Dipl.-Inform. Thomas Ziemer
www.tfh-berlin.de/~ziemer
Stand: 4. September 2006

4 Anhang

EINFÜHRUNG IN BORLAND TOGETHER CONTROLCENTER

Ich danke Frau Ilse Schmiedecke für ihre nette Erlaubnis, die generelle Struktur ihres Skripts *Hinweise zur Modellierung mit Together* für diese Lehrunterlage verwenden zu dürfen.

4.6 Einführung

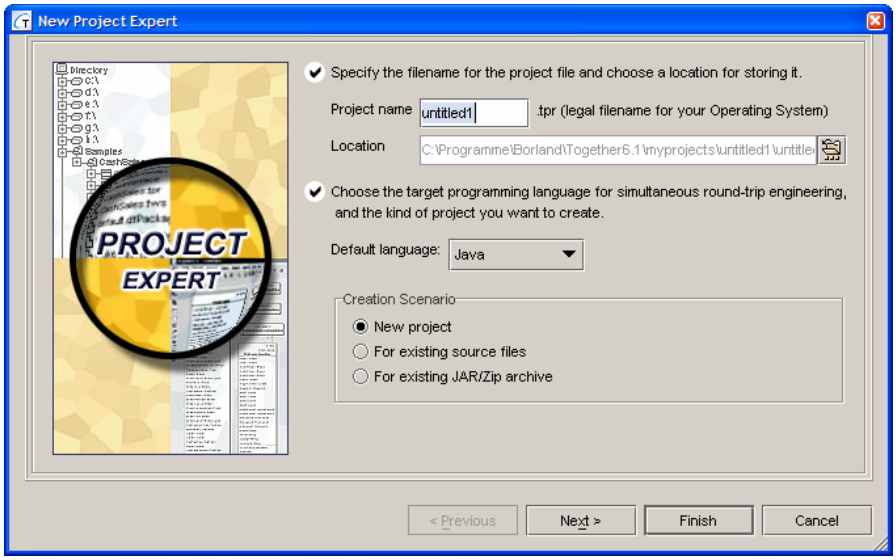
Borland Together ControlCenter ist ein CASE-Tool (*computer aided software engineering tool*), das im Rahmen dieser Lehrveranstaltung zur Modellierung eingesetzt wird. Together 6.2 unterstützt die UML 1.4 vollständig und enthält auch die meisten Neuerungen der im Jahre 2004 verabschiedeten UML Version 2.0.

Together unterscheidet sich vom Konzept einiger anderer CASE-Tools durch das Leistungsmerkmal *LiveSource*™, bei dem jedes UML-Diagramm, zu dem es eine Entsprechung im Programmquelltext geben kann, auch sofort im Quelltext der gewählten Programmiersprachedargestellt wird. Jede Änderung im Diagramm oder im Quelltext wirkt sich dabei sofort in der jeweils anderen Richtung aus (*reverse engineering*). In anderen CASE-Tools muss der Entwickler die Generierung des Programmquelltextes aus den Diagrammen (oder umgekehrt) zu geeigneten Zeitpunkten manuell veranlassen. Der Vorteil des *LiveSource*™ liegt darin, dass in Together das Modell und der Quellcode jederzeit zueinander passen.

Damit dieser Vorteil in Softwareprojekten bis zu deren Auslieferung erhalten bleibt, enthält Together komfortable Enwicklungswerkzeuge, wie einen sprach- und musterkundigen (*patterns*) Editor, einen Debugger, Testwerkzeuge, einen leistungsfähigen UI-Designer und Werkzeuge zur Entwicklung von webbasierten Anwendungen und -services. Softwareprojekte können somit ohne Medienbruch vollständig in Together realisiert werden, ohne dass weitere Entwicklungswerkzeuge benötigt werden.

4.7 Modellierung mit Together

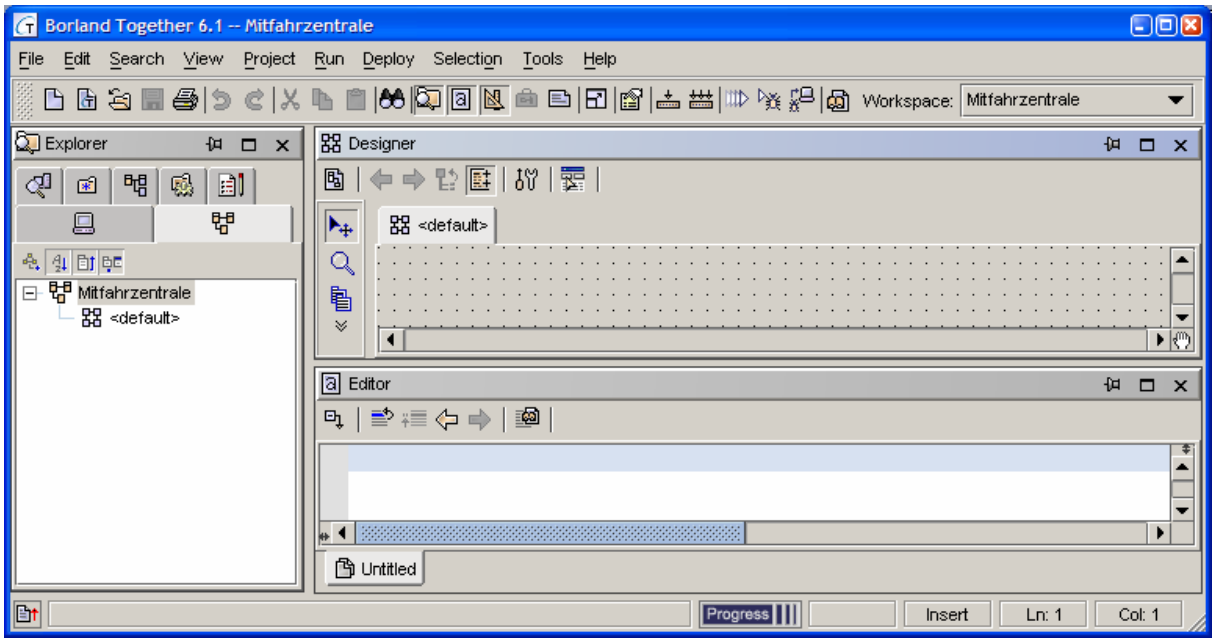
Wie beispielsweise aus Eclipse bekannt, wird in Together mit Projekten gearbeitet. Im Rahmen der Lehrveranstaltung ist unser Fallbeispiel die Mitfahrzentrale. Es wird dafür in Together genau ein Projekt erstellt, das von Woche zu Woche fortgeschrieben wird:



Ein Projekt wird erstellt mit dem New Project Expert unter *File→New Project Expert…*

Der gewünschte Projektname wird unter *Project name* eingetragen, und das zukünftige Arbeitsverzeichnis für alle zu erstellenden Modelle unter *Location*.

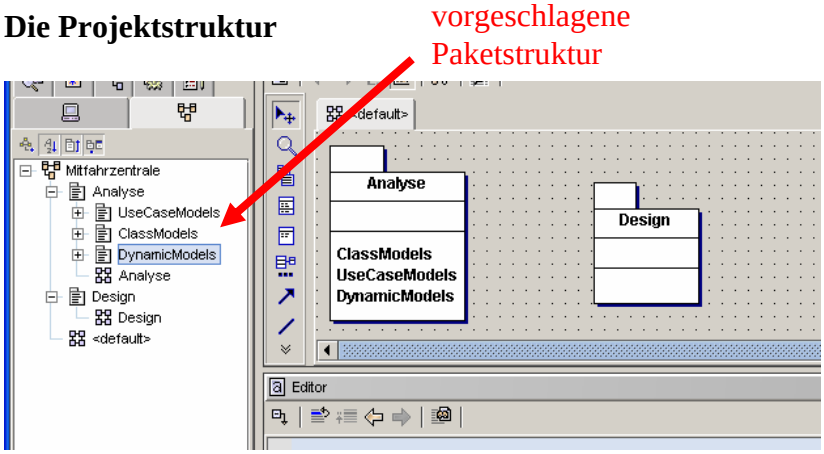
Es erscheint ein dreigeteiltes Fenster (siehe Abbildung), das auch *Workspace* genannt wird. Unter der Hauptmenüzeile und der Symbolleiste befinden sich links das Explorer-Fenster, rechts oben das Designer-Fenster und rechts unten das Editor-Fenster. Weitere Fenster können später hinzugefügt werden, beispielsweise während der Arbeit mit dem grafischen UI-Designer.



Weitere wichtige Sichten (*views*) des Explorer-Fensters, die über die einzelnen Reiter dieses Fensters ausgewählt werden können, sind die *Directory*-Sicht (das Monitorsymbol), die die Verzeichnisstruktur der Festplatte darstellt, und die *Diagrams*-Sicht (drei miteinander verbundene Klassensymbole), die einen Überblick über die verfügbaren und verwendeten UML-Diagrammtypen gibt.

Sogenannte *default*-Diagramme werden von Together standardmäßig für jedes Projekt und jedes Paket erstellt. Sie stellen gewissermaßen ein Inhaltsverzeichnis dar. Im Explorer-Fenster sind sie an dem Symbol mit den vier verbundenen Kästchen zu erkennen. Es empfiehlt sich nicht, diese Diagramme zur Modellierung zu benutzen, sondern dafür eigene Diagramme zu erzeugen!

Die Projektstruktur



Das Together-Projekt soll im Rahmen der Lehrveranstaltung alle Diagramme und den Quelltext aufnehmen. Daher ist es sinnvoll, bereits zu Beginn eine gute Struktur festzulegen, die im laufe des Projekts keine unerwarteten Probleme er-

zeugt. Es wird daher empfohlen, zunächst die oben abgebildete Paketstruktur aufzubauen.

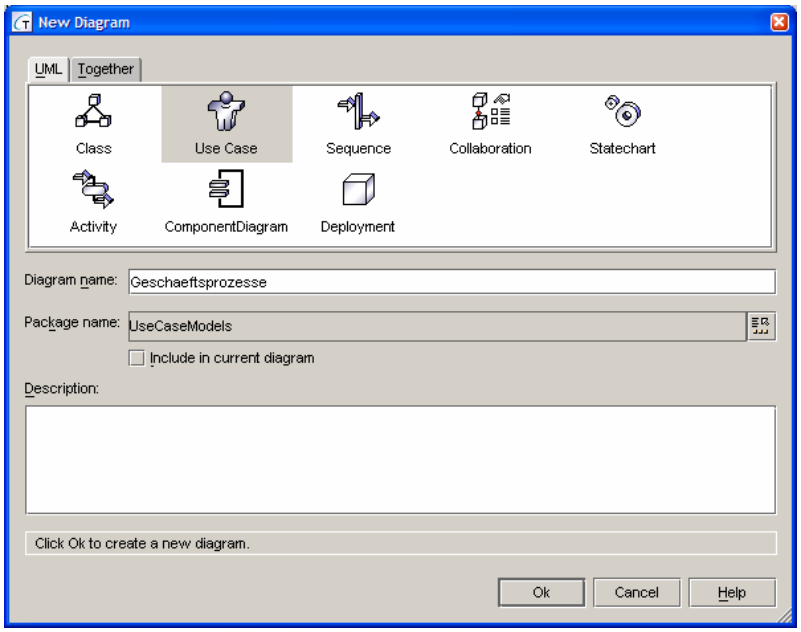
Together-Projekte sind in Pakete gegliedert, die ihrerseits Diagramme und Programmquelltexte enthalten. Pakete entsprechen Java-Paketen und sind gleichermaßen Namensräume.

Ein neues Paket wird erzeugt, indem über das Kontextmenü des *default*-Pakets im Explorer-Fenster (mit rechter Maustaste darauf klicken) *New→Package* gewählt wird, oder alternativ das Symbol *New Diagram* im Designer-Fenster geklickt und darin ein neues Paket ausgewählt wird.

Innerhalb des Projektbaumes sind nun zwei Pakete namens Analyse und Design zu erstellen. Das Analysepaket enthält drei weitere Pakete namens UseCaseModels, ClassModels und DynamicModels. Namen müssen generell java-konform sein, dürfen also keine Sonderzeichen oder Umlaute enthalten. Bezeichner wie *static* müssen unbedingt vermieden werden, da es sich bei diesen um reservierte Java-Schlüsselwörter handelt, die ansonsten dem Compiler bei der Quelltextübersetzung Probleme bereiten würden.

Ein neues Diagramm anlegen

Das erste im Rahmen der Lehrveranstaltung angelegte Diagramm ist das UseCase-Diagramm. Es soll *Geschaeftsprozesse* heißen und im Paket *Analyse.UseCaseModels* erstellt werden.

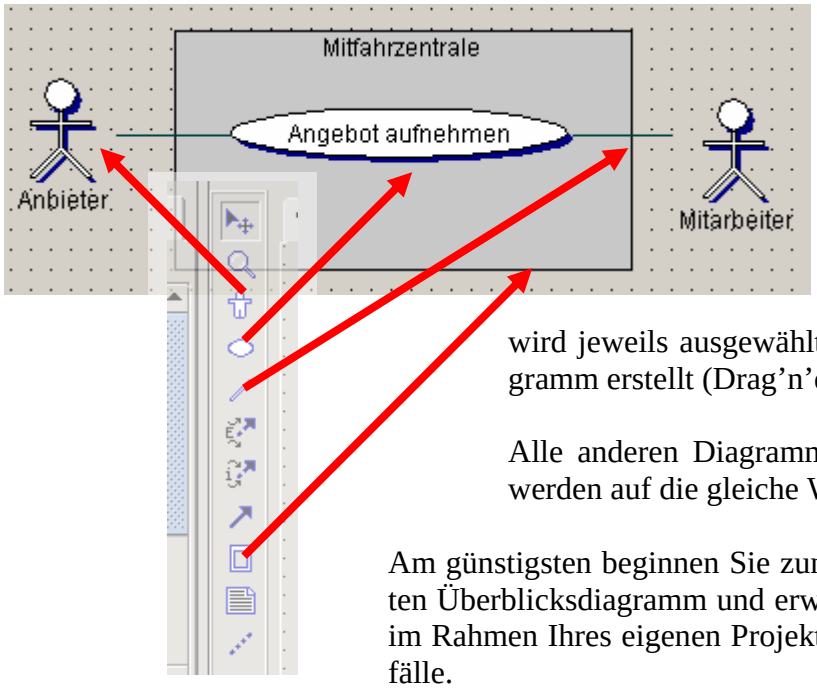


Im Explorer-Fenster wird das Paket *UseCaseModels* markiert und dann über dessen Kontextmenü *New→Diagram→Use Case* das gewünschte Diagramm erstellt. Der Name wird im Feld *Diagram name* eingetragen.

Wird das Diagramm irrtümlich im falschen Paket erstellt, so lässt es sich auch im Nachhinein noch per Drag'n'drop verschieben. Dazu muss allerdings zunächst in den Together-Einstellungen die Möglichkeit des Drag'n'drop aktiviert werden. Dazu ist über das Hauptmenü die Option *Tools→Options→Default Level* auszuwählen. Innerhalb der Sektion *General* im *Default Options*-Fenster befindet sich die Einstellung *Enable Drag-and-Drop*, welche zu aktivieren ist.

Achtung: Together „verschluckt" sich beim Verschieben recht gern. Es empfiehlt sich daher, vor größeren Verschiebeaktionen das Projekt mittels *File→Save All* abzuspeichern. Regelmäßiges Abspeichern kann übrigens auch sonst nie verkehrt sein…

Im neuen Use Case-Diagramm wird nun ein Use Case mit zwei Akteuren modelliert. Dazu muss zunächst das soeben erstellte Diagramm mittels Doppelklick im Explorer-Fenster geöffnet werden.



Das gewünschte Symbol des Akteurs (*Actor*), des Geschäftsprozesses (*Use Case*), der Beziehung (*Communicates*) und der Systemgrenze (*System Boundary*) in der Werkzeugleiste am linken Rand des Designer-Fensters wird jeweils ausgewählt, und durch Klick in das Diagramm erstellt (Drag'n'drop geht hier leider nicht…).

Alle anderen Diagrammtypen, die Together anbietet, werden auf die gleiche Weise bedient.

Am günstigsten beginnen Sie zunächst mit dem hier abgebildeten Überblicksdiagramm und erweitern dieses sukzessiv um die im Rahmen Ihres eigenen Projekts identifizierten Anwendungsfälle.

Diagramme kopieren

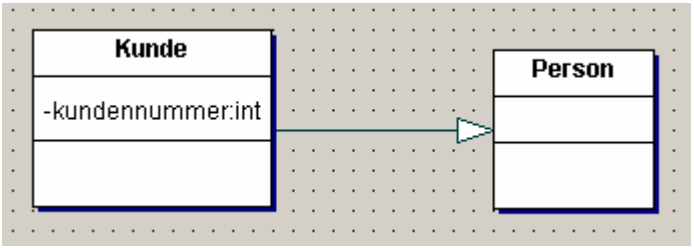
Oft besteht der Wunsch, Diagramme zu kopieren, beispielsweise um in einem zweiten Diagramm weitere Details hinzuzufügen. Dazu enthält im Explorer-Fenster das Kontextmenü jedes Diagrammtyps den Eintrag *Clone*, mit dem eine genaue Kopie des Diagramms erzeugt werden kann. Das entstehende Diagramm ist unabhängig vom Original, d.h. Modifikationen des einen bleiben ohne Auswirkung auf das andere. Achtung: dies gilt nicht für Klassendiagramme, dazu jedoch später mehr!

Diagramme drucken oder als Grafik exportieren

Über das Dateimenü *File→Print Diagram…* lassen sich Diagramme drucken. Damit hierbei nicht unnütz Papier verschwendet wird, müssen folgende Einstellungen gewählt werden: Im ersten Dialogfenster die Eigenschaft *Current*, damit nur das augenblickliche Diagramm gedruckt wird. Dann die Schaltfläche *Print Options…* klicken und im sich öffnenden Fenster *Diagram options: Print* die Einstellung *Fit to page* (sie steht in der vorletzten Zeile) auswählen. Das augenblickliche Diagramm wird dann optimal gedruckt.

Diagramme können auch als Grafikdatei exportiert werden, um sie beispielsweise in Worddokumente einbinden zu können. Dies geschieht über das Menü *File→Save Image →<Format>*. Die entstehende Grafik wird standardmäßig im selben Verzeichnis gespeichert wie das Diagramm selbst, im vorliegenden Fall also unter *<Arbeitsverzeichnis>\Analyse \UseCase*. Aber: alle Together-Diagramme liegen ohnehin bereits im WMF-Format vor. Dieses kann auch ohne die beschriebene Konvertierung direkt in Word eingebunden werden.

4.8 Klassendiagramme und LiveSource™



Ein neues Klassendiagramm wird natürlich in unserem Paket *ClassModels* erstellt. Danach wird beispielsweise eine Klasse (*Class*) *Kunde* und eine Klasse *Person* eingezeichnet, und die beiden Klassen werden über eine Vererbung (*Generalization/Implementation*) miteinander verbunden. Über das Kontextmenü *New→Attribute* der Kunden-Klasse wird noch ein Attribut *kundennummer* definiert. Markieren Sie nun die Klasse *Kunde*. Im Editor-Fenster wird sofort der entsprechende Programmquelltext dieser Klasse angezeigt. Wie man sieht, erweitert sie die Klasse *Person* („*public class kunde extends Person*“) und enthält ein privates, typnumerisches Attribut *kundennummer*. Wird die Klasse *Person* markiert, wird stattdessen ihr Programmquelltext im Editor angezeigt.

