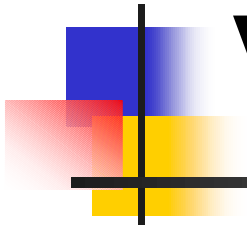
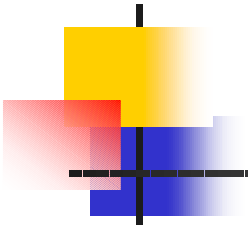


Verteidigung Großer Beleg

Die GoF-Entwurfsmuster in Java

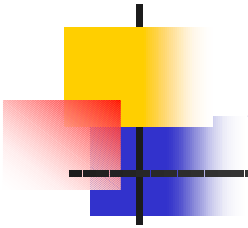
Corinna Herrmann

ch17@inf.tu-dresden.de



Gliederung

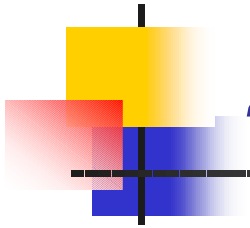
1. Aufgabenstellung
2. Entwurfsmuster
3. Verwandte Arbeiten
4. Beispiele:
   - 4.1. Adapter
   - 4.2. Flyweight
5. Zusammenfassung & Ausblick
6. Literatur



1. Aufgabenstellung

- „GoF“: Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides

  → Ausgangspunkt der „Musterbewegung“
- Entwicklung von Beispielprogrammen für die GoF-Entwurfsmuster
- Entwurf eines Template zur Beschreibung der Muster



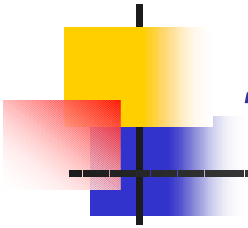
2. Entwurfsmuster

- Einsatz im Entwurfsprozeß
- Bewährte Lösung für eine immer wiederkehrendes Problem
- Unterstützt die Wiederverwendung von Lösungen
- Dokumentiert existierende und erprobte Entwurfserfahrungen
- Einteilung: Erzeugungs-, Struktur- und Verhaltensmuster
- **Template:** Allgemeines, Struktur, Beispiel, Besonderheiten

3. Verwandte Arbeiten

	[Gam96]	[Coo98]	[Gra98]	[Eck01]	[Her01]
Inhalt					
Anzahl	23	23	41	16	23
Verständlichkeit	x	x	x	o	
Beispiele					
Modellierungssprache	OMT	OMT	UML	UML	UML
Programmiersprache	C++	Java	Java	Java	Java
Sinnvolle Beispiele	o	o	o	o	
Verständlichkeit	o	x	o	o	

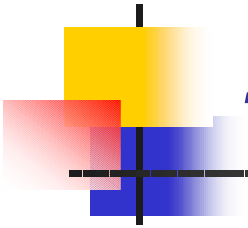
x ja bzw. vorhanden; **o** teilweise; **-** nein



4. Beispiele

- Ziel: sinnvoll, verständlich, einfach
  → Wiederverwendung & Ausbau
  → Konsole & Oberfläche
  → Nutzung vorhandener Vorschläge

- Zeichenprogramm zur Darstellung von Kreisen, (Rechtecken und Smilies)
- Realisierung: **Adapter**, Decorator, **Flyweight** und Visitor

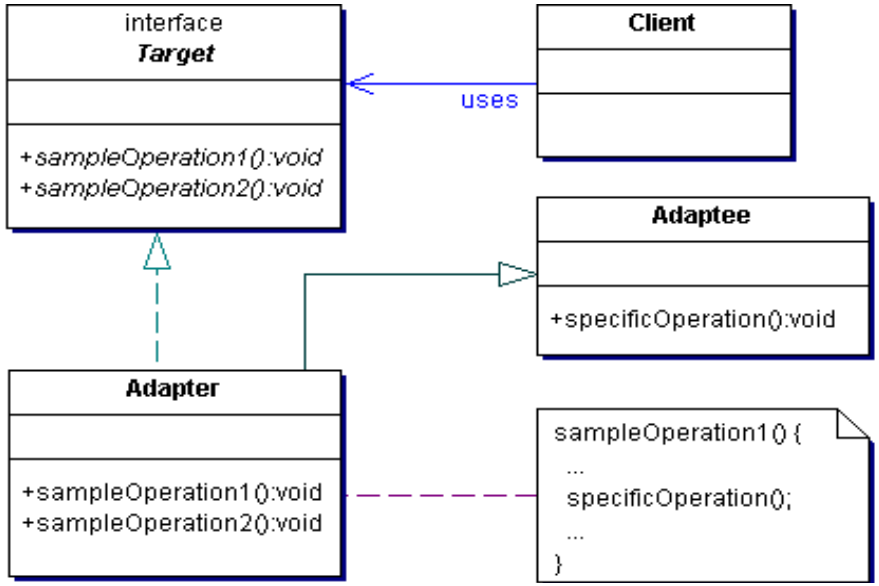


4.1. Adapter

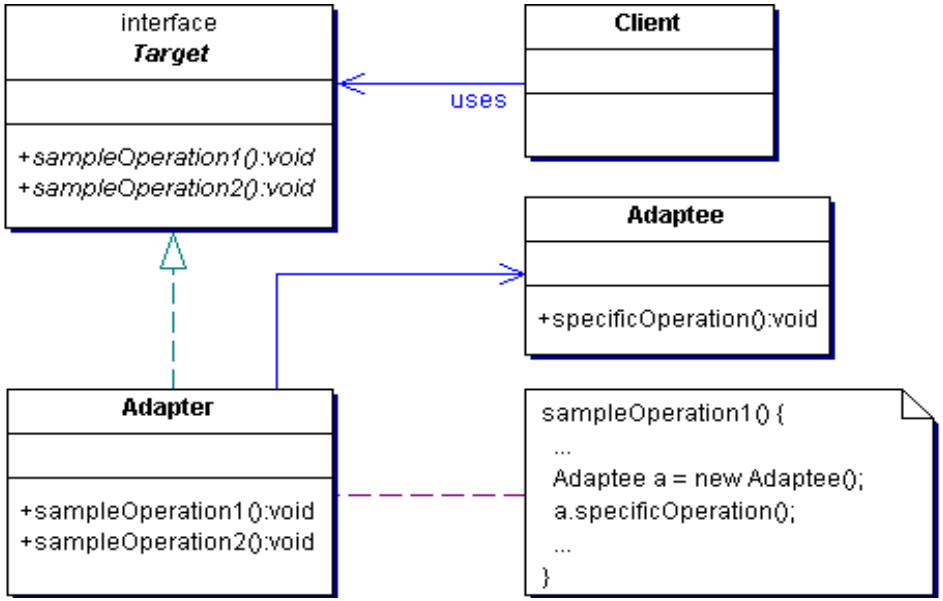
- Strukturmuster
- **Ausgangspunkt:** Vorhandene Klassen bzw. Klassenbibliotheken
- **Ziel:** Einbindung in die eigene Programmstruktur
- **Lösung:** Anpassung der vorhandenen Schnittstellen
- Adaption durch:
  - Klassenadapter
  - Objektadapter

4.1. Adapter (2)

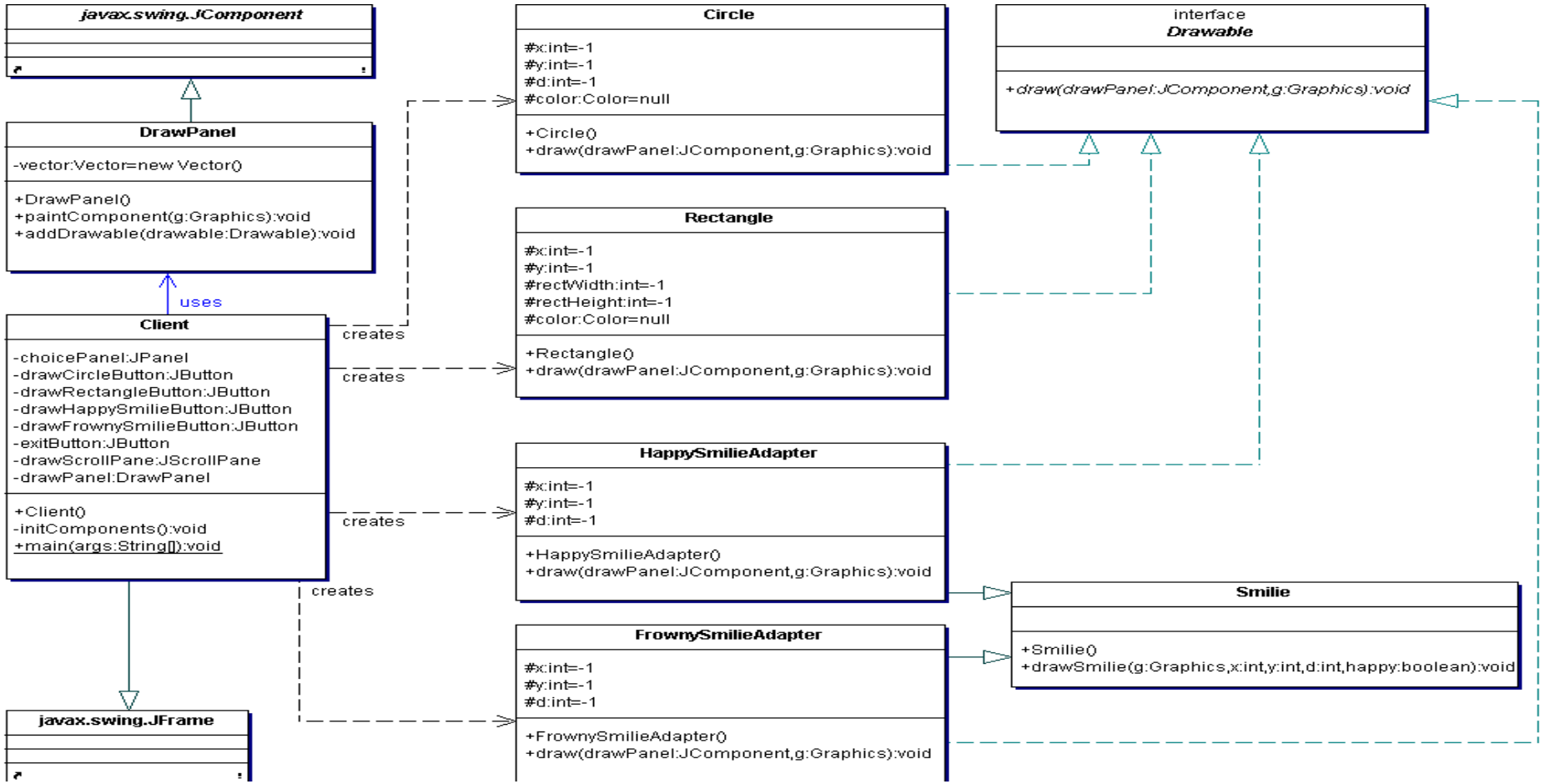
Klassenadapter



Objektadapter



4.1. Adapter (3)


javax.swing.JComponent
DrawPanel
-vector:Vector=new Vector()
+DrawPanel()
+paintComponent(g:Graphics):void
+addDrawable(drawable:Drawable):void
uses
Client
-choicePanel:JPanel
-drawCircleButton:JButton
-drawRectangleButton:JButton
-drawHappySmilieButton:JButton
-drawFrownySmilieButton:JButton
-exitButton:JButton
-drawScrollPane:JScrollPane
-drawPanel:DrawPanel
+Client()
-initComponents():void
+main(args:String[]):void
javax.swing.JFrame
creates
creates
creates
creates
Circle
#x:int=-1
#y:int=-1
#d:int=-1
#color:Color=null
+Circle()
+draw(drawPanel:JComponent,g:Graphics):void
Rectangle
#x:int=-1
#y:int=-1
#rectWidth:int=-1
#rectHeight:int=-1
#color:Color=null
+Rectangle()
+draw(drawPanel:JComponent,g:Graphics):void
HappySmilieAdapter
#x:int=-1
#y:int=-1
#d:int=-1
+HappySmilieAdapter()
+draw(drawPanel:JComponent,g:Graphics):void
FrownySmilieAdapter
#x:int=-1
#y:int=-1
#d:int=-1
+FrownySmilieAdapter()
+draw(drawPanel:JComponent,g:Graphics):void
interface
Drawable
+draw(drawPanel:JComponent,g:Graphics):void
Smilie
+Smilie()
+drawSmilie(g:Graphics,x:int,y:int,d:int,happy:boolean):void

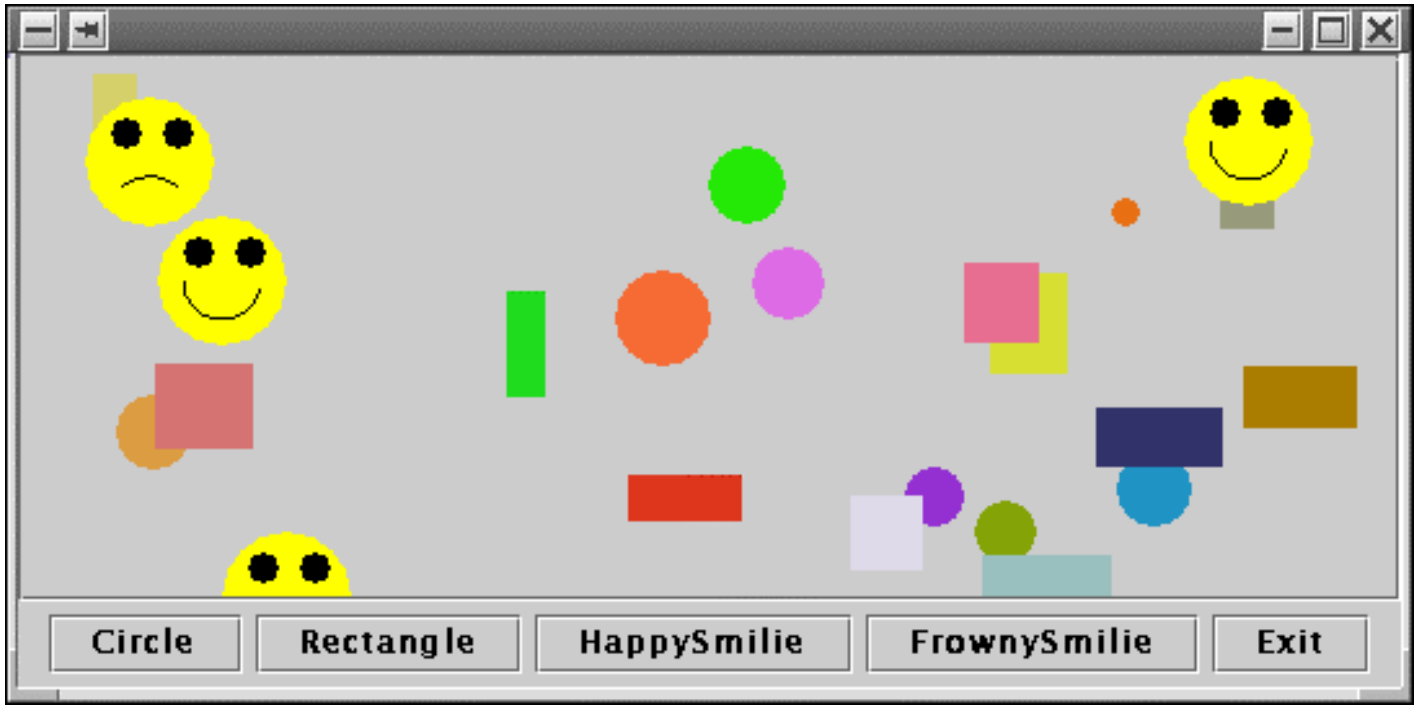
4.1. Adapter (4)

