

1. Ein erstes Beispiel

Wir wollen Zufallsrechtecke mit Zufallsfarben auf den Bildschirm „zaubern“. Dazu bestimmen wir jeweils die obere linke Ecke eines Rechtecks, seine Breite und seine Höhe durch (geeignete) Zufallszahlen. Die erhalten wir durch eine nette Bitte ans *Math-Objekt*, uns solche zu liefern: `Math.random()`; Die erhaltenen Zahlen liegen zwischen Null und Eins. Wir „strecken“ deshalb zuerst den Zahlenbereich mit einem Faktor *m* und erhalten Zufallszahlen zwischen Null und *m*, danach bitten wir erneut das *Math-Objekt*, das Ergebnis zu runden:

```
Math.round(Math.random()*m);
```

Da das Ergebnis nicht unbedingt eine ganze Zahl vom Typ `int` sein muss, erzwingen wir eine entsprechende Typwandlung (type-casting), indem wir die Anweisung (`int`) voranstellen.

```
(int)Math.round(Math.random()*m);
```

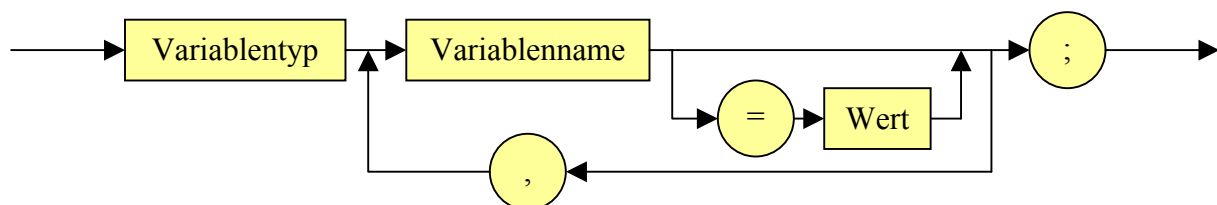
Wenn wir mehrmals Zufallszahlen dieser Art erzeugen wollen, dann stellen wir am besten eine entsprechende Funktion bereit, die solche Zahlen berechnet. Der Rückgabewert der Funktion ist `int`. Mit `return` definieren wir diesen Wert.

```
public int zz(int m)
{
    return (int)Math.round(Math.random()*m);
}
```

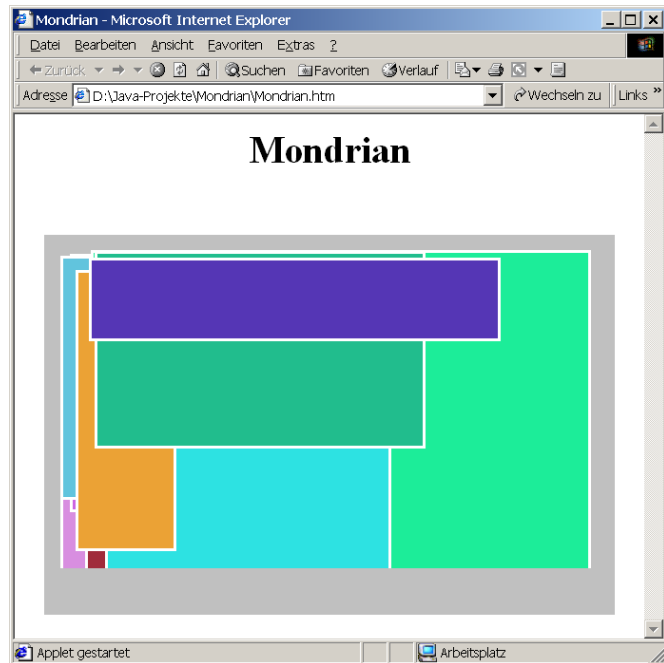
Variablen müssen in Java zusammen mit ihrem Wert vereinbart werden, bevor man sie benutzen kann. Für unsere Rechtecke benötigen wir vier ganze Zahlen *x,y,b* und *h*, die die obere linke Ecke sowie Breite und Höhe des Rechtecks beschreiben. Weil wir noch eine Zufallsfarbe bestimmen wollen, vereinbaren wir diese gleich mit.

```
int x,y,b,h;
Color c;
```

Allgemein muss eine **Variablenvereinbarung** der folgenden Syntax gehorchen:



Als Typen stehen Zahlen (`int`, `double`, `long`, ...), Wahrheitswerte (`boolean`), Zeichenketten (`String`), Farben (`Color`) und weitere Klassen zur Verfügung. Die Vorgabewerte müssen natürlich dem Datentyp der Variablen entsprechen.