Wird der Typ des Attributs im Editor-Fenster beispielsweise von *int* in *string* geändert, wird sofort die entsprechende Klasse im Klassendiagramm aktualisiert. Die sichtbare Übernahme geschieht allerdings erst nach einen Klick in das Diagramm oder durch Druck auf die Taste *F5*, (*Update* im Kontextmenü) während der Fokus sich bereits im Diagrammfenster befindet.

Da Klassendiagramme programmquelltextbasierte Diagramme sind, erscheinen die Modellelemente der Klassen *Person* und *Kunde* auch im Explorer-Fenster.

Bitte beachten: wenn von einen Klassendiagramm eine Kopie mittels *Clone* erstellt wird, dann wirkt sich jede Veränderung des einen Diagramms auch auf das andere Diagramm aus, da sich beide Diagramme auf denselben Programmquelltext beziehen!

Elemente ausblenden

Sollen in einem geklonten Diagramm einzelne Elemente ausgeblendet werden, ohne dass sie tatsächlich aus dem Modell entfernt werden sollen, ist *Delete from View* im Kontextmenü des Elements zu verwenden. Das Element (evtl. mit all seinen Beziehungen etc.) kann jederzeit wieder mittels Drag’n’drop aus dem Explorer-Fenster in das Diagramm dargestellt werden.

Namensräume

Wird in einem nächsten Klassendiagramm versucht, eine weitere Klasse *Kunde* anzulegen, kommt es zu einer Fehlermeldung, da diese Klasse bereits existiert. Zur Lösung gibt es zwei Möglichkeiten: Entweder möchte man die Klasse erneut in ein Klassendiagramm integrieren, und wählt dazu im Kontextmenü der Klasse innerhalb des Explorer-Fensters *Add as Shortcut*. Man erhält eine Verknüpfung mit der schon vorhandenen Klasse. Oder man benötigt einen neuen Namensraum (*name space*), in dem der Name *Kunde* noch nicht bekannt ist, d.h. man legt das neue Diagramm in einem anderen Paket an.

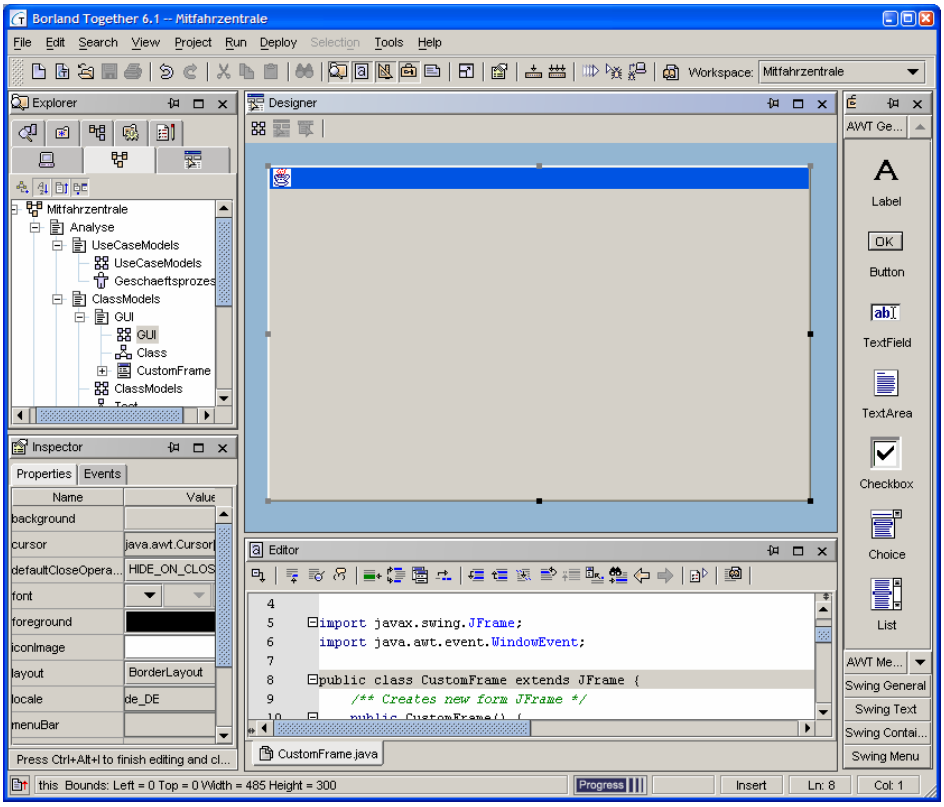
4.9 Oberflächenprototypen mit dem UI-Designer

