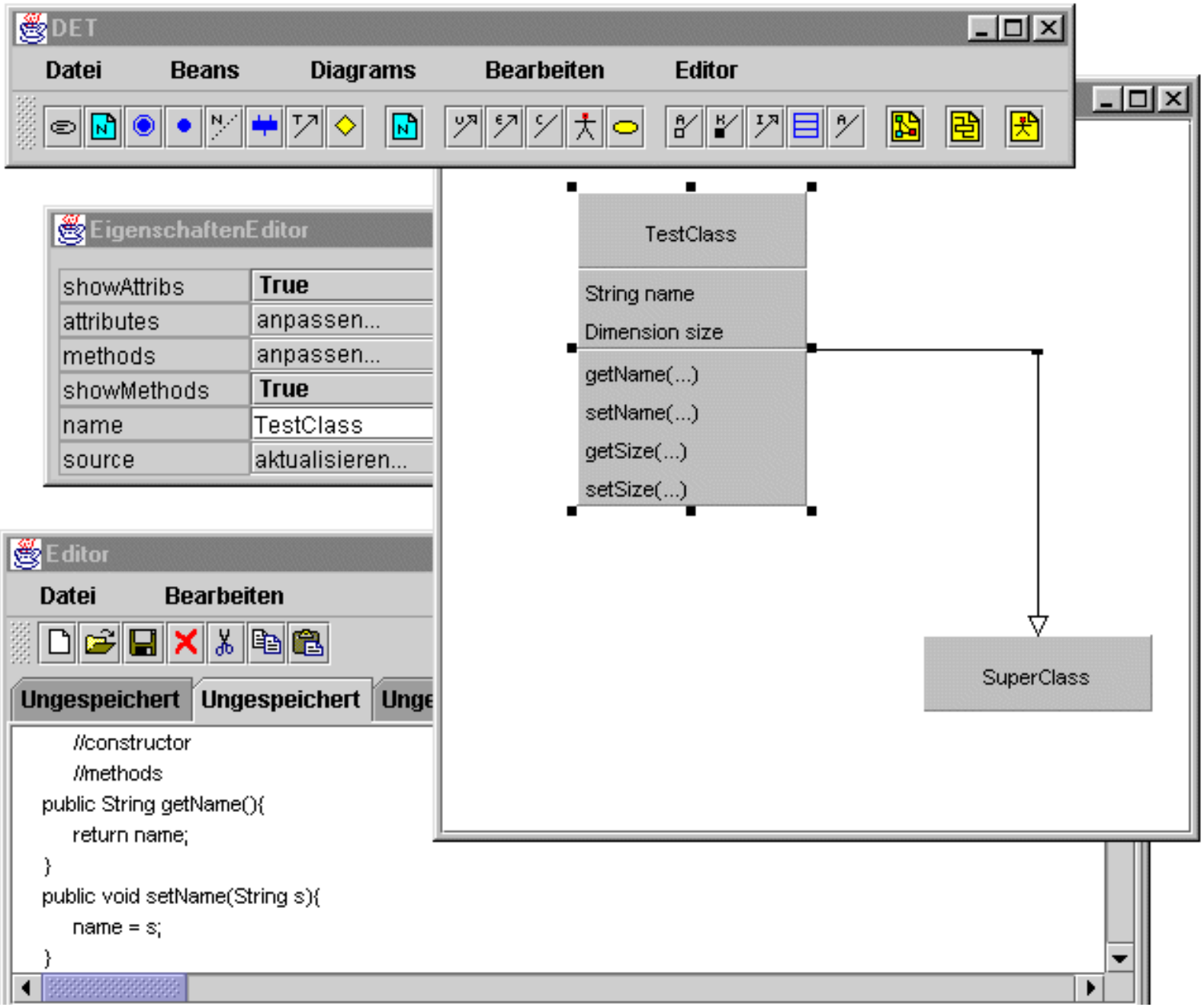
DET - Ein erweiterbarer Diagramm-Editor

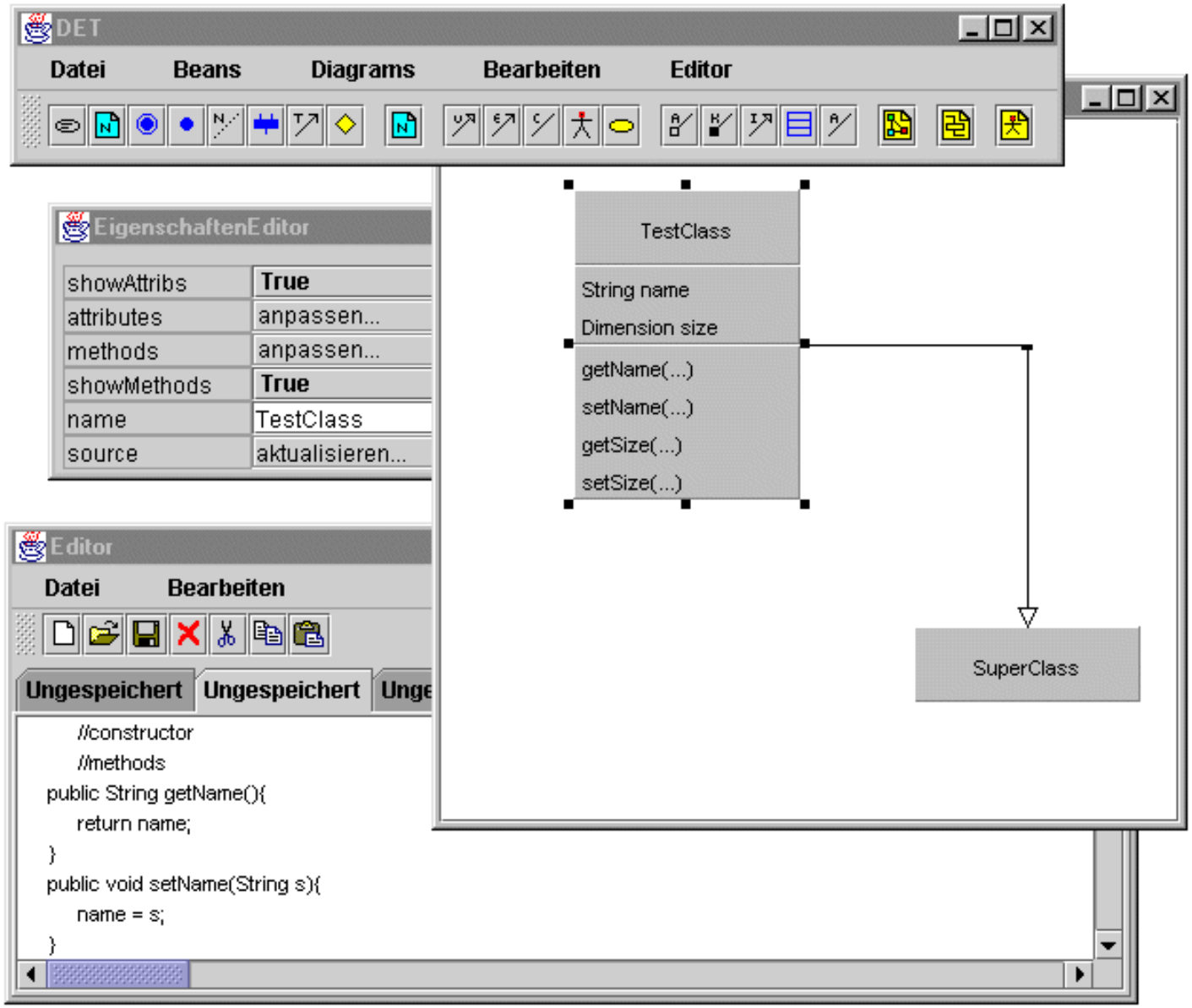


1. Allgemeines

1.1. Ziele und Anwendungszweck

Der Diagramm Editor DET erlaubt die Darstellung und Manipulation von Diagrammen, die auf Elementen und Verknüpfungen zwischen diesen Elementen basieren. Die darstellbaren Elemente und Verknüpfungen sind abgeschlossene Softwarebausteine und können Funktionen enthalten, die über die reine Darstellung hinausgehen. So können Elemente und Verknüpfungen innerhalb der Darstellung in definierter Weise miteinander interagieren.

DET soll die Ausbildung von Programmier-anfängern unterstützen. Eine kommerzielle integrierte Entwicklungsumgebung beinhaltet zumeist unzählige Funktionen, die den Programmieranfänger eher verwirren als unterstützen. Deshalb wurde DET so entworfen, daß sich ihm sowohl neue Diagrammarten als auch weitere Funktionen (z.B. Kompilieren, Parsen)



beliebig hinzufügen lassen. So verfügt DET zu Beginn nur über die Funktionen und Diagrammarten, die der Lernende benötigt, um die Beispiele und Aufgaben des Kurses nachzuvollziehen bzw. zu bearbeiten. Im Laufe der Ausbildung kann DET dann, auf den Kurs und die zu vermittelnden Inhalte zugeschnitten, Schritt für Schritt ausgebaut werden.

