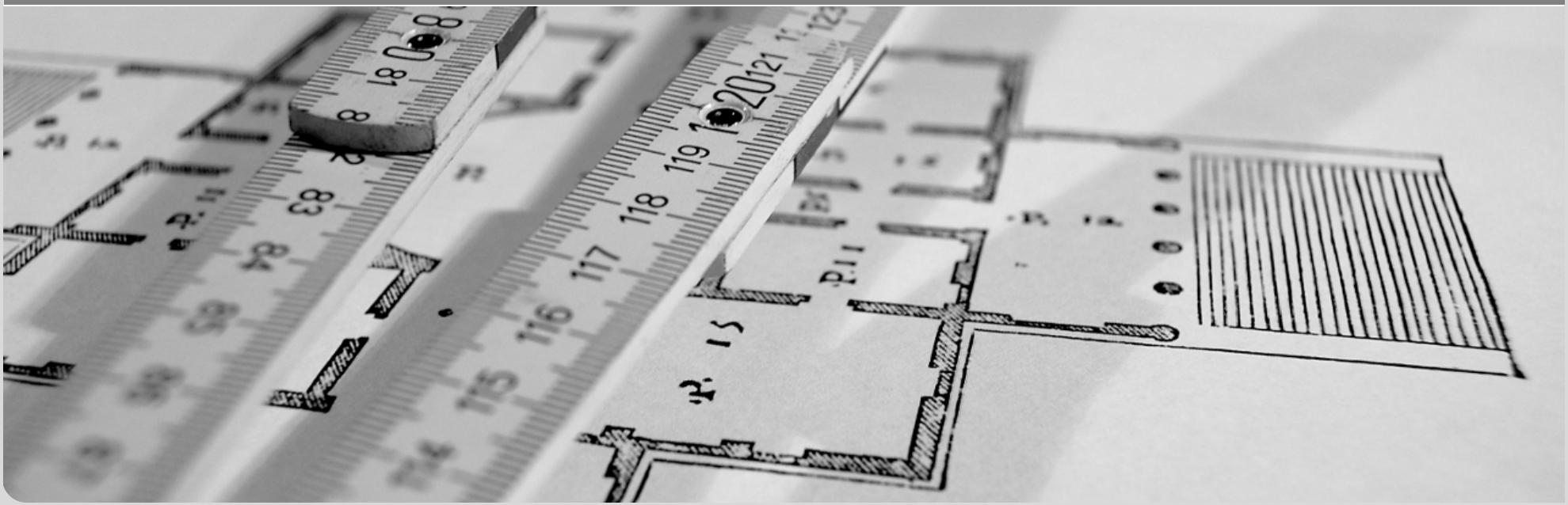
Karlsruher Institut für Technologie

# Tutorium 02: UML in Aktion

Softwaretechnik im SS 2011, Tutorien 4 + 11 + 17

Christian Jülg | 19. Mai 2011

CHAIR FOR SOFTWARE DESIGN AND QUALITY



# Was machen wir heute?

1. Altes Übungsblatt
   - Altes Übungsblatt
   - Zum Aufwärmen ...
   - Swing
   - Eclipse
2. UML
   - Zustandsdiagramm
   - Sequenzdiagramm
3. Ende
   - Tipps zum nächsten Übungsblatt

# Altes Übungsblatt

## Aufgabe 1: Klassendiagramm

- Von Hand Zeichnen!
- Attributname : Attributtyp, nicht umgekehrt
- Vererbungspfeile sind nicht ausgemalt...

## Aufgabe 2: Anwendungsfalldiagramm

- Von Hand Zeichnen!
- Alles muss zum Endknoten verbunden werden!
- Die meissten haben nur die Hälfte der Aufgabe gelöst!

## Aufgabe 3 Floyd-Steinberg-Algorithmus

- Checkstyle: “Utility classes should not have a public or default constructor”
- Lösung: **private** Klassenname() {}

## Aufgabe 3 Floyd-Steinberg-Algorithmus

- Checkstyle: “Utility classes should not have a public or default constructor”
- Lösung: **private** Klassenname() {}

# Wahr oder falsch?

- Das einzige Ziel der Softwaretechnik ist, die Kosten der Erstellung von Software möglichst weitgehend zu senken.
- UML-Anwendungsfalldiagramme werden während der Planungsphase verwendet, um das von außen sichtbare Verhalten des Systems darzustellen.
- Komposition ist eine strengere Aggregation, bei der die Teile keine Daseinsberechtigung ohne das Ganze haben.
- Grundsatz der Vererbung: Mache eine Klasse A erst dann zu einer Unterklasse einer Klasse B, wenn sicher ist, dass jede Instanz von B auch als Instanz von A gesehen werden kann.
- Kontravariante Eingabe-Parameter erfüllen das Substitutionsprinzip.
- Ein Pflichtenheft spezifiziert die Anforderungen an eine Software in eindeutiger Weise, so dass sie implementiert werden können

# Wahr oder falsch?

- Das einzige Ziel der Softwaretechnik ist, die Kosten der Erstellung von Software möglichst weitgehend zu senken.
- UML-Anwendungsfalldiagramme werden während der Planungsphase verwendet, um das von außen sichtbare Verhalten des Systems darzustellen.
- Komposition ist eine strengere Aggregation, bei der die Teile keine Daseinsberechtigung ohne das Ganze haben.
- Grundsatz der Vererbung: Mache eine Klasse A erst dann zu einer Unterklasse einer Klasse B, wenn sicher ist, dass jede Instanz von B auch als Instanz von A gesehen werden kann.
- Kontravariante Eingabe-Parameter erfüllen das Substitutionsprinzip.
- Ein Pflichtenheft spezifiziert die Anforderungen an eine Software in eindeutiger Weise, so dass sie implementiert werden können

# Wahr oder falsch?

- Das einzige Ziel der Softwaretechnik ist, die Kosten der Erstellung von Software möglichst weitgehend zu senken.
- UML-Anwendungsfalldiagramme werden während der Planungsphase verwendet, um das von außen sichtbare Verhalten des Systems darzustellen.
- Komposition ist eine strengere Aggregation, bei der die Teile keine Daseinsberechtigung ohne das Ganze haben.
- Grundsatz der Vererbung: Mache eine Klasse A erst dann zu einer Unterklasse einer Klasse B, wenn sicher ist, dass jede Instanz von B auch als Instanz von A gesehen werden kann.
- Kontravariante Eingabe-Parameter erfüllen das Substitutionsprinzip.
- Ein Pflichtenheft spezifiziert die Anforderungen an eine Software in eindeutiger Weise, so dass sie implementiert werden können

# Wahr oder falsch?

- Das einzige Ziel der Softwaretechnik ist, die Kosten der Erstellung von Software möglichst weitgehend zu senken.
- UML-Anwendungsfalldiagramme werden während der Planungsphase verwendet, um das von außen sichtbare Verhalten des Systems darzustellen.
- Komposition ist eine strengere Aggregation, bei der die Teile keine Daseinsberechtigung ohne das Ganze haben.
- Grundsatz der Vererbung: Mache eine Klasse A erst dann zu einer Unterklasse einer Klasse B, wenn sicher ist, dass jede Instanz von B auch als Instanz von A gesehen werden kann.
- Kontravariante Eingabe-Parameter erfüllen das Substitutionsprinzip.
- Ein Pflichtenheft spezifiziert die Anforderungen an eine Software in eindeutiger Weise, so dass sie implementiert werden können

# Wahr oder falsch?

- Das einzige Ziel der Softwaretechnik ist, die Kosten der Erstellung von Software möglichst weitgehend zu senken.
- UML-Anwendungsfalldiagramme werden während der Planungsphase verwendet, um das von außen sichtbare Verhalten des Systems darzustellen.
- Komposition ist eine strengere Aggregation, bei der die Teile keine Daseinsberechtigung ohne das Ganze haben.
- Grundsatz der Vererbung: Mache eine Klasse A erst dann zu einer Unterklasse einer Klasse B, wenn sicher ist, dass jede Instanz von B auch als Instanz von A gesehen werden kann.
- Kontravariante Eingabe-Parameter erfüllen das Substitutionsprinzip.
- Ein Pflichtenheft spezifiziert die Anforderungen an eine Software in eindeutiger Weise, so dass sie implementiert werden können

# Wahr oder falsch?

- Das einzige Ziel der Softwaretechnik ist, die Kosten der Erstellung von Software möglichst weitgehend zu senken.
- UML-Anwendungsfalldiagramme werden während der Planungsphase verwendet, um das von außen sichtbare Verhalten des Systems darzustellen.
- Komposition ist eine strengere Aggregation, bei der die Teile keine Daseinsberechtigung ohne das Ganze haben.
- Grundsatz der Vererbung: Mache eine Klasse A erst dann zu einer Unterklasse einer Klasse B, wenn sicher ist, dass jede Instanz von B auch als Instanz von A gesehen werden kann.
- Kontravariante Eingabe-Parameter erfüllen das Substitutionsprinzip.
- Ein Pflichtenheft spezifiziert die Anforderungen an eine Software in eindeutiger Weise, so dass sie implementiert werden können

# Wahr oder falsch?

- Das einzige Ziel der Softwaretechnik ist, die Kosten der Erstellung von Software möglichst weitgehend zu senken.
- UML-Anwendungsfalldiagramme werden während der Planungsphase verwendet, um das von außen sichtbare Verhalten des Systems darzustellen.
- Komposition ist eine strengere Aggregation, bei der die Teile keine Daseinsberechtigung ohne das Ganze haben.
- Grundsatz der Vererbung: Mache eine Klasse A erst dann zu einer Unterklasse einer Klasse B, wenn sicher ist, dass jede Instanz von B auch als Instanz von A gesehen werden kann.
- Kontravariante Eingabe-Parameter erfüllen das Substitutionsprinzip.
- Ein Pflichtenheft spezifiziert die Anforderungen an eine Software in eindeutiger Weise, so dass sie implementiert werden können

# Swing Tipps

## Layouts

- um Swing Komponenten anzuordnen verwendet man Layouts
- es gibt sehr viele Varianten: `http://download.oracle.com/javase/tutorial/uiswing/layout/visual.html`
- für ÜB3 habt ihr freie Wahl, daher bieten sich `FlowLayout` oder ein vertikales `BoxLayout` an
- für bessere Strukturierung: gruppiert mehrere Komponenten mit einem JPanel
- jeder Swing Container kann sein eigenes Layout haben
- default: JFrame - `BorderLayout`, JPanel - `FlowLayout`)

# Swing Tipps

## Layouts

- um Swing Komponenten anzuordnen verwendet man Layouts
- es gibt sehr viele Varianten: `http://download.oracle.com/javase/tutorial/uiswing/layout/visual.html`
- für ÜB3 habt ihr freie Wahl, daher bieten sich `FlowLayout` oder ein vertikales `BoxLayout` an
- für bessere Strukturierung: gruppiert mehrere Komponenten mit einem JPanel
- jeder Swing Container kann sein eigenes Layout haben
- default: JFrame - `BorderLayout`, JPanel - `FlowLayout`)

# Swing Tipps

## Layouts

- um Swing Komponenten anzuordnen verwendet man Layouts
- es gibt sehr viele Varianten: `http://download.oracle.com/javase/tutorial/uiswing/layout/visual.html`
- für ÜB3 habt ihr freie Wahl, daher bieten sich `FlowLayout` oder ein vertikales `BoxLayout` an
- für bessere Strukturierung: gruppiert mehrere Komponenten mit einem JPanel
- jeder Swing Container kann sein eigenes Layout haben
- default: JFrame - `BorderLayout`, JPanel - `FlowLayout`)

# Swing Tipps

## Layouts

- um Swing Komponenten anzuordnen verwendet man Layouts
- es gibt sehr viele Varianten: `http://download.oracle.com/javase/tutorial/uiswing/layout/visual.html`
- für ÜB3 habt ihr freie Wahl, daher bieten sich `FlowLayout` oder ein vertikales `BoxLayout` an
- für bessere Strukturierung: gruppiert mehrere Komponenten mit einem JPanel
- jeder Swing Container kann sein eigenes Layout haben
- default: JFrame - `BorderLayout`, JPanel - `FlowLayout`)

# Swing Tipps

## Layouts

- um Swing Komponenten anzuordnen verwendet man Layouts
- es gibt sehr viele Varianten: `http://download.oracle.com/javase/tutorial/uiswing/layout/visual.html`
- für ÜB3 habt ihr freie Wahl, daher bieten sich `FlowLayout` oder ein vertikales `BoxLayout` an
- für bessere Strukturierung: gruppiert mehrere Komponenten mit einem JPanel
- jeder Swing Container kann sein eigenes Layout haben
- default: JFrame - `BorderLayout`, JPanel - `FlowLayout`)

# Swing Tipps

## Layouts

- um Swing Komponenten anzuordnen verwendet man Layouts
- es gibt sehr viele Varianten: `http://download.oracle.com/javase/tutorial/uiswing/layout/visual.html`
- für ÜB3 habt ihr freie Wahl, daher bieten sich `FlowLayout` oder ein vertikales `BoxLayout` an
- für bessere Strukturierung: gruppiert mehrere Komponenten mit einem JPanel
- jeder Swing Container kann sein eigenes Layout haben
- default: JFrame - `BorderLayout`, JPanel - `FlowLayout`)

# Swing Tipps

## Bilder anzeigen

- grundsätzlich kann man auf jede Swing Komponente malen
- am einfachsten: `Icon` setzen
  `http://download.oracle.com/javase/tutorial/uiswing/components/icon.html`

## Beispiel

```
JFrame f = new JFrame();
f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
f.add(new JLabel(new ImageIcon(image)));
f.setSize(200,200);
f.setVisible(true);
```

# Swing Tipps

## Bilder anzeigen

- grundsätzlich kann man auf jede Swing Komponente malen
- am einfachsten: `Icon` setzen
  `http://download.oracle.com/javase/tutorial/uiswing/components/icon.html`

## Beispiel

```
JFrame f = new JFrame();
f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
f.add(new JLabel(new ImageIcon(image)));
f.setSize(200,200);
f.setVisible(true);
```

# Swing Tipps

## Bilder anzeigen

- grundsätzlich kann man auf jede Swing Komponente malen
- am einfachsten: `Icon` setzen
  `http://download.oracle.com/javase/tutorial/uiswing/components/icon.html`

## Beispiel

```
JFrame f = new JFrame();
f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
f.add(new JLabel(new ImageIcon(image)));
f.setSize(200,200);
f.setVisible(true);
```

# Swing Tipps

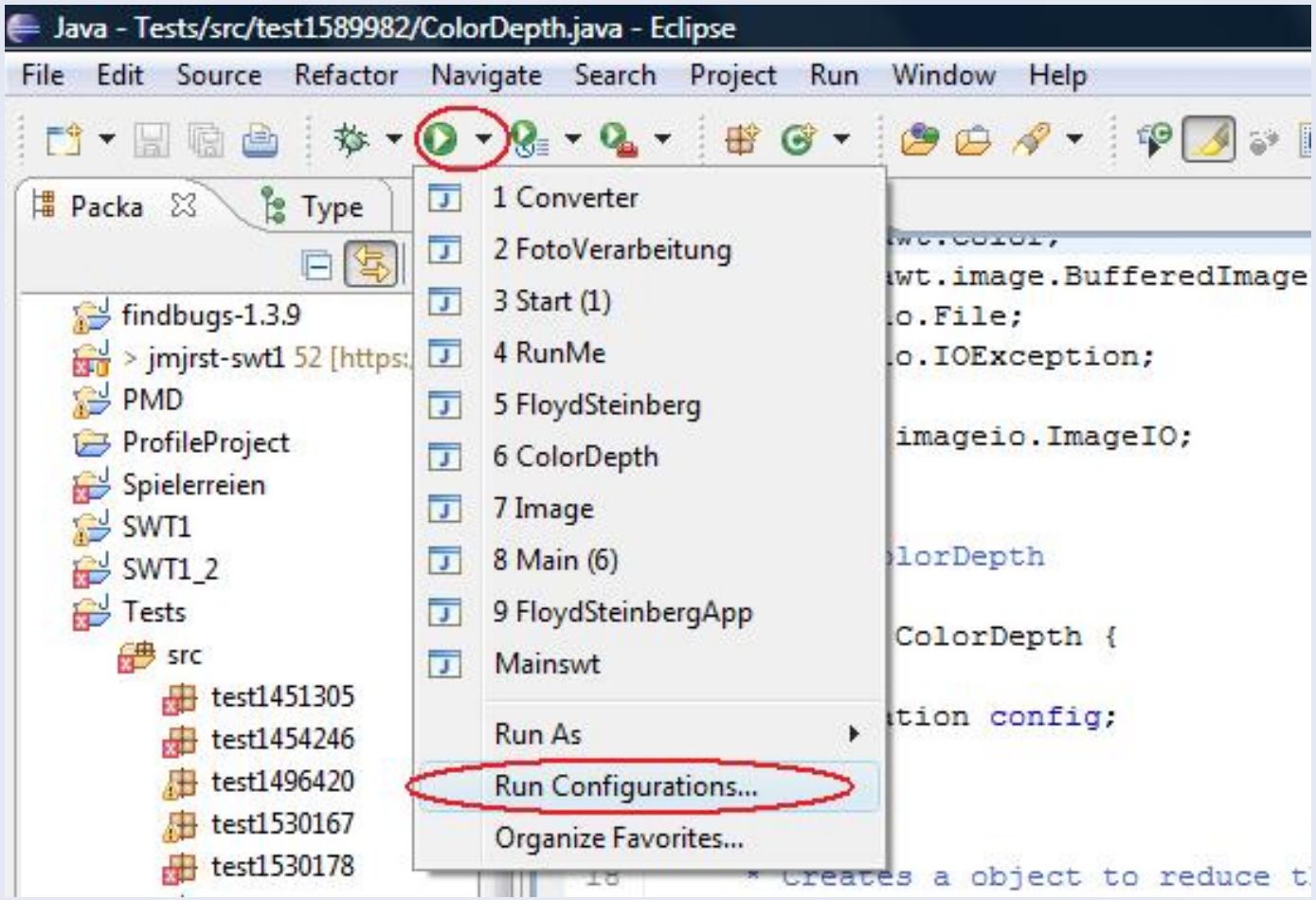
## Beispiel

```
JFrame f = new JFrame();
f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
f.setLayout(new FlowLayout());
f.setSize(220,325);

for (int i=0; i<20; i++) {
    f.add(new JLabel("Text"));
    f.add(new JButton("Button"));
}
f.setVisible(true);
```

