DER JAVA-PLOTTER

Stephan Euler

Technische Hochschule Mittelhessen
Fachbereich MND

ABSTRACT

Die Java-Bibliothek Plotter bietet eine einfache Möglichkeit zur grafischen Darstellung von Werten. Damit eröffnet sie die Möglichkeit, frhzeitig bei Lehrveranstaltungen zur Programmierung in Java entsprechende Beispiele und Übungen einzusetzen. Ein typischer Anwendungsfall ist das Zeichnen von Funktionsverläufen oder Simulationsergebnissen. Gleichzeitig bietet die Bibliothek auch weitergehende Möglichkeiten, um beispielsweise den Ablauf von Sortierverfahren oder Lösungsalgorithmen zu visualisieren. In diesem Beitrag wird zunchst das Grundkonzept vorgestellt. Anschließend werden anhand eines Beispiels der konkrete Einsatz erläutert. Schließlich wird über den Einsatz der Bibliothek in einer Lehrveranstaltung berichtet.

Index Terms— Java Programmierung

1. EINLEITUNG

Eine Reihe von Ansätzen wurde entwickelt, um Programmier-Anfängern den Einstieg zu erleichtern [1]. Eine typische Anwendung ist dabei, Objekte in einer virtuellen Welt zu steuern. Beispielsweise kann man im Java-Hamster-Modell einen virtuellen Hamster Aufgaben lösen lassen. Damit werden spielerisch sowohl die Grundlagen der imperativen als auch der objektorientierten Programmierung vermittelt [2, 3]. Die Programmierung erfolgt dabei in einer speziellen Entwicklungsumgebung – dem Hamster-Simulator.

In diesem Beitrag wird ein anderer Ansatz verfolgt. Programmiert wird von Anfang direkt in Java innerhalb einer der üblichen Entwicklungsumgebungen wie Eclipse oder BlueJ. Gleichzeitig wird durch eine entsprechende Klassen-Bibliothek eine sehr einfache Mglichkeit zur grafischen Ausgabe bereit gestellt. Damit kann man frühzeitig mit ansprechenden und im Idealfall motivierenden Beispielen und Aufgaben beginnen. Gleichzeitig ist die Bibliothek umfangreich genug, um aufwändigere Anwendungen zu ermöglichen.

2. GRUNDKONZEPT

Java verfügt über eine umfangreiche Bibliothek von Klassen zur Grafikprogrammierung. Der Einstieg ist nicht wirklich schwierig, erfordert aber doch solide Grundkenntnisse. Andererseits ist es sicherlich motivierend, wenn man frühzeitig ansprechende grafische Darstellungen erzeugen kann. Dieser Gedanke ist der Ausgangspunkt für das Plotter-Projekt. Ziel ist es, eine möglichst einfache Schnittstelle zur grafischen Programmierung anzubieten. Bereits beim Einstieg in die Programmierung sollen erste grafischen Darstellungen wie z. B. Funktionsverläufe entstehen. Durch entsprechende Aufgaben kann damit der Umgang mit grundlegende Konzepten wie Variablen, Kontrollstrukturen, Methoden und Klassen geübt werden.

Kernstück des Projektes ist die Klasse `Plotter`. Ein `Plotter`-Objekt (vereinfacht gesprochen *ein Plotter*) dient als Hülle um die Zeichenmethoden. Gleichzeitig übernimmt es die Verwaltung der Daten. Im einfachsten Fall schickt man nur einige Werte an einen Plotter. Der Plotter speichert die Werte intern und erstellt daraus automatisch eine grafische Darstellung mit angepassten Wertebereichen. Über den Schnell-Einstieg hinaus bietet der Plotter allerdings auch vielfältige Möglichkeiten zur Verfeinerung der Darstellungen. Man kann Texte hinzufügen, Fonts, Zeichenstile und Farben ändern oder sogar Bilder einbauen.

Der Komfort ist allerdings nicht ganz umsonst. Der zusätzliche Rechen- und Speicheraufwand hält sich meistens in Grenzen und wirkt sich nur bei sehr großen Datenmengen störend aus. Schwerwiegender ist, dass in manchen Situationen die interne Organisation des Plotters nicht optimal zur Aufgabenstellung passt. Eine Lösung unter direkter Verwendung der Java-Bibliothek ist dann einfacher. Der Plotter kann – und soll auch gar nicht – die Einarbeitung in die Java Programmierschnittstellen und Bibliotheken ersetzen. Aber er bietet einen einfachen Einstieg und soll beim Erlernen von Java helfen.

3. EIN BEISPIEL

Die Klasse `Plotter` ist im wesentlichen dazu gedacht, Kurven zu zeichnen. Als Einstieg zeigt Listing 1 eine einfache Klasse mit einer Methode `main()` zur Darstellung der Funktion $sin(x)$ im Bereich $[-\pi, \pi]$. Zur besseren Übersicht sind die Klassen in mehrere Pakete (Package) eingeteilt. Die direkt zu Plotter gehörenden Klassen finden sich in einem Paket `plotter`. Demgegenüber sind die Beispiele in einem Paket

Listing 1. Erstes Beispiel

```
package demos;

import plotter.Graphic;
import plotter.Plotter;

public class Demo1 {

  public static void main(String[] args)
  {
    Graphic graphic =
         new Graphic("Demo 1");
    Plotter plotter =
         graphic.getPlotter();

    for (double x=-Math.PI;
         x<=Math.PI;  x+=0.01) {
      plotter.add( Math.sin(x) );
    }
    graphic.repaint();
  }
}
```

demos zusammen gefasst. Wie in Zeile 1 festgelegt, gehört auch das erste Beispiel zu diesem Paket. Um auf die beiden benötigten Klassen aus dem Paket plotter direkt zugreifen zu können, werden sie importiert.

Die eigentliche Verarbeitung erfolgt in der main-Methode. Zunächst wird ein Objekt der Klasse Graphic angelegt. Die Klasse Graphic ist von JFrame aus der Swing-Bibliothek abgeleitet. Sie erstellt ein Hauptfenster und darin eingebettet ein Plotter-Objekt sowie eine Statuszeile. Man kann einen Plotter auch in eigene Anwendungen einbauen. Aber zur Vereinfachung übernimmt in unserem Beispiel das Graphic-Objekt diese Aufgabe. Den implizit erzeugten Plotter erhält man über eine Getter-Methode.

Nach diesen Vorbereitungen können wir Datenpunkte eingeben. Im Beispiel werden in einer Schleife Werte der Funktion $sin(x)$ mit der Methode add an den Plotter übergeben. Wenn wie in dieser Form nur y-Werte übergeben werden, übernimmt der Plotter eine automatische Nummerierung. Die eingegebenen Werte werden daher intern zu Wertepaaren in der Form

x	y
0	$\sin(-\pi)$
1	$\sin(-\pi + 0.01)$
2	$\sin(-\pi + 0.02)$
...	...

ergänzt. Sind alle Daten eingegeben, wird mit dem Aufruf der Methode repaint() eine Aktualisierung ausgelöst. Dabei bestimmt der Plotter anhand der vorliegenden Daten den darzustellenden Bereich und trägt die Punkte dort ein. Bild 1 zeigt das resultierende Fenster.

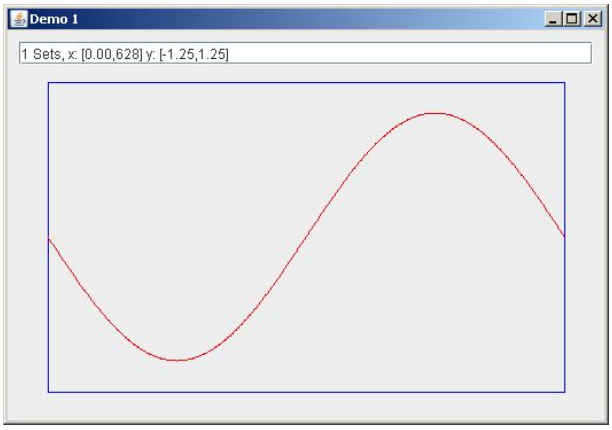

Fig. 1. Erstes Beispiel für Plotter

Das Beispiel zeigt, wie man mit minimalem Aufwand – einfach durch Eintragen von Werten – mit einem Plotter eine graphische Darstellung erhält. Aber selbst bei einfachen Programmierübungen wird man schnell eine verfeinerte Darstellung wünschen. Die Klasse Plotter stellt dazu eine ganze Reihe von Methoden zur Verfügung. Ausgehend von unserem ersten Beispiel werden wir einige Möglichkeiten durchgehen. Eine umfangreiche Beschreibung der einzelnen Methoden findet man in der Dokumentation [4].

In unserem Beispiel wird durch die automatische Nummerierung als x-Bereich 0 bis 628 angegeben. Schöner wäre es, den tatsächlichen Bereich angezeigt zu bekommen. Dazu muss man lediglich beim Eingeben der Werte die zugehörigen x-Werte mitgeben. Für diesen Fall steht eine andere Methode add(double x, double y) mit zwei Parametern zur Verfügung. Eine entsprechend geänderte Klasse zeigt Listing 2. Bei dieser Gelegenheit wird auch der Darstellungsbereich angepasst und die Achsen werden beschriftet. Im einzelnen werden folgende Erweiterungen vorgenommen:

- der x-Bereich wird um den Faktor 1.1 vergrößert
- die Hauptachsen werden eingezeichnet
- für die y-Achse wird eine Unterteilung in Schritten von 0.25 gewählt
- auf der x-Achse sollen die Punkte $-\pi, -\pi/2, 0, \pi/2, \pi$ markiert werden, die entsprechenden Werte werden als Feld übergeben.
- für die Beschriftungen an der x-Achse sollen zwei Nachkommastellen angezeigt werden, Plotter verwendet das angegeben Format mit der Methode printf, um die Achsbeschriftungen zu erzeugen.

Die so ergänzte Darstellung zeigt Bild 2.

Listing 2. Verbesserte Version mit x-Werten und Achsbeschriftungen

```
...

  plotter.setXrange(-Math.PI * 1.1,
                    Math.PI * 1.1 );
  plotter.setXLine(0);
  plotter.setYLine(0);
  plotter.setAutoYgrid(0.25);
  double[] xgrid = {-Math.PI, -1,
                    0, 1, Math.PI};
  plotter.setXLabelFormat("%.2f");
  plotter.setXGrid(xgrid);

  for (double x=-Math.PI; x<=Math.PI;
                   x+=0.01) {
      plotter.add( x, Math.sin(x) );
  }
  graphic.repaint();
}
```

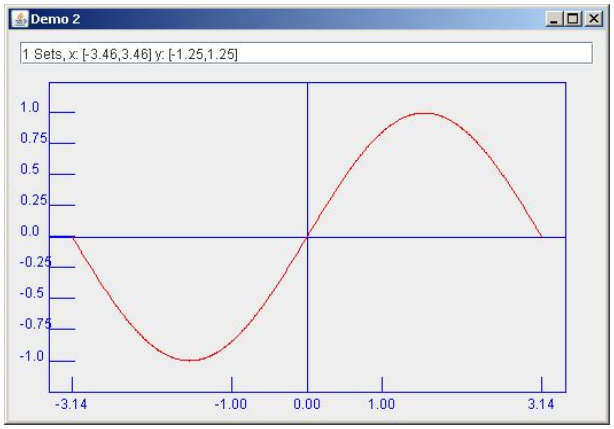

Fig. 2. Darstellung durch Klasse Demo2

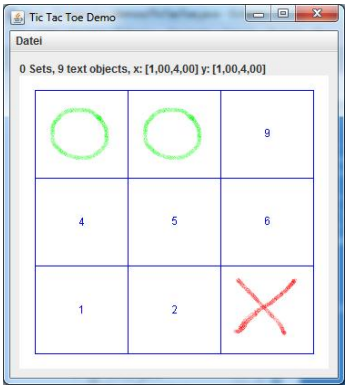

Fig. 3. Tic Tac Toe Spiel

4. WEITERE MÖGLICHKEITEN UND BEISPIELE

Standardmäßig werden alle an einen Plotter geschickten Daten als ein gemeinsamer Datensatz behandelt. Es ist aber auch möglich, die Daten getrennt zu halten. Beispielsweise kann man bei der Methode `add` zusätzlich als ersten Parameter den Namen des jeweiligen Datensatzes mitgeben. So wird unser Beispiel durch

```
for (double x=-Math.PI; x<= Math.PI;
                          x+=0.01) {
    plotter.add("sin", x, Math.sin(x) );
    plotter.add("cos", x, Math.cos(x) );
}
```

auf zwei Datensätze *sin* und *cos* erweitert. Jeder Datensatz wird getrennt gezeichnet, wobei man individuelle Werte für Zeichenstil und Farbe einstellen kann.

Zusätzlich zu Daten können Texte und Bilder eingefügt werden. Dabei können über entsprechende Methoden Eigenschaften wie Größe und Position gesetzt werden. Bild 3 zeigt als Beispiel eine Umsetzung des Spiels *Tic Tac Toe*. Zunächst werden die Feldnummern als Text angezeigt. Nach einem Spielzug wird dann das Bild mit dem Spielersymbol auf das entsprechende Feld gelegt.

Alle von einem Plotter verwaltete Objekte – Daten, Texte und Bilder – können verändert oder auch wieder gelöscht werden. Somit können auch einfache Animationen programmiert werden. Insbesondere eignete sich der Plotter zur Visualisierung von Algorithmen oder Abläufen. In der Dokumentation [4] sind einige Beispiele wie Sortierverfahren, Lösung von Sudoko-Rätseln oder Bewegung von Langton-Ameisen beschrieben. Häufig wird rekurisve Programmierung anhand des Spiels *Türme von Hanoi* erläutert. Bild 4 zeigt eine Visualisierung mit Hilfe eines Plotters[1]. Die – in diesem Fall 20 – Scheiben werden durch entsprechend viele Datenstze mit jeweils den vier Eckpunkten eines Rechtecks dargestellt.

[1]Quellcode ist unter `code.google.com/p/plotter-demo` frei verfgbar

Fig. 4. Zwischenstand bei *Türme von Hanoi* mit 20 Scheiben

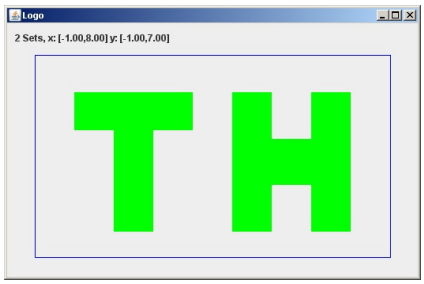

Fig. 5. Logo-Aufgabe

5. EINSATZ IN VORLESUNG

Der Plotter wird derzeit hauptsächlich in einer zweisemestrigen Vorlesung *Programmierung* im Studiengang Wirtschaftsmathematik eingesetzt. Eine der ersten bungsaufgaben für die Teilnehmer/innen ist, mittels Plotter einige Buchstaben wie in Bild 5 gezeigt auszugeben. Mit dieser Aufgabe lernen die Studierenden das Koordinatensystem sowei einfache Befehle kennen. Eine ähnliche zweite Aufgabe *Länderflaggen* soll ermutigen, je nach Kenntnisstand mehr oder weniger komplexe Flaggen zu bearbeiten.

Im weiteren Verlauf der Vorlesung folgen Aufgaben zur Darstellung von u. a. Funktionsverläufen, Histogrammen und Simulationsergebnissen. Als eines der ersten Beispiele für Objektorientierte Programmierung (OOP) werden Vektoren und Matrizen behandelt. Der Plotter dient dann zur Ausgabe. Die dabei entwickelte Darstellung einer Matrix wird anschließend verwendet, um ein Schachbrett mit den Lösungen des N-Damen-Problems zu zeichnen. Schließlich vereinfacht der Plotter auch im klassischen OOP-Beispiel *Geometrische Figuren* das Zeichnen der einzelnen Figuren.

Am Ende des zweiten Semesters bearbeiten die Teilnehmer/innen kleinere Programmierprojekte. Einige Teams setzen dabei wiederum einen Plotter als grafische Komponente ein. Beispielsweise werden damit Spiele wie *Vier gewinnt* oder *Roulette* umgesetzt.

Das Ziel, frühzeitig grafisch ansprechende Beispiele und Übungen umzusetzen, wird mit dem Plotter erreicht. Die Rückmeldungen der Studierenden sind gemischt. Einige nutzen den Plotter gerne und erkunden auch die vielen Möglichkeiten. Andere empfinden ihn als zusätzliche Hürde beim ohnehin nicht einfachen Einstieg in die Java-Programmierung. Eine systematische Evaluation des didaktischen Nutzens ist für den nächsten Vorlesungszyklus geplant.

6. ZUSAMMENFASSUNG

In diesem Beitrag ein Ansatz vorgestellt, um in Java mit einer Klassen-Bibliothek *Plotter* auf einfache Art und Weise Datenwerte grafisch darzustellen. Damit können auch ohne die sonst erforderlichen Kenntnisse in den entsprechenden Java Programmierschnittstellen und Bibliotheken beispielsweise Funktionsverläufe oder Simulationsergebnisse gezeichnet werden. In einer zweisemestrigen Vorlesung zur Java-Programmierung werden damit entsprechende Übungsaufgaben gestellt. Der Bogen spannt sich dabei vom Zeichnen einfacher Muster als Einstieg bis zu abschließenden Projekten mit durchaus bereits anspruchsvollen Anwendungen.

7. REFERENCES

[1] Sabine Terwelp and Markus Dahm, "Entwicklungsumgebungen für Informatik-Anfänger.," in *Mensch & Computer*. 2011, Mensch & Computer, pp. 371–374, Oldenbourg Wissenschaftsverlag.

[2] Dietrich Boles, *Programmieren spielend gelernt mit dem Java-Hamster-Modell*, Vieweg+Teubner, Wiesbaden, 2008.

[3] Dietrich Boles and Cornelia Boles, *Objektorientierte Programmierung spielend gelernt mit dem Java-Hamster-Modell*, Vieweg+Teubner, Wiesbaden, 2010.

[4] Stephan Euler, *Java Plotter*, TH Mittelhessen, `wan15.mnd.thm.de/plotter/cover.pdf`.