1.2. Entwurfsziele

Unterstützung beliebiger Darstellungen	Aussehen und Verhalten der darzustellen Elemente und Verknüpfungen sollen beliebig definierbar sein. Mit DET soll sich sowohl ein UML-Editor als auch z.B. ein Dialogfeld-Editor realisieren lassen.
Mehrere Ebenen der Darstellung	Es soll möglich sein, eine Menge von Elementen und Verknüpfungen auf einer höheren Ebene als Einheit betrachten zu können und zwischen diesen Einheiten (die nun als Elemente aufgefaßt werden können) wiederum Verknüpfungen herzustellen.
Dynamische Erweiterbarkeit des Funktionsumfangs	Der Funktionsumfang von DET soll nach Belieben ergänzt werden können. Neue Funktionen sollen als Module zur Laufzeit in die Anwendung eingebunden werden können. Da die verfügbaren Funktionen im Voraus nicht bekannt sind, soll die Kommunikation zwischen den Funktionsmodulen in einer definierten Weise erfolgen, ohne daß die beteiligten Module voneinander Kenntnis haben müssen.
Dynamische Erweiterbarkeit der darzustellenden Diagrammarten	Die Darstellungsmöglichkeiten des Editors sollen wie die Funktionen dynamisch erweiterbar sein. Die für eine spezifische Darstellung benötigten Darstellungsarten sollen sich ebenfalls zur Laufzeit in die Anwendung einbinden lassen.
Laden der Module über ein Netzwerk	Funktions- und Diagrammodule sollen über ein Netzwerk geladen und verteilt werden können. Dadurch läßt sich z.B. eine automatische Aktualisierung der Anwendung von zentraler Stelle realisieren.
Interaktion zwischen Bestandteilen der Darstellung	Elemente und Verknüpfungen sollen innerhalb einer Darstellung miteinander interagieren können. Die Interaktion zwischen Elementen soll sowohl von den Verknüpfungen realisiert werden als auch für den Benutzer anpaßbar bzw. frei definierbar sein.

1.3. Entwurfsentscheidungen

Verwendung von Java als Implementierungssprache	Zur Implementierung wird Java gewählt, weil Java plattformunabhängig, objektorientiert und frei verfügbar ist. Gegen Java spricht nur die mangelhafte Geschwindigkeit der virtuellen Maschine.
Verwendung von JavaBeans	JavaBeans ist eine Komponententechnologie und eignet sich deshalb für die Implementierung der Diagrammelemente und -verknüpfungen. JavaBeans nutzen die Java API, sind somit ebenfalls plattformunabhängig.
Verwendung von Infobus	Die Anwendung soll bis auf die Kernklassen (d.h. bis auf die für die Anwendung zwingend erforderlichen Klassen) aus Softwarekomponenten bestehen. Damit diese Komponenten miteinander interagieren können, ohne direkt voneinander zu wissen, bietet sich die Infobus API an. Der Infobus dient als standardisierte Schnittstelle dem Austausch von Informationen zwischen Softwaremodulen, eine Implementierung in Java liegt vor.
Offene graphische Benutzer- oberfläche	Bis auf die Kernklassen der Anwendung sollen die Module in einem eigenen Fenster untergebracht werden. Dadurch bleibt die Darstellung auch dann konsistent, wenn neue Module, die eine eigene graphische Benutzeroberfläche mitbringen, über den Infobus eingebunden werden.
Verwendung von RMI	Zum Austausch von Modulen über ein Netzwerk wird Remote Method Invocation benutzt. Die Verwendung von CORBA ist geplant, bisher unterstützt das JDK1.2 CORBA nicht vollständig.

2. Architektur

2.1. Überblick

Die Anwendung nutzt die JavaBeans-Technologie von Sun Microsystems, indem sie Teile von Sun's BeanBox übernimmt und andere neu implementiert. Es werden zwei Instanzen der Klasse InfoBus benutzt: MainBus, über den die Kernklassen kommunizieren und BeanBus, über den die Diagrammelemente Daten austauschen können. Um Bean den Zugriff auf Dienste der Kernklassen zu ermöglichen, ist eine Klasse BusBroker vorgesehen, die ähnlich einem Gateway in einem Netzwerk den Datenaustausch zwischen beiden Bussen vornehmen soll.

Die darzustellenden Diagrammelemente sind in Jar-Dateien als Beans abgelegt und werden beim Start des Programms aus dem Verzeichnis "Jars" durch eine Klasse JarLoader geladen. Damit stehen sie dem Benutzer sowohl in einer Werkzeugleiste als auch im Menü der Anwendung zur Auