des Dateispeicherorts von *BlueJ* (z. B. nach `C:\Program Files (x86)\BlueJ\bluej\lib\userlib`) kopiert werden.
Der Unterschied bei der Programmierung von Klassen in beiden Werkzeugen ist unter dem Punkt „7 Implementierung von Klassen mit den Werkzeugen von *Processing* bzw. *BlueJ*“ genauer dokumentiert.

3 Die Klasse `PApplet` als Oberklasse nutzen

Die wichtigste Klasse innerhalb der Programmbibliothek von *Processing* ist die Klasse `PApplet`. Mit dem Konzept der Vererbung ist es möglich, dass jede selbst geschriebene Klasse, die `PApplet` als Oberklasse festlegt, deren Attribute (z. B. zur Speicherung der aktuellen Mauskoordinaten oder der zuletzt gedrückten Taste der Tastatur) und Methoden (z. B. Methoden zum Zeichnen geometrischer Figuren bzw. zur Farbauswahl oder verschiedene Ereignismethoden (Eventmethoden)) direkt nutzen bzw. überschreiben kann.
Jede (mit *BlueJ* entwickelte) Klasse, die von der Klasse `PApplet` erbt, muss als `public` deklariert sein und den Pfad der Oberklasse `processing.core-PApplet` angeben. Ebenso müssen alle Methoden, die in der Klasse überschrieben werden müssen, als `public` deklariert sein. Dies sind die Methoden `settings`, `setup` und `draw` sowie Ereignismethoden wie `mousePressed`, die weiter unten noch genauer erläutert werden. Dies geschieht, indem man `class` bzw. `void` den Bezeichner `public` voranstellt.

Beispiel:

```
public class Anwendung extends processing.core.PApplet {
  public void settings() {
     ...
  }
  public void setup() {
     ...
  }
  public void draw() {
     ...
  }
  public void mousePressed() {
     ...
  }
  ...
}
```


Seite zu: Informatik 2
ISBN-13: 978-3-12-731141-9

Autorin: Dr. Siglinde Voß

4 Programmsteuerung eines *Processing*-Programms

Für die Steuerung des Programmflusses in *Processing* gelten folgende Regeln:

(1) Ein *Processing*-Programm wird immer mit der Methode `main` vom Anwender gestartet. Diese Methode muss aufgerufen werden.

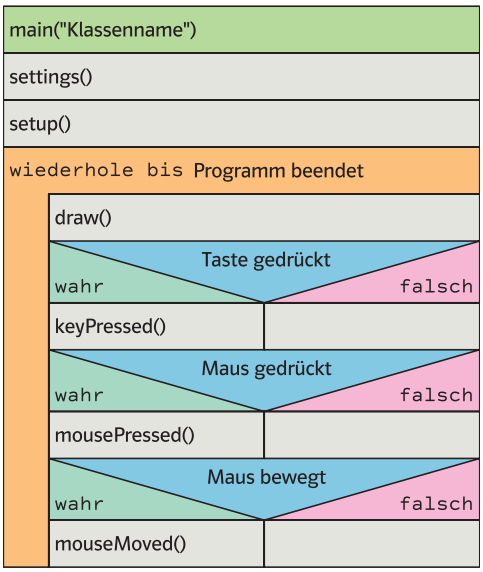
(2) Anschließend wird einmal automatisch die Methode `settings` ausgeführt (zur Einstellung der Fenstergröße). Diese Methode muss definiert werden.

(3) Dann wird `setup` (falls definiert) einmal automatisch ausgeführt (z. B. für die Zuweisung sinnvoller Anfangswerte für Attribute oder zum Aufruf von Methoden, die zu Beginn ausgeführt werden sollen).

(4) Schließlich wird (falls definiert) automatisch die Methode `draw` so lange (standardmäßig 60-mal pro Sekunde) wiederholt, bis das Programm beendet wird.

