Java04 − Erste eigene Klasse

Noch einmal: Kreise-Figur erzeugen, nun „automatisch“

Um die rechts abgebildete Figur aus fünf Kreisen zu erstellen, müssen fünf Instanzen der Klasse „Kreis“ erzeugt werden.

Die Farben der Kreise von außen nach innen:
lila, blau, grün, gelb und rot.

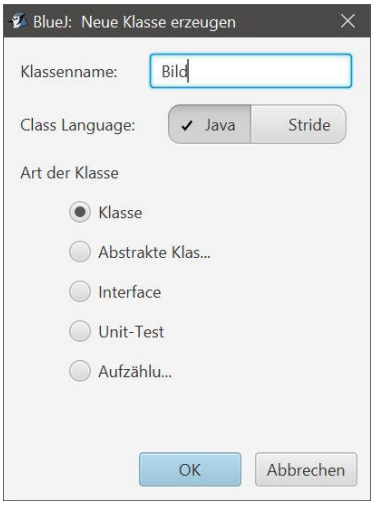
Programmieren heißt nicht:

einzelne Methoden mit der rechten Maustaste aus einem Kontextmenü auswählen. Das ist mühsam! Außerdem lässt sich so das Kreise-Bild nicht dauerhaft speichern (oder wiederholt „einfach“ erzeugen).

Programmieren heißt:

Methodenaufrufe sowie die nötigen Parameter in der richtigen Reihenfolge in eine Textdatei schreiben

Diese Textdatei wird als “Quelltext” bezeichnet. Der Quelltext wird dann kompiliert, das heißt in ausführbaren Maschinencode übersetzt. Wie von Zauberhand erscheint dann plötzlich in einem neuen Programmfenster das Bild mit den fünf Kreisen.



Schritt 1 − Neue Klasse erzeugen

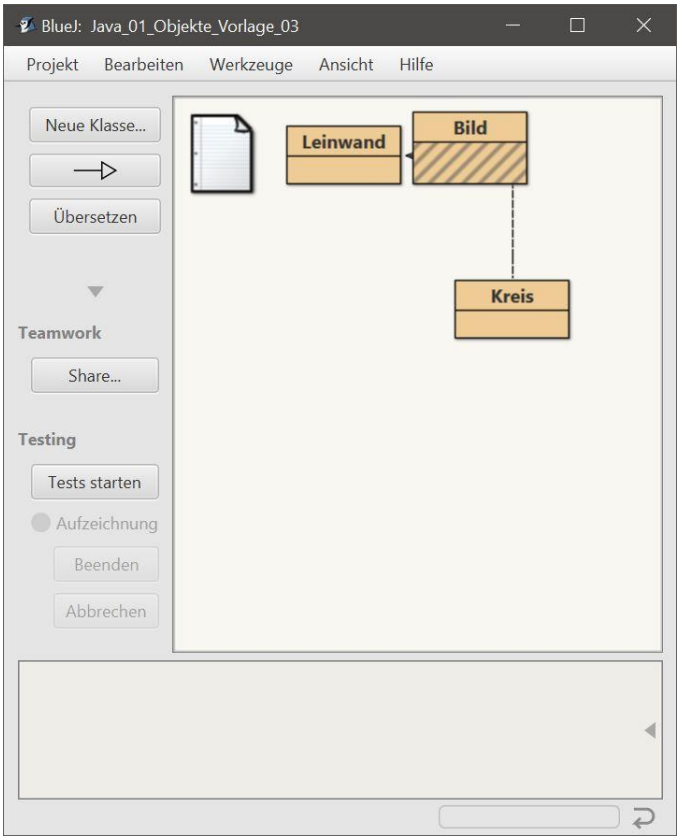
Öffne Dein BlueJ-Projekt und klicke auf den Button „Neue Klasse“ bzw. „New class“. Du wirst aufgefordert, der neuen Klasse einen Namen zu geben und einen Klassentyp zuzuweisen.

Wähle als Name “Bild” und als Typ “Klasse” bzw. “Class“, dann „OK“.

Danach müsstest Du im Hauptfenster eine neues Klassen-Symbol „Bild“ sehen. Du kannst dieses Klassen-Symbol im Hauptfenster verschieben.

Die Schraffierung der Klasse „Bild“ deutet darauf hin, dass Du den Quelltext dieser Klasse noch nicht erfolgreich kompiliert hast.