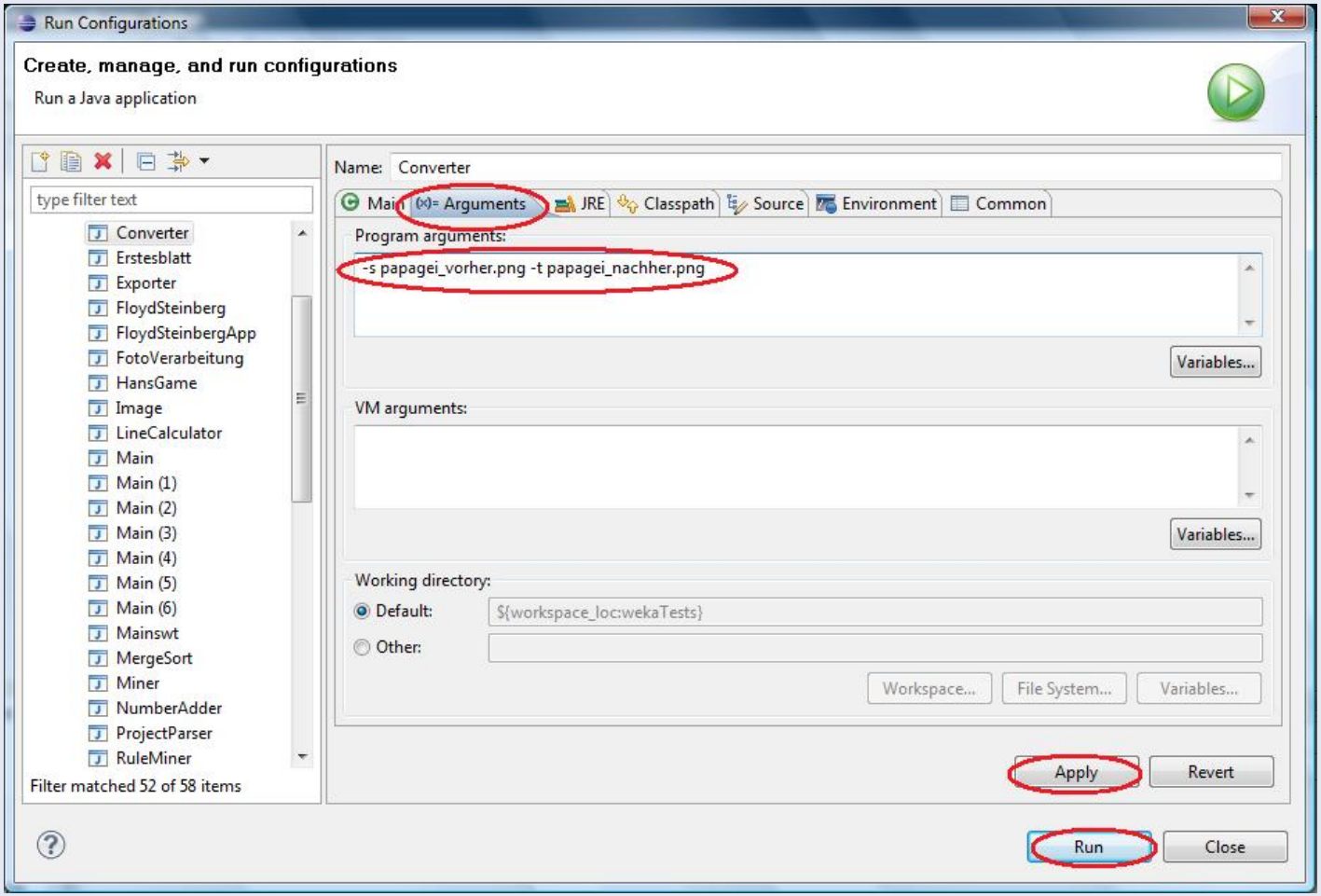
# Parameter in Eclipse übergeben

## Parameter übergeben I



# Parameter in Eclipse übergeben

## Parameter übergeben II



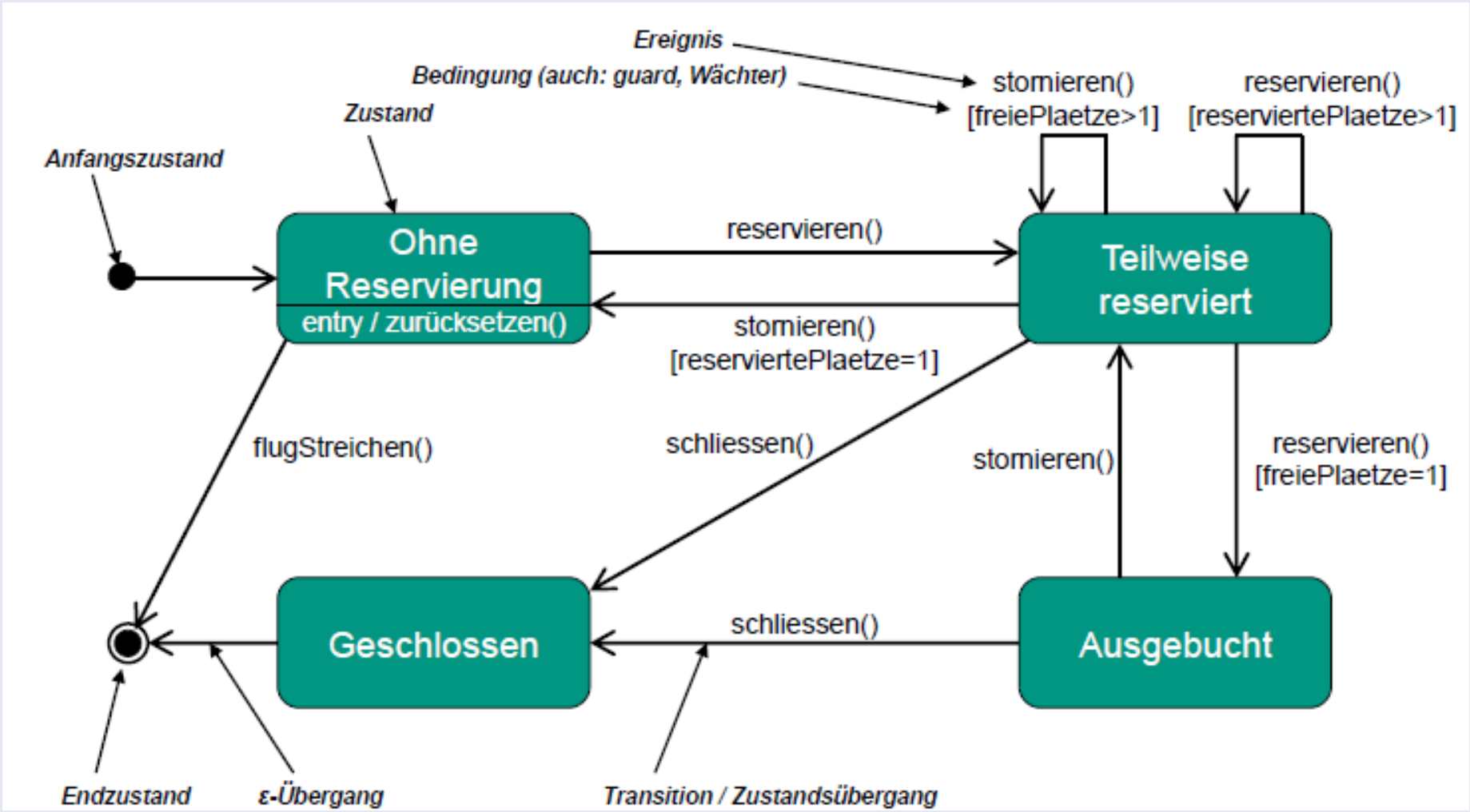
# Zustandsdiagramm

- Beschreibt mögliche Zustände eines Objekts sowie mögliche Zustandsübergänge