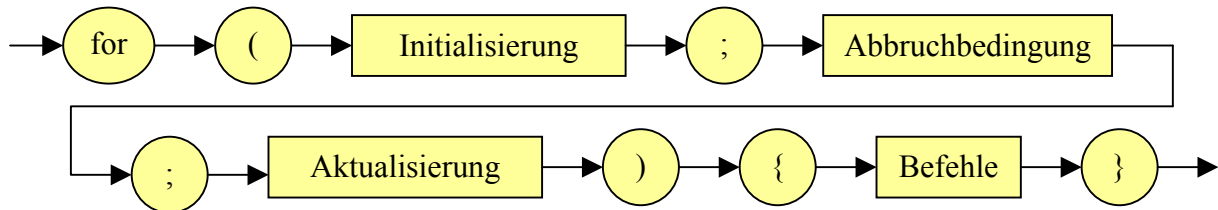


In unserem Programm wollen wir jeweils 10 Zufallsrechtecke zeichnen. Da hierfür immer die gleichen Vorgänge (Zufallszahlen und Farben bestimmen, Rechtecke zeichnen, ...) erforderlich sind, packen wir diese in eine **Zählschleife**, die dafür sorgt, dass der Schleifenkörper so oft ausgeführt wird, wie der Schleifenkopf angibt.

```
for (int i=0;i<max;i++)
{
    ...
}
```

Die Zählvariable i erhält anfangs den Wert 0. Die Schleife wird ausgeführt, solange i kleiner als max ist (Abbruchbedingung). Nach jedem Durchlauf wird i um 1 erhöht.

Allgemein muss eine **Zählschleife** der folgenden Syntax gehorchen:



- Im **Initialisierungsteil** werden Variable vereinbart und Anfangswerte gesetzt.
- In der **Abbruchbedingung** wird vereinbart, unter welchen Bedingungen die Schleife weiter ausgeführt werden soll.
- Im **Aktualisierungsteil** werden die Werte der Variablen verändert.

Innerhalb der Schleife basteln wir in unserem Programm aus den Farbwerten Rot, Grün und Blau im RGB-System eine Zufallsfarbe zusammen, mit der wir die Rechteckfläche zeichnen. Dann wird noch ein etwas breiterer weißer Rand erzeugt.

```
import java.awt.*;
import java.applet.*;

public class Mondrian extends Applet
{
    int max=10; //Anzahl der Rechtecke

    public int zz(int m)
    {
        return (int)Math.round(Math.random()*m);
    }

    public void paint(Graphics g)
    {
        int x,y,b,h;
        Color c;
        for (int i=0;i<max;i++)
        {
            x = zz(50)+10;
            y = zz(30)+10;
            b = zz(600-x-20)+10;
            h = zz(400-y-20)+10;
            c = new Color(zz(255),zz(255),zz(255));
            g.setColor(c);
            g.fillRect(x,y,b,h);
            g.setColor(Color.white);
            g.drawRect(x,y,b,h);
            g.drawRect(x+1,y+1,b-2,h-2);
            g.drawRect(x+2,y+2,b-4,h-4);
        }
    }
}
```

eine Variable

eine selbst geschriebene Funktion, die aus der Mathematikbibliothek „Math“ die Methoden „round()“ und „random()“ benutzt.

Farbe aus Zufallszahlen zusammenbasteln

Rechteck mit Rand zeichnen

2. Aufgaben

1. a: Erzeugen Sie Zufallsrechtecke auf dem Bildschirm. Sie können das Programm mehrfach ausführen, indem Sie „Aktualisieren“ im Browser anklicken.
 - b: Sorgen Sie dafür, dass die Rechtecke farbig werde.
 - c: Wählen Sie die Zufallsfarben „geschmackvoll“, also nicht beliebig, sondern gezielt. Sie könnten z.B. Farben überwiegend aus dem Bereich grün-blau wählen, mit gelegentlichen gelben „Ausrutschern“.

2. a: Erzeugen Sie gefüllte Zufallsrechtecke auf dem Bildschirm. Informieren Sie sich dazu über die Methode *SetXORMode()* des *Graphics*-Objekts.
 - b: Sorgen Sie dafür, dass die Rechtecke farbig werden.
 - c: Wählen Sie wiederum „geschmackvolle“ Zufallsfarben (s. Aufgabe 1.c).
 - d: Entsteht in Ihrem Programm eine „Gewichtung“ der Zufallsrechtecke wie in den nebenstehenden Beispielen? („Kleinere Strukturen unten-rechts.“) Wie kommt das? Kann man den Effekt gezielt erzeugen? Kann man andere Gewichtungen erzielen?

3. a: Informieren Sie sich im Hilfesystem über die Methode *fillArc()* des *Graphics*-Objekts. Erzeugen Sie eine Grafik ähnlich der nebenstehenden mit Hilfe dieser Methode.
 - b: Wechseln Sie bei den Figuren (zufallsgesteuert?) zwischen Rechtecken, Ellipsen, Polygonen, ...
 - c: Versuchen Sie es einmal mit „Zufallstexten“, also Texten, die zufällig über den Bildschirm verteilt werden. Informieren Sie sich im Hilfesystem über die Methode *drawString()* des *Graphics*-Objekts.