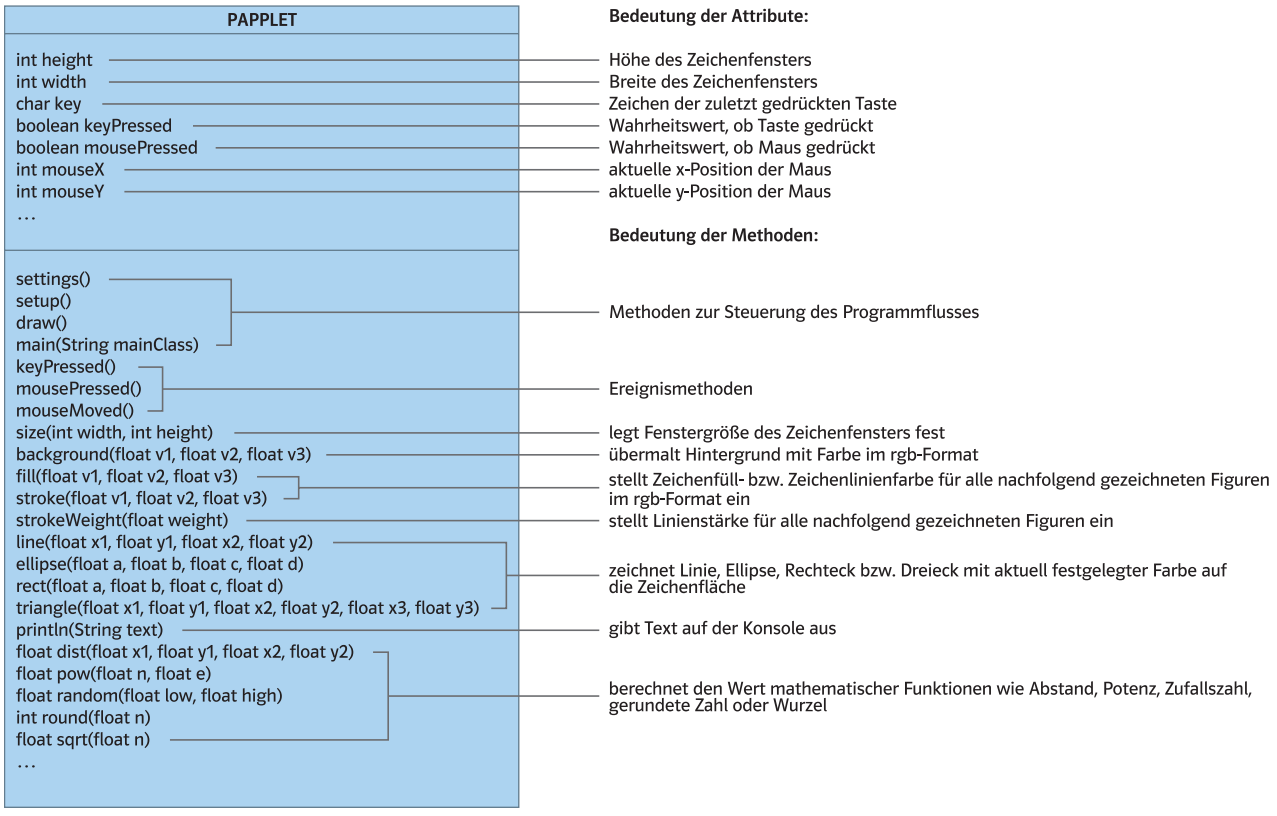
(5) Die Methode `draw` wird dabei höchstens von den ereignisorientierten Methoden `keyPressed`, `mousePressed`, `mouseMoved` ... (falls definiert) unterbrochen, die jedes Mal dann ausgeführt werden, wenn der Anwender eine Taste bzw. die Maus drückt bzw. bewegt. Der Kontrollfluss setzt dann die Wiederholung der `draw`-Methode fort.

***Processing*-Programm**



5 Klassenkarte von `PApplet`

Die abgebildete Klassenkarte fasst nur einige der wichtigsten Attribute und Methoden zusammen. (Eine komplette Liste mit Erläuterungen und Beispielimplementierungen findet man im Internet.)



6 Beispiel für die Implementierung einer kleinen Anwendung

Das folgende Beispiel stellt die Implementierung einer Anwendung für *BlueJ* dar. Die einzelnen Codezeilen sind kommentiert.

```
public class Anwendung extends processing.core.PApplet {
  void starten() {
    main("Anwendung");                  // Diese Methode muss aufgerufen werden, um das Pro-
                                        // gramm zu starten. Der Wert des Eingabeparameters
                                        // muss gleich dem Klassenbezeichner sein.
  }

  public void settings() {              // Methode, die direkt nach dem Programmstart ausge-
    size(500, 300);                     // führt wird und überschrieben werden muss, um z.B.
                                        // mit dem Aufruf von size die Fenstergröße
                                        // festzulegen.
  }

  public void setup() {                 // Methode, die überschrieben werden muss und direkt
    background(250, 250, 100);          // nach settings genau einmal ausgeführt wird, um z.B.
    fill(0, 255, 0);                    // den Hintergrund farbig zu übermalen und die nach-
                                        // folgend verwendete Zeichenfüllfarbe grün
                                        // festzulegen.
  }

  public void draw() {                  // Methode, die überschrieben werden muss, und
    ellipse(mouseX, mouseY, 50, 50);    // direkt nach setup standardmäßig 60-mal pro Sekunde
                                        // wiederholt ausgeführt wird, um z.B. einen Kreis an
                                        // die aktuelle Mausposition zu zeichnen.
  }

  public void mousePressed() {          // Methode, die draw unterbricht, um beim Drücken der
    fill(random(255), 0, random(255)); // Maustaste als aktuelle Zeichenfüllfarbe eine zu-
                                        // fällige Farbe festzulegen.
  }
}
```

7 Implementierung von Klassen mit den Werkzeugen von *Processing* bzw. *BlueJ*

Das obige Beispiel nutzt die *Processing*-Bibliothek `core.jar` innerhalb der Entwicklungsumgebung *BlueJ*.
Es gibt aber auch die Möglichkeit, die *Processing*-Entwicklungsumgebung direkt zu nutzen.
Die Programmiersprache ist *Java*. Es gibt bis auf einige Anpassungen in der „Anwender-Klasse“ keine Unterschiede. Weitere Klassen (die später programmiert werden, um diese zu referenzieren) unterscheiden sich in den beiden Programmierumgebungen nicht.

Gegenüberstellung der Anwenderklasse mit beiden Entwicklungsumgebungen an einem Beispiel:

BlueJ-Entwicklungsumgebung

```
public class Anwendung extends
        processing.core.PApplet {

  void starten() {
    main("Anwendung");
  }

  public void settings() {
    size(500, 300);
  }

  public void setup() {
    noStroke();
  }

  public void draw() {
    background(255, 255, 255);
    fill(mouseX, 0, mouseY);
    ellipse(mouseX, mouseY, 30, 30);
  }
}
```

Processing-Entwicklungsumgebung

```
// Nur in der „Anwenderklasse“ kann auf
// den Klassenkopf verzichtet werden.

// Gestartet wird mit dem Startbutton.
// „public“ wird nicht benötigt.


void settings() {
  size(500, 300);
}

void setup() {
  noStroke();
}

void draw() {
  background(255, 255, 255);
  fill(mouseX, 0, mouseY);
  ellipse(mouseX, mouseY, 30, 30);
}
```