Schritt 2 − Objekt „bild1“ erzeugen



2a − Ein Versuch

Klicke mit der rechten Maustaste auf die neue Klasse. Du wirst feststellen, dass Du noch keine Objekte dieser Klasse erzeugen kannst. Vorher ist ein weiterer Schritt notwendig: Du musst den Java-Quelltext der neuen Klasse kompilieren (übersetzen).

2b − Kompilieren

Klicke mit der rechten Maustaste auf die neue Klasse und wähle den Befehl „Übersetzen“ bzw. „Compile“. Der Quelltext der Klasse Bild wird nun in maschinenlesbare Anweisungen übersetzt. Erst nach diesem Schritt kannst Du Objekte der Klasse erzeugen.

2c − Objekt erzeugen

Erstelle jetzt mit Hilfe der rechten Maustaste ein Objekt bild1 der neuen Klasse „Bild“.

2d − Methoden? Welche Methoden?

Klicke mit der rechten Maustaste auf das Objekt „bild1“. Es dürfte Dir die völlige Abwesenheit von Methoden auffallen. Unsere neue Klasse „Bild“ besitzt lediglich eine „beispielMethode()“ bzw. „sampleMethod()“, die wir aber eigentlich gar nicht benötigen.

Schritt 3 − Quelltext

Wir wollen der Sache jetzt auf den Grund gehen. Wieso existieren keine Methoden für die Objekte der Klasse „Bild“? Gegenfrage: Hast Du denn bereits eine einzige Methode programmiert? Na, dann wird es aber höchste Zeit dafür!

3a − Quelltext öffnen

Doppelklicke auf die Klasse „Bild“. Es öffnet sich ein Fenster mit dem Quelltext der Klasse. Schauen wir uns den Quelltext näher an. Er besteht fast nur aus (gut gemeinten) Kommentaren, die aber sowohl Anfänger wie auch Fortgeschrittene eher verwirren. Richtig sinnvoll sind diese Kommentare nur für Leute, die schon mal mit einer anderen Programmiersprache objektorientiert gearbeitet haben. Kommentare dienen an sich zur Erläuterung von komplizierten Java-Befehlen. Man muss keine Kommentarzeilen in den Quelltext schreiben, der Computer versteht auch so, was Du meinst. Aber denke an Dich selbst: Wenn Du später mal Deinen Quelltext verbessern willst, ist es sehr hilfreich, wenn Du bestimmte wichtige Stellen des Quelltextes ausreichend kommentiert hast.

3b − Quelltext aufräumen

Entferne sämtliche Kommentare aus dem Quelltext! Danach sieht das Ganze schon wesentlich aufgeräumter aus. Wenn Du nun auch noch die Zeile mit dem Attribut private int x sowie die Methode „beispielMethode()“ bzw. „sampleMethod()“ entfernst, hast Du einen wirklich minimalen Quelltext der Klasse „Bild“. Diesen Minimal-Quelltext wollen wir uns jetzt näher anschauen. Wenn Du möchtest, kannst Du auch den Quelltext komplett entfernen, der beim Erzeugen der Klasse Bild angelegt wurde.
Tippe dann einfach den folgenden Minimal-Quelltext in den Texteditor ein.
So lernst Du vielleicht schneller, wie eine Java-Klasse aufgebaut ist.

Schritt 4 − Der Minimal-Quelltext

Schauen wir uns den Minimal-Quelltext jetzt einmal genauer an und nehmen wir ihn Wort für Wort auseinander. Zunächst einmal der Minimal-Quelltext:

```
public class Bild {
   public Bild() { }
}
```

Wenn Du ein Programm wie Photoshop startest und den Befehl “Datei/Neu” wählst, so öffnet sich eine neue leere Zeichnung. Bei Word erzeugst Du mit dem gleichen Befehl ein neues leeres Textdokument. Ähnlich ist dieser Minimal-Quelltext zu verstehen. Es handelt sich um den syntaktisch völlig korrekten, aber gleichzeitig völlig leeren Quelltext einer Java-Klasse. Der Minimal-Quelltext besteht aus der Klassen-Deklaration, in die eine spezielle Methode eingebettet ist, nämlich der Konstruktor der Klasse. Der Konstruktor enthält hier keine einzige Zeile Quelltext, ist also leer.

Schritt 5 − Der Konstruktor

Der Konstruktor ist eine Methode, die immer dann aufgerufen wird, wenn ein neues Objekt der Klasse erzeugt werden soll. Angenommen, Du hast eine Klasse erzeugt, z. B. „Bild“. Im BlueJ-Fenster klicke jetzt mit der rechten Maustaste auf diese Klasse und wähle den Befehl new Bild(). Wenn Du das machst, rufst Du − ohne dass Du es merkst − den Konstruktor der Klasse auf und erzeugst so ein neues Objekt der Klasse „Bild“. Umgekehrt kann man sagen: Nur wenn die Klasse einen Konstruktor enthält, kann man Objekte dieser Klasse erzeugen.

Schritt 6 − Methoden

Wir wollen jetzt die erste richtige Methode für unsere Klasse schreiben.
Ergänze den Quelltext der Klasse Bild durch eine Methode bildMalen():

```
public class Bild {
   public Bild() { }
   public void bildMalen() { }
}
```

Kompiliere dann den Quelltext, erzeuge ein Objekt „bild1“ der Klasse „Bild“, und klicke dann mit der rechten Maustaste auf das neue Objekt. Tatsächlich taucht jetzt die neue Methode „bildMalen()“ im Kontextmenü des Objektes „bild1“ auf. Allerdings passiert nichts, wenn Du auf die Methode klickst. Das ist aber auch kein Wunder, schließlich hast Du noch keine Java-Befehle in die Methode hineingeschrieben.

Schritt 7 − Befehle

Ergänze den Quelltext der Klasse Bild wie folgt:

```
public class Bild {
   private Kreis kreis1;
   public Bild() {
      kreis1 = new Kreis(0, 0, "lila");
   }
   public void bildMalen() {
      kreis1.groesseAendernZentrieren(200);
   }
}
```

7.1 − Attribut deklarieren

Als erstes schreibst Du oberhalb des Konstruktors die folgende Zeile:

```
   private Kreis kreis1;
```

Damit deklarierst Du ein Attribut „kreis1“, welches gleichzeitig ein Objekt der Klasse „Kreis“ ist. Die Methode „bildMalen()“, die Du jetzt mit Inhalt füllen willst, muss ja etwas haben, auf das sie einwirken kann. Wenn innerhalb von bildMalen() ein Kreis in Größe und Position verändert werden soll, muss dieser Kreis erstmal vorhanden sein. Dieser Kreis wird durch das Objekt „kreis1“ repräsentiert. Der neu entstandene Pfeil vom Klassen-Symbol „Bild“ zum Klassen-Symbol „Kreis“ zeigt Dir, dass die Klasse „Bild“ jetzt tatsächlich auf die Klasse „Kreis“ zurückgreift. Man spricht hier auch von einer **Hat-Beziehung** zwischen den Klassen: Die Klasse „Bild“ hat Objekte der Klasse „Kreis“.

7.2 − Initialisierung des Attributs

Kommen wir wieder zurück zum Quelltext, den Du verändern sollst. Mit der Deklaration des Objektes „kreis1“ ist es noch nicht getan. Die Deklaration eines Objektes ist sozusagen die Vorbereitung: Es wird Speicherplatz reserviert, damit überhaupt erst ein Objekt angelegt werden kann. Das Objekt selbst existiert aber noch nicht, wenn Du es deklariert hast. Dazu ist ein weiterer Schritt erforderlich, nämlich die Initialisierung des Objektes. Die Initialisierung von Objekten erfolgt meistens im Konstruktor derjenigen Klasse, die das Objekt benötigt:

```
   public Bild() {
      kreis1 = new Kreis(0, 0, "lila");
   }
```

Die Initialisierung besteht aus drei Teilen. Erst kommt der Bezeichner des neuen Objektes, hier also „kreis1“. Dann kommt der Befehl new. Jetzt weiß der Computer bzw. das Java-Programm, dass ein neues Objekt „kreis1“ erzeugt werden soll. Der dritte Teil der Initialisierung besteht in dem Aufruf des Konstruktors des Objektes. Da „kreis1“ ein Objekt der Klasse „Kreis“ ist, muss deshalb der Konstruktor der Klasse „Kreis“ aufgerufen werden. Erinnerst Du Dich noch, wie Du „per Hand“ ein Objekt der Klasse „Kreis“ erzeugt hast? Richtig: Du hast mit der rechten Maustaste auf die Klasse „Kreis“ geklickt und dann den Befehl „new Kreis()“ ausgewählt. Anschließend wurdest Du gefragt, wie das neue Objekt heißen soll und welche Position und welche Farbe der neue Kreis haben soll. Genau das Gleiche haben wir jetzt in den Konstruktor der Klasse „Bild“ hineinprogrammiert.

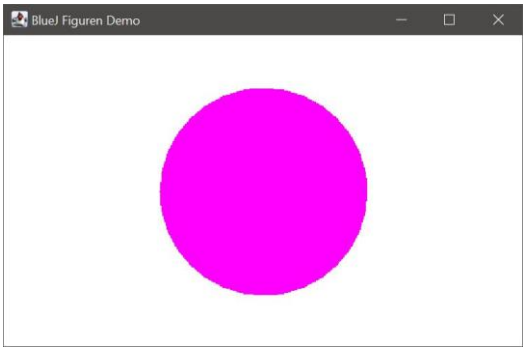
7.3 − Der erste richtige Befehl

Die dritte Stelle, die Du ergänzen musst, ist die Methode bildMalen(). Die Klasse „Bild“ hat jetzt ein Objekt „kreis1“. Keine Angst, Du musst jetzt keine komplizierten Java-Befehle zum Zeichnen von Kreisen lernen, sondern kannst als Programmierer einfach die bereits vorhandenen Methoden der Klasse „Kreis“ aufrufen:

```
   public void bildMalen() {
      kreis1.groesseAendernZentrieren(200);
   }
```

Die Methode bildMalen() der Klasse Bild muss selbst kaum etwas machen, sie ruft lediglich die Methode „groesseAendernZentrieren(int neuerDurchmesser)“ des Objektes „kreis1“ auf und delegiert damit die Hauptarbeit an das Objekt.

Schritt 8 − Das Bild wird erstellt



Erzeuge jetzt − nachdem Du also den Quelltext wie oben angegeben ergänzt hast − ein Objekt „bild1“ der Klasse „Bild“.

Klicke das Objekt mit der rechten Maustaste an und rufe dann die Methode „bildMalen()“ auf.

Wenn Du alles richtig gemacht hast, müsste jetzt ein Fenster mit einem zentrierten lila Kreis mit Durchmesser 200 erscheinen.

Schritt 9 − Weitere Kreis-Objekte

Es ist − ehrlich gesagt − noch nicht allzu beeindruckend, was wir bisher geschafft haben.

Wir wollen jetzt den Quelltext der Klasse „Bild“ um die weiteren Kreis-Objekte ergänzen, um das Kreise-Bild mit den *fünf* Kreisen zu erzeugen. Was ist dazu alles notwendig?

9.1 − Weitere Kreis-Objekte deklarieren

Wir benötigen für das Kreise-Bild vier weitere Objekte der Klasse „Kreis“ innerhalb der Klasse „Bild“.
Also müssen wir zunächst vier weitere Objekte „kreis2“, …, „kreis5“ der Klasse „Kreis“ deklarieren.

```
private Kreis kreis2;
…
private Kreis kreis5;
```

9.2 − Die weiteren Kreis-Objekte initialisieren

Mit der Deklaration ist es noch nicht getan. Die neuen Kreis-Objekte müssen auch initialisiert werden.
Dazu erweitern wir den Konstruktor der Klasse „Bild“ um weitere Zeilen:

```
public Bild() {
   kreis1 = new Kreis(0, 0, "lila");
   kreis2 = new Kreis(0, 0, "blau");
   …
}
```

Die weiteren Kreise werden damit erzeugt, liegen aber noch „übereinander“ in der Ecke oben links.

Schritt 10 − Verändern der weiteren Objekte

Die weiteren Kreise müssen nun noch mit dem passenden Durchmesser versehen werden und in die Mitte der „Leinwand“ verschoben werden.

Dazu müssen wir auf *allen* Kreisen die passende Methode „groesseAendernZentrieren(…)“ aufrufen.

```
public void bildMalen() {
   kreis1.groesseAendernZentrieren(200);
   kreis2.groesseAendernZentrieren(160);
   …
}
```

Wenn Du den Quelltext der Klasse „Bild“ jetzt kompilierst und dann mit der rechten Maustaste ein Objekt der Klasse „Bild“ erzeugst, so führt ein Aufruf der Methode „bildMalen()“ zu dem gewünschten Ergebnis:

Die Kreise-Figur aus fünf Kreisen wird „automatisch“ erzeugt.

Hinweis: Code für die Methode „groesseAendernZentrieren(…)“ der Klasse „Kreis“ (ggf. dort noch einfügen):

```
public void groesseAendernZentrieren(int neuerDurchmesser) {
   loeschen();
   durchmesser = neuerDurchmesser;
   xPosition = 250 - durchmesser/2;
   yPosition = 150 - durchmesser/2;
   zeichnen();
}
```