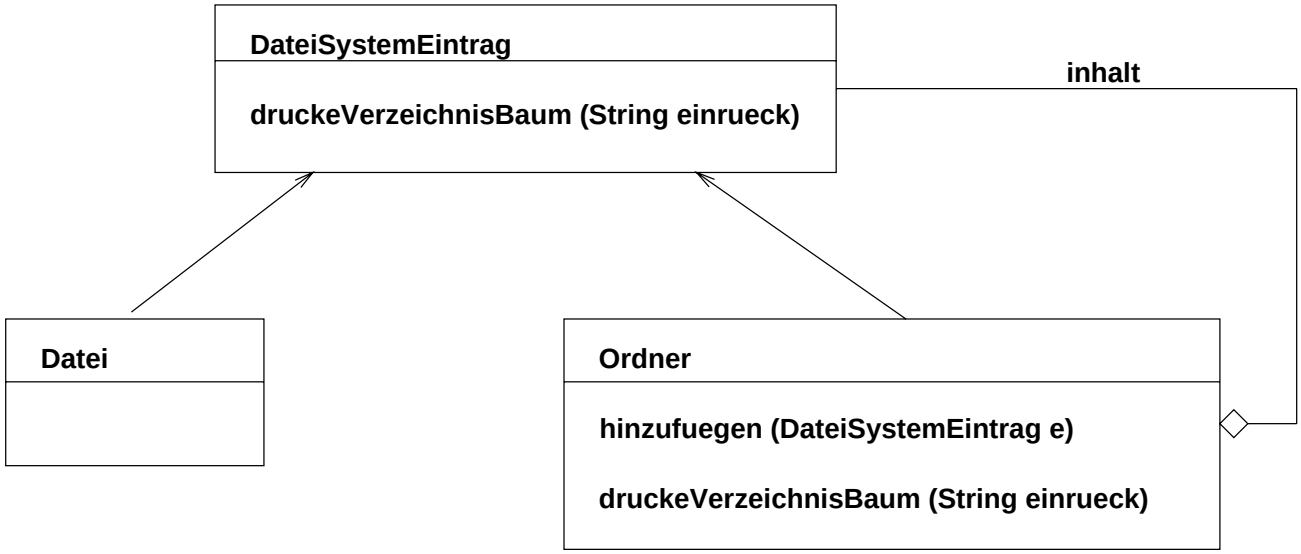
Objektorientierte Programmierung WS 2013/2014 - Lösung 4

Prof. Dr. U. Kastens
Institut für Informatik, Fakultät für Elektrotechnik, Informatik und Mathematik, Universität Paderborn

Lösung zu Aufgabe 1

Aus der Aufgabenstellung und dem vorgegebenen Rahmen ergibt sich folgende Struktur:



Die Implementierung könnte wie folgt aussehen:

Die Klasse `DateiSystemEintrag`

```
class DateiSystemEintrag
{  private String name;
   public DateiSystemEintrag (String name)
   { this.name = name;
   }
   public void druckeVerzeichnisBaum (String einrueck)
   { System.out.println(einrueck + name);
   }
}
```

Die Klasse `Datei`

```
class Datei extends DateiSystemEintrag
{  public Datei (String name)
   { super(name);
   }
}
```

Die Klasse `Ordner`

```
import java.util.ArrayList;
class Ordner extends DateiSystemEintrag {
   ArrayList<DateiSystemEintrag> inhalt;
   public Ordner (String name) {
      super(name);
      inhalt = new ArrayList<DateiSystemEintrag>();
   }
   public void hinzufuegen (DateiSystemEintrag e) {
      inhalt.add(e);
   }
   public void druckeVerzeichnisBaum (String einrueck) {
      super.druckeVerzeichnisBaum(einrueck);
      for(DateiSystemEintrag dse : inhalt) {
```

```
            dse.druckeVerzeichnisBaum(einrueck + "   ");
        }
    }
}
```

Hier sind die Klassen zum Herunterladen: `blatt4/verzbaumloes`.

Lösung zu Aufgabe 2

a) Die Aufrufketten für die beiden Aufrufe

```
ball1.moveSpielBall();
ball2.moveSpielBall();
```

sind

```
ball1.moveSpielBall() == Spielball.moveSpielBall()
                      --> Maus.moveSpielBall()
                      --> Spielball.move()
                      --> Ball.move

ball2.moveSpielBall() == Spielball.moveSpielBall()
                      --> Katze.moveSpielBall()
                      --> Spielball.move()
                      --> Ball.move
```

b) Das Wechseln der Rollen von Katze und Maus, nachdem die Maus vollständig aufgefressen ist, kann man durch das Einfügen folgenden Codes in die `paint()`-Methode von `KatzUndMaus` erreichen:

```
if (ball1.x() == ball2.x() && ball1.y() == ball2.y())
{      // die Maus ist weg
   if (ball1.getRolle() instanceof Maus)
   {   // ball1 war die Maus, wird jetzt zur Katze
      ball1.setRolle(new Katze(ball1, ball2));
      // ball2 also zur neuen Maus
      ball2.setRolle(new Maus(ball2, this));
   }
   else
   {   // und andersrum: ball1 war also die Katze
      ball1.setRolle(new Maus(ball1, this));
      // ball2 wird jetzt Katze
      ball2.setRolle(new Katze(ball2, ball1));
   }

   // jeder in seine Ecke:
   ball1.moveTo(20 - RADIUS, FRAMEHEIGHT - 20 - RADIUS);
   ball2.moveTo(FRAMEWIDTH - 20 - RADIUS, 40 - RADIUS);
}
```

Anstatt die `instanceof`-Methode zu verwenden, könnten die Interfaces/Klassen `SpielRolle`, `Katze` und `Maus` auch eine Methode zur Identifikation (etwa `boolean ichBinMaus()`) zur Verfügung stellen.

Die vollständige Lösung findet sich in `blatt4/LoesKatzUndMaus`.

Lösung zu Aufgabe 3

a) Die Methode `countMe(String s)` erhöht den Zähler für den Eintrag `s` um 1, bzw. setzt ihn auf 1 beim ersten Auftreten von `s`:

```
public void countMe(String s) {
   Integer freq = get(s);
   put(s, (freq == null) ? 1 : freq + 1);
}
```

b) Die Variante von `WordCount`, die `TreeMap`-Komponente verwendet: `WordCount2.java`.