Together verfügt über einen leistungsfähigen UI-Designer (*user interface designer*), mit dem Swing- und AWT-Oberflächen für Java grafisch erzeugt werden können. Dieser ist sogar so weit konfigurierbar, dass mit ihm Oberflächen weiter bearbeitet werden können, die beispielsweise mit *Borland JBuilder* oder *NetBeans* erzeugt wurden.

Wir werden den UI-Designer jedoch nur als reines Zeichenwerkzeug einsetzen, um einen explorativen Prototypen der Benutzungsschnittstelle zu erzeugen.

Um eine neue grafische Oberfläche zu generieren, benötigt man zunächst eine Klasse *MeinFenster*, die *JFrame* erweitert. Dazu wird in einem neu zu erstellenden Paket *GUI* eine Klasse erzeugt, jedoch nicht wie bisher über *New→Class*, sondern über *New→Class by Pattern…* In dem sich öffnenden Fenster *Choose Pattern* ist innerhalb der *UI Components* die Klasse *JFrame* auszuwählen. Alternativ kann auch aus dem Dateimenü heraus *File→New→User Interface→Swing Containers→JFrame* aufgerufen werden.

Für den Fall, dass sich der UI-Designer nun von allein öffnet, kann er über das entsprechende Symbol *UI Designer* in der Symbolleiste des Designer-Fensters manuell aktiviert werden, oder über das Hauptmenü *View→Designer View→UI Designer*.



Im Designer-Fenster ist nun wie abgebildet zunächst eine graue, leere Java-Oberfläche zu sehen. Rechts befindet sich eine große Auswahl an Swing- und AWT-Komponenten, sowie -Containern, die per Mausklick ausgewählt und mit einem weiteren Mausklick beliebig auf der Oberfläche platziert und in der Größe verändert werden können.

Sollte das Symbol des UI Designers überhaupt nicht vorhanden sein, so kann es über *Tools→Activate/Deactivate Features…→Together Features→UI Builder* aktiviert werden.

Der Inspektor

Um vollständige Kontrolle über das Layout der Oberfläche zu erhalten, muss das Layout der JFrame-Klasse noch von *BorderLayout* auf *NullLayout* umgestellt werden. Dazu ist das JFrame-Objekt zu markieren (durch einen Klick irgendwo in die noch leere Java-Oberfläche), und im Reiter *Properties* des unterhalb des Explorer-Fensters liegenden Inspektors (*inspector*) die Eigenschaft *layout* in *NullLayout* zu ändern.

Sollte der Inspektor überhaupt nicht sichtbar sein, so kann er über das Hauptmenü *View→Main Panes→Properties…* aktiviert werden.

Das Properties-Fenster des Inspektors ist der Schlüssel zur Gestaltung der Oberfläche. Dort können für jede Komponente und jeden Container die Eigenschaften wie beispielsweise Farbe und Beschriftung festgelegt werden.

Die Verwendung erfordert natürlich Kenntnisse über die Eigenschaften und das Verhalten der einzelnen Java-Komponenten und -Container (wie beispielsweise *JTabbedPane*), die im Rahmen dieser Lehrveranstaltung jedoch nicht thematisiert werden.

