```
1  import sum.kern.*;
2
3  public class Kugel
4  {    private Buntstift hatStift;
5       private Bildschirm kenntBildschirm;
6       // Attribute
7       private int zGroesse;
8       private double zGeschwindigkeit;
9
10      // Konstruktor
11      public Kugel(int pH, int pV, int pGroesse, double pGeschwindigkeit,
12                   int pFarbe, int pRichtung, Bildschirm pBildschirm)
13      {    hatStift = new Buntstift();
14           hatStift.bewegeBis(pH, pV);
15           hatStift.setzeFarbe(pFarbe);
16           hatStift.setzeFuellmuster(Muster.GEFUELLT);
17           zGroesse = pGroesse;
18           zGeschwindigkeit = pGeschwindigkeit;
19           this.setzeRichtung(pRichtung);
20           kenntBildschirm = pBildschirm;
21      }
22
23      // Dienste
24      public void gibFrei()
25      {  hatStift.gibFrei(); }
26
27      public void zeichne()
28      {  hatStift.zeichneKreis(zGroesse);     }
29
30      public void loesche()
31      {  hatStift.radiere();
32         this.zeichne();
33         hatStift.normal();
34      }
35
36      public void bewege()
37      {  this.loesche();
38         hatStift.bewegeUm(zGeschwindigkeit);
39         this.zeichne();
40         if(this.amLinkenRand() || this.amRechtenRand())
41              this.setzeRichtung(180 - hatStift.winkel());
42         if(this.amOberenRand() || this.amUnterenRand())
43              this.setzeRichtung(360 - hatStift.winkel());
44      }
45
46      public double hPosition()
47      {  return hatStift.hPosition();     }
48
49      public double vPosition()
50      {  return hatStift.vPosition();     }
51
52      public void setzeRichtung(double pRichtung)
53      {  hatStift.dreheBis(pRichtung);     }
54
55      private boolean amLinkenRand()
56      {  return this.hPosition() < zGroesse;     }
57
58      private boolean amRechtenRand()
59      {  return this.hPosition() >(kenntBildschirm.breite() - zGroesse);     }
60
61      private boolean amOberenRand()
62      {  return this.vPosition() < zGroesse;     }
63
```

```
64      private boolean amUnterenRand()
65      {  return this.vPosition() >(kenntBildschirm.hoehe() - zGroesse);     }
66
67      public void setzeGroesse(int pGroesse)
68      {  zGroesse = pGroesse;     }
69
70      public int groesse()
71      {  return zGroesse;     }
72
73      public void setzeGeschwindigkeit(double pGeschwindigkeit)
74      {  zGeschwindigkeit = pGeschwindigkeit;     }
75
76      public double geschwindigkeit()
77      {  return zGeschwindigkeit;  }
78  } // Ende von Kugel
```

```
1   import sum.kern.*;
2
3   public class Hauptprogramm
4   {
5       // Objekte
6       private Bildschirm hatBildschirm;
7       private Maus hatMaus;
8       private Kugel hatKugel1, hatKugel2, hatKugel3;
9       // Achtung: Hier ist kein Stift mehr!
10
11      // Konstruktor
12      public Hauptprogramm()
13      {   hatBildschirm = new Bildschirm(800, 600);
14          hatMaus = new Maus();
15          hatKugel1 = new Kugel(130, 100, 5,  0.20, Farbe.ROT,   30, hatBildschirm);
16          hatKugel2 = new Kugel(30,  200, 15, 0.25, Farbe.GRUEN, 60, hatBildschirm);
17          hatKugel3 = new Kugel(330, 300, 25, 0.30, Farbe.BLAU, -50, hatBildschirm);
18      }
19
20      // Dienste
21      public void fuehreAus()
22      { do
23        {     hatKugel1.bewege();
24              hatKugel2.bewege();
25              hatKugel3.bewege();
26        } while (!hatMaus.doppelKlick());
27
28        // Aufraeumen
29        hatKugel3.gibFrei();
30        hatKugel2.gibFrei();
31        hatKugel1.gibFrei();
32        hatMaus.gibFrei();
33        hatBildschirm.gibFrei();
34      }
35  }
```

1. Unten dargestellt sind die beiden zugehörigen **Klassendiagramme** in UML. Das Format ist etwas anders als in unserem Buch. Statt `!` für einen Auftrag und `?` für eine Anfrage ist hier der Typ des Rückgabewertes (z.B. `boolean`) angegeben. Übersetzt in Java ergibt sich z.B. für die Zeile

   ```
   + hPosition(): double
   ```

   folgender Methodenkopf:

   ```
   public double hPosition()
   ```

   Desweiteren sind hier immer die Klammern angegeben, auch wenn die Parameterliste leer ist.

   In manchen Büchern findet man auch die Form

   ```
   + double hPosition()
   ```

   Es sollte einem klar sein, dass ein Klassendiagramm ein Werkzeug **zum Entwurf** einer Klasse ist.