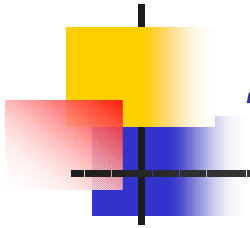
```
public class Smilie {
    public void drawSmilie(Graphics g, int x, int y, int d, boolean happy) {
        ...
        if (happy == true) {
            g.drawArc(x+10,y+10,30,30,180,165);
        } else {
            g.drawArc(x+10,y+30,30,30,45,90);
        }
    }
}

public class HappySmilieAdapter extends Smilie implements Drawable {
    …
    public void draw(JComponent drawPanel, Graphics g) {
        …
```

4.1. Adapter (5)

```
        drawSmilie(g,x,y,d, true);
    }
}

public class Client extends JFrame {
    ...
    private void initComponents () {
        ...
        drawHappySmilieButton.addActionListener (new ActionListener () {
            public void actionPerformed (ActionEvent evt) {
                HappySmilieAdapter happy = new  HappySmilieAdapter();
                drawPanel.addDrawable(happy);
            }
        });
    }
}
```

4.1. Adapter (6)

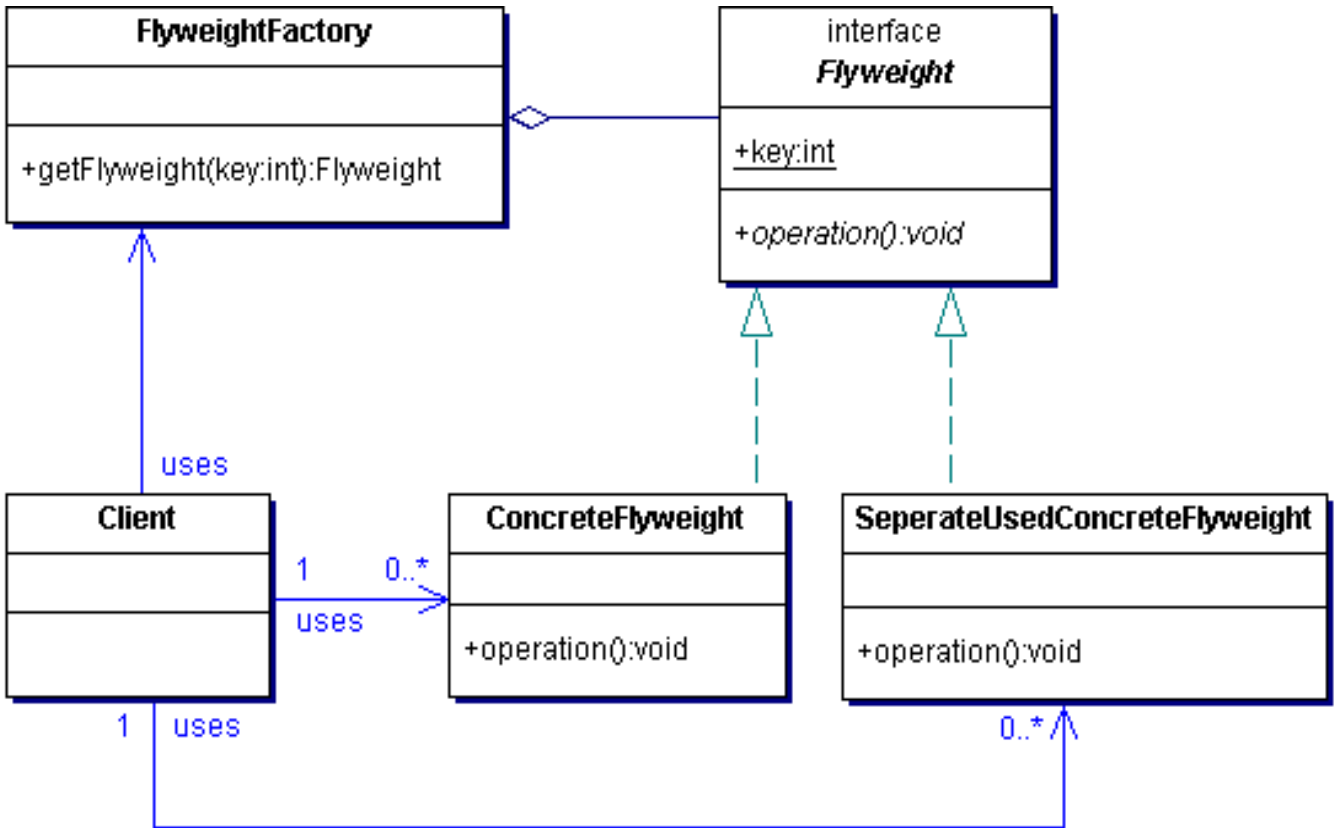




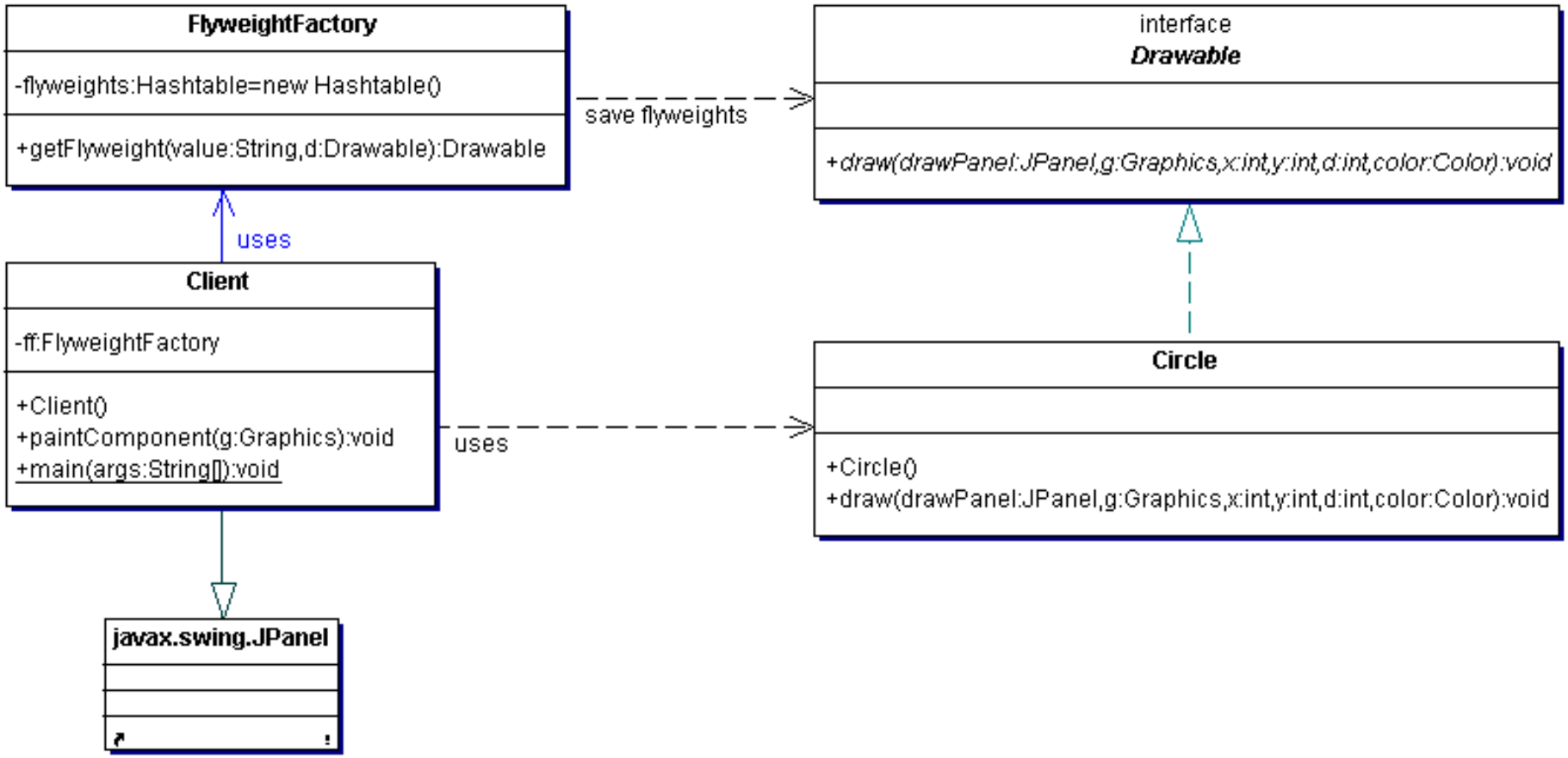
4.2. Flyweight

- Strukturmuster
- **Ausgangspunkt:** Erzeugung einer großen Menge ähnlicher Objekte