- Zustandsübergang (Transition) wird durch ein Ereignis ausgelöst

```
ereignis(argumente)
[bedingung]
/operation(argumente)
```

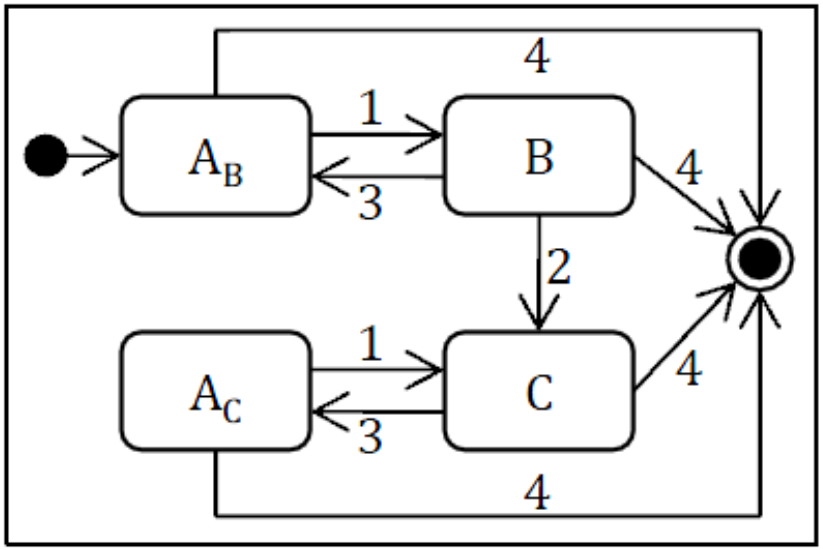
- Zustandsübergang findet nur statt, wenn zu diesem Zeitpunkt die Bedingung erfüllt ist
- $\epsilon$ -Übergang erlaubt
- Aktionen

# Beispiel Zustandsdiagramm

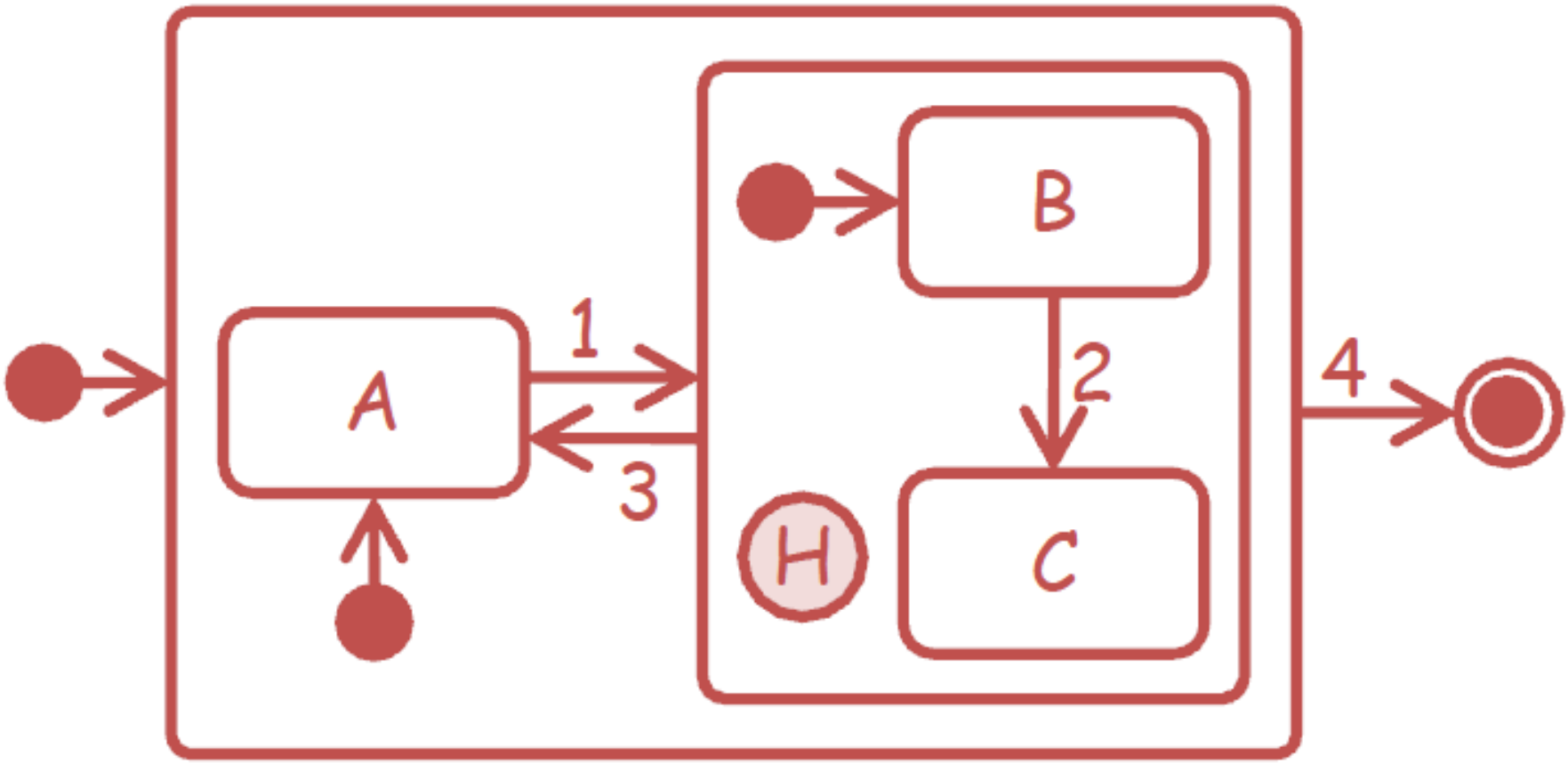


# Zustandsdiagramm Aufgabe Klausur 2010 (3P)

Wandeln Sie den rechts abgebildeten UML-Zustandsautomaten durch Zusammenlegen der Zustände $A_B$ und $A_C$ zu einem neuen Zustand A in einen äquivalenten hierarchischen Zustandsautomaten um. (3 P)
Hinweis: Beachten Sie, dass die gleiche Eingabefolge in Ihrem transformierten Automaten zu einem äquivalenten Zustand führt, wie im Originalautomaten (z. B.: Start $\rightarrow$ 1, 2, 3, 1 $\rightarrow$ C).



# Musterlösung

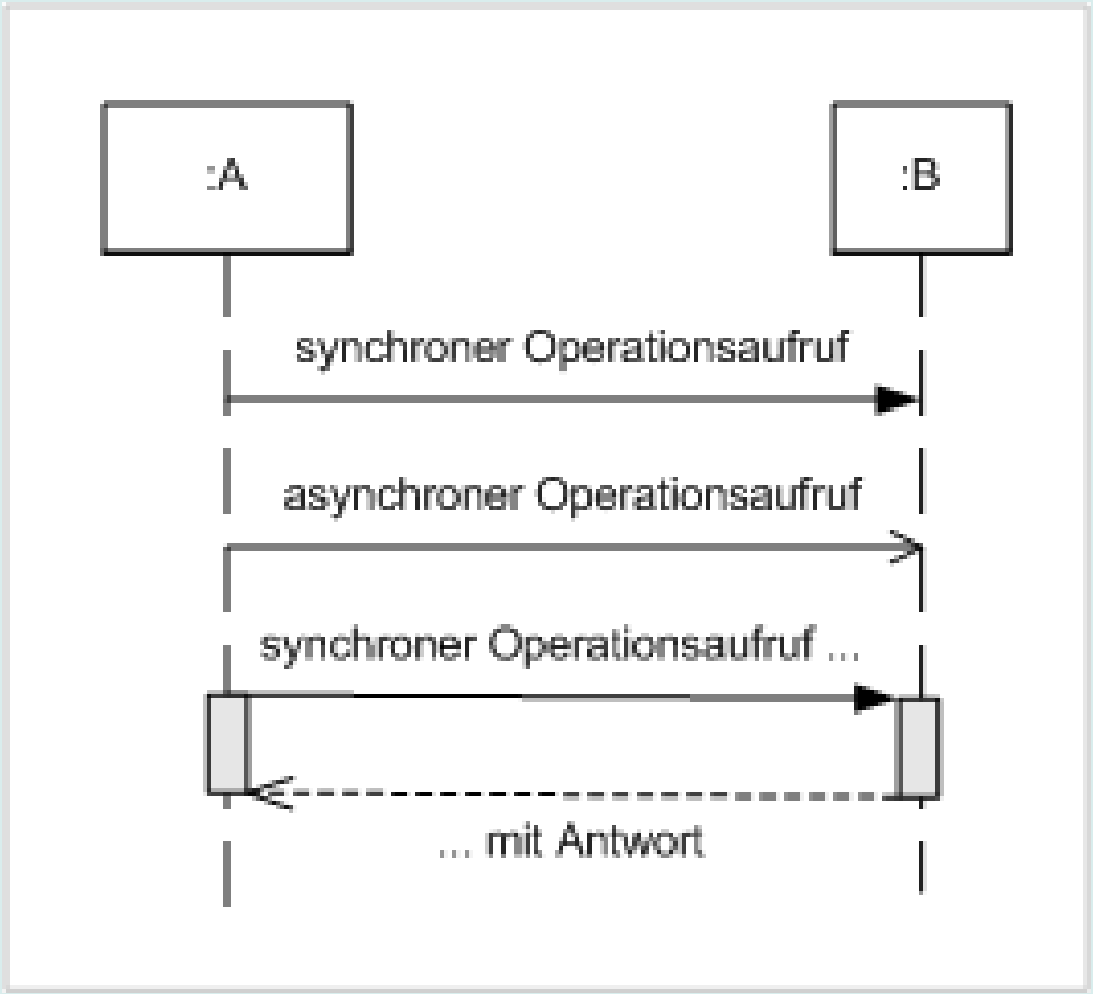


# Sequenzdiagramm Allgemein

- Darstellung eines möglichen Ablauf eines Anwendungsfalls
- Konzentration auf zeitlichen Verlauf von Nachrichten
- Zeit verläuft von unten nach oben
- Lebenslinien durch gestrichelte Linien
- Nachrichten durch waagerechte Pfeile
  $\Rightarrow$ Unterscheide synchrone und asynchrone Nachrichten
- Antworten (optional) durch gestrichelte Pfeile

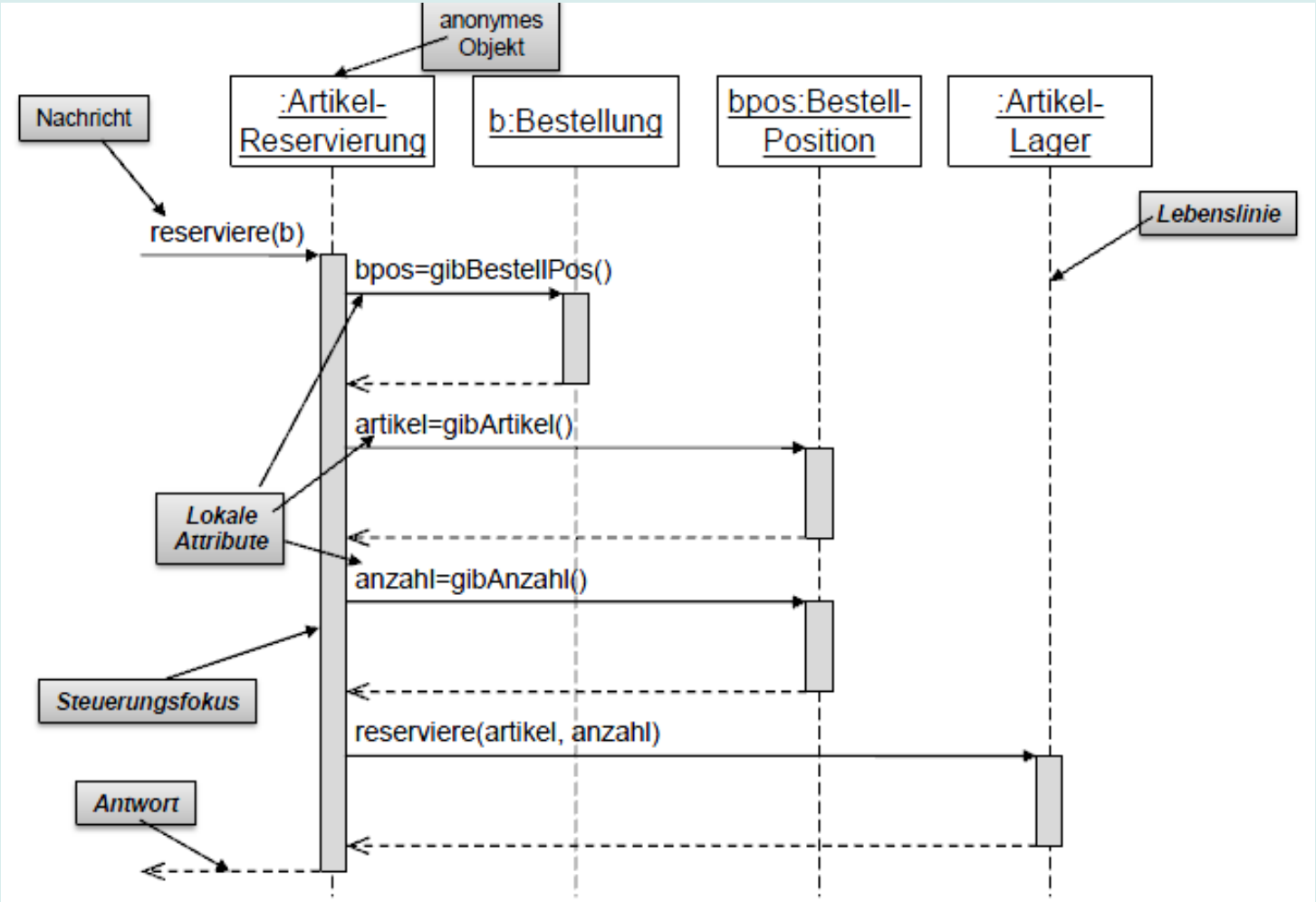
# Sequenzdiagramm

## Sequenzdiagramm - Wie verschicke ich Nachrichten?



# Sequenzdiagramm

## Sequenzdiagramm - Weiteres Beispiel

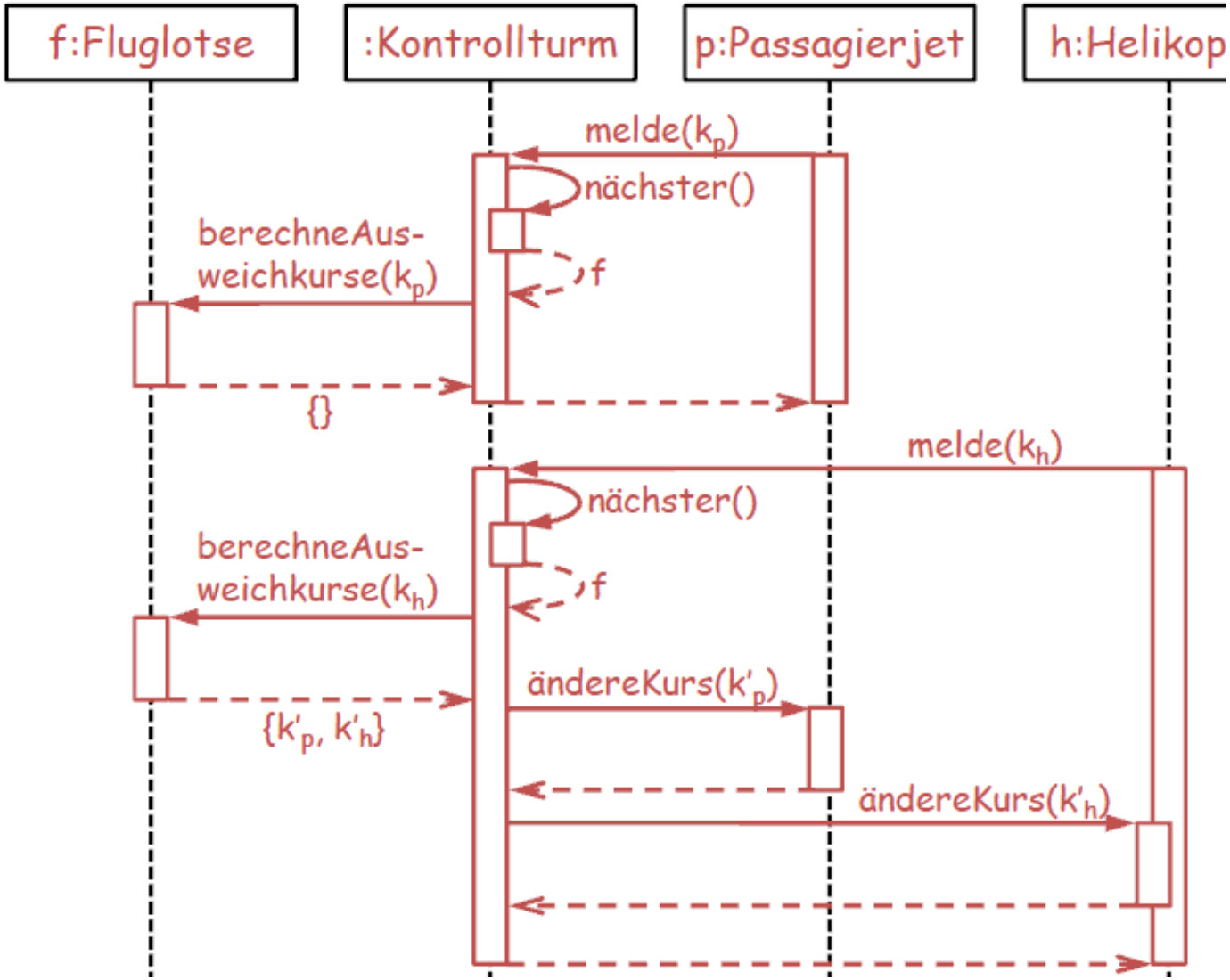


# Aufgabe Sequenzdiagramm 2010 (5P)

Ein Passagierjet p meldet seinen momentanen Kurs $k_p$ an den Kontrollturm. Der Kontrollturm beauftragt den nächsten freien Fluglotsen f zu berechnen, ob irgendwelche Flugobjekte ausweichen müssen. Der Fluglotse kommt zu dem Ergebnis, dass keine Kollisionsgefahr besteht und alle Flugobjekte ihren Kurs beibehalten können. Folglich antwortet er mit einem leeren Feld von Kursen. Anschließend meldet ein Helikopter h seinen Kurs $k_h$ an den Kontrollturm. Wieder ist der nächste verfügbare Fluglotse der Flug-lotse f. Dieser berechnet unter Berücksichtigung von $k_h$, ob nun Flugobjekte ausweichen müssen. Diesmal besteht eine Kollisionsgefahr zwischen dem Passagierjet und dem Helikopter. Deswegen antwortet der Fluglotse mit zwei Ausweichkursen $k'_p$ und $k'_h$ für den Passagierjet bzw. den Helikopter. Der Kontrollturm weist daraufhin den Passagierjet und den Helikopter an ihre Kurse auf den jeweiligen Ausweichkurs zu ändern.

Hinweis: Modellieren Sie die Kurse nicht als Objekte mit Lebenslinie. Wenn Sie Kurse als Parameter/Rückgabewert benötigen, verwenden Sie die vorgegebenen Bezeichner aus dem Szenario. Zeichnen Sie zu jedem Methodenaufruf auch die Rückantwort ein

# Musterlösung



# Tipps zum nächsten Übungsblatt

## Aufgabe 1 - Sequenzdiagramm

- denkt an Lebenslinien, synchrone vs. asynchrone Nachrichten, Steuerungsfokus
- laut Folien sind Fokus und Antworten egal, können aber in Klausur gefordert sein
- zeichnet sie also am Besten immer ein

## Aufgabe 2 - Zustandsdiagramm

- gebt stets das “Gedächtnis”” des Diagramms mit an

# Tipps zum nächsten Übungsblatt

## Aufgabe 1 - Sequenzdiagramm

- denkt an Lebenslinien, synchrone vs. asynchrone Nachrichten, Steuerungsfokus
- laut Folien sind Fokus und Antworten egal, können aber in Klausur gefordert sein
- zeichnet sie also am Besten immer ein

## Aufgabe 2 - Zustandsdiagramm

- gebt stets das “Gedächtnis”” des Diagramms mit an

# Tipps zum nächsten Übungsblatt

## Aufgabe 1 - Sequenzdiagramm

- denkt an Lebenslinien, synchrone vs. asynchrone Nachrichten, Steuerungsfokus
- laut Folien sind Fokus und Antworten egal, können aber in Klausur gefordert sein
- zeichnet sie also am Besten immer ein

## Aufgabe 2 - Zustandsdiagramm

- gebt stets das “Gedächtnis”” des Diagramms mit an

# Tipps zum nächsten Übungsblatt

## Aufgabe 1 - Sequenzdiagramm

- denkt an Lebenslinien, synchrone vs. asynchrone Nachrichten, Steuerungsfokus
- laut Folien sind Fokus und Antworten egal, können aber in Klausur gefordert sein
- zeichnet sie also am Besten immer ein

## Aufgabe 2 - Zustandsdiagramm

- gebt stets das “Gedächtnis”” des Diagramms mit an

# Tipps zum nächsten Übungsblatt

## Aufgabe 3 - Programmieren

- besteht aus den Teilaufgaben “GUI” und Floyd-Steinberg für 3,6,..,24-Bit RGB
- denkt an `int p = (o + 128 ) / 256 * 255`
  hier müsst ihr ansetzen um von 256 möglichen Ergebnissen auf $2^i$ mögliche Ergebnisse zu reduzieren
- für das GUI gibt es unzählige Tutorials u.a. bei Oracle:
  `http://download.oracle.com/javase/tutorial/uiswing/components/index.html`
  `http://download.oracle.com/javase/tutorial/uiswing/components/componentlist.html`
  `http://download.oracle.com/javase/tutorial/uiswing/layout/visual.html`

# Tipps zum nächsten Übungsblatt

## Aufgabe 3 - Programmieren

- besteht aus den Teilaufgaben “GUI” und Floyd-Steinberg für 3,6,..,24-Bit RGB
- denkt an `int p = (o + 128 ) / 256 * 255`
  hier müsst ihr ansetzen um von 256 möglichen Ergebnissen auf $2^i$ mögliche Ergebnisse zu reduzieren
- für das GUI gibt es unzählige Tutorials u.a. bei Oracle:
  `http://download.oracle.com/javase/tutorial/uiswing/components/index.html`
  `http://download.oracle.com/javase/tutorial/uiswing/components/componentlist.html`
  `http://download.oracle.com/javase/tutorial/uiswing/layout/visual.html`

# Tipps zum nächsten Übungsblatt

## Aufgabe 3 - Programmieren

- besteht aus den Teilaufgaben “GUI” und Floyd-Steinberg für 3,6,..,24-Bit RGB
- denkt an `int p = (o + 128 ) / 256 * 255`
  hier müsst ihr ansetzen um von 256 möglichen Ergebnissen auf $2^i$ mögliche Ergebnisse zu reduzieren
- für das GUI gibt es unzählige Tutorials u.a. bei Oracle:
  `http://download.oracle.com/javase/tutorial/uiswing/components/index.html`
  `http://download.oracle.com/javase/tutorial/uiswing/components/componentlist.html`
  `http://download.oracle.com/javase/tutorial/uiswing/layout/visual.html`

# Tipps zum nächsten Übungsblatt

## Bonusaufgabe - Klassendiagramm

- gute Übungsmöglichkeit
- für alle deren Klassendiagramme nicht fehlerfrei waren ;)
- zeichnet die Diagramme von Hand:
  in der Klausur hilft euch auch keine Software

# Tipps zum nächsten Übungsblatt

## Bonusaufgabe - Klassendiagramm

- gute Übungsmöglichkeit
- für alle deren Klassendiagramme nicht fehlerfrei waren ;)
- zeichnet die Diagramme von Hand:
  in der Klausur hilft euch auch keine Software

# Bis zum nächsten Mal