Ausführen der Oberfläche

Um die Oberfläche als unseren explorativen Prototypen der Benutzungsoberfläche ausführen zu können, muss zunächst noch eine Main-Methode generiert werden. Dazu ist erst die dafür gewünschte Klasse zu markieren, und in ihrem Kontextmenü *New→Member by Pattern…* auszuwählen. Im sich öffnenden Fenster *Choose Pattern - Main method* ist dann der Eintrag *Main method* zu wählen. Die Main-Methode wird nun automatisch generiert.

Im Rumpf der Main-Methode muss als allererstes ein Objekt der Klasse *MeinFenster* instanziiert und im Anschluss sichtbar gemacht werden, da sonst nichts zu sehen sein wird:

```
new MeinFenster().setVisible(true);
```

Jetzt kann die Anwendung mit Hilfe der Schaltfläche *Run* auf der Symbolleiste oder durch Druck auf die Taste *F9* ausgeführt werden.

Es geht nicht…

Sollte die Darstellung des Fensters sich nicht verhalten wie erwartet, so wird dies höchstwahrscheinlich zum einen daran liegen, dass die Operation *pack()* im Konstruktor der Fensterklasse aufgerufen wird. Sie sollte auskommentiert werden.

Zum anderen bewirkt eine beliebige Größenänderung des Fensters im UI-Designer, dass es im Programmquelltext mit einer expliziten Größenangabe versehen und auch so angezeigt wird, und nicht mit der minimalen Größe. Dies kann auch manuell durch das Einfügen der Zeile

```
setBounds(new java.awt.Rectangle(left, top, right, bottom));
```

in der Operation *initGUI()* des Quelltextes geschehen, wobei *left*, *top*, *right* und *bottom* für die gewünschten Ausmaße stehen.

Eine zusätzliche Zeile

```
setLocationRelativeTo(null);
```

in *initGUI()* bewirkt ferner eine bildschirmzentrierte Ausgabe des Fensters.

Andererseits kann das unerwartete Verhalten einer Anwendung freilich auch unendlich viele andere Gründe haben…

Ereignisbehandlung hinzufügen

Unsere Anwendung ist leider noch vollständig reaktionslos (das Fenster lässt sich zumindest schließen, was ja auch schon mal eine Reaktion ist).

Nach der Erstellung und Markierung einer Komponente, beispielsweise eines *Buttons*, kann im Reiter *Events* des Inspektors eine Routine zur Ereignisbehandlung definiert werden, beispielweise *actionPerformed*. Mit einem Klick auf diesen Typ der Ereignisbehandlungsroutine erscheint ein generischer Name der Routine als Vorschlag im rechts davon liegenden Textfeld, im Beispiel *button1ActionPerformed*. Nach erneuten Klick auf den Typ *actionPerformed* wird diese Routine im Programmquelltext erzeugt und registriert. Ihr Rumpf ist leer. Die Zeile

```
button1.setBackground(new jawa.awt.Color(255,255,255));
```

führt beim Klick auf die Schaltfläche zu ihrer eigenen Farbänderung. Wie einfallsreich…

4.10 Weitere Together-Einstellungen

In Klassendiagrammen der UML ist die Darstellungsreihenfolge Attribute, dann Operationen, in Together jedoch *Operations, then Attributes*. In den Einstellungen *Tools→Options→Default Level→Source Code→Order* kann die Eigenschaft *Position members* auf *Attributes, then Operations* geändert werden.

In Klassendiagrammen der UML zeigt die Pfeilrichtung einer Assoziation die Richtung der Assoziation an. In Together wird jedoch keine Pfeilspitze angezeigt. In den Einstellungen *Tools→Options→Default Level→Diagram→Associations* kann die Eigenschaft *Draw directed* auf *All* geändert werden.

Für die Weiterverarbeitung der in Together erstellten Diagramme ist es oft nützlich, ihre 3D-Darstellung auszuschalten. Dies geschieht über *Tools→Options→Default Level→Diagram*. Die Eigenschaft *3D look* kann darin deaktiviert werden.