- **Ziel:** Einsparung von Speicherplatz
- **Lösung:** Implementierung einer Fliegengewicht-Klasse zur Variation der Eigenschaften
- Erzeugung genau einer Instanz

4.2. Flyweight (2)



4.2. Flyweight (3)

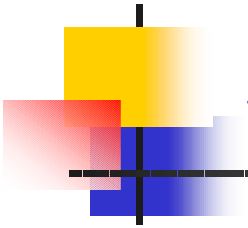


4.2. Flyweight (4)

```
public class Circle implements Drawable {
    public void draw(JPanel drawPanel, Graphics g, int x, int y, int d, Color color) {
        …
    }
}

public class FlyweightFactory {
    private Hashtable flyweights = new Hashtable();
    public Drawable getFlyweight(String value, Drawable d) {
        if (flyweights.get(value)== null) {
            flyweights.put(value, d);
            return d;
        } else {
          return ((Drawable)flyweights.get(value));
        }
     }
}
```

4.2. Flyweight (5)

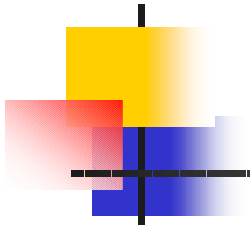
```
public class Client extends JPanel {
    private FlyweightFactory ff;
    public Client() {
        ff = new FlyweightFactory();
    }
    public void paintComponent(Graphics g) {
        Circle circle = (Circle)ff.getFlyweight("circle", new Circle());
        circle.draw(this, g, 50, 50, 100, new Color(200, 250, 250));
        circle.draw(this, g, 100, 100, 100, new Color(100, 100, 150));
        circle.draw(this, g, 150, 150, 100, new Color(250, 200, 200));
    }
    public static void main (String[] args) {
        …
    }
}
```



5. Zusammenfassung & Ausblick

- 23 Programme → 13 eigene Ideen → 126 Seiten → 1140 Dateien
- Javadoc & Schriftteil zur Dokumentation

- Weitere Möglichkeiten:
  - Konsolen- und GUI-Programm für jedes Entwurfsmuster
  - Ausführlichere Beschreibungen
  - Erweiterung des Katalogs



6. Literatur

- **[Gam96]** Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides: „Entwurfsmuster Elemente wiederverwendbarer Software“, Addison-Wesley Verlag, München 1996
- **[Coo98]** James W.Cooper: „The Design Patterns Java Companion“, IBM Thomas J. Watson Research Center, http://www.patterndepot.com/put/8/JavaPatterns.htm, 1998
- **[Gra98]** Mark Grand: „Patterns in Java Volume 1, A Catalog of Reusable Design Patterns Illustrated with UML“, Wiley Computer Publishing, New York 1998
- **[Eck01]** Bruce Eckel: „Thinking in Patterns with Java, Revision 0.5a, http://64.78.49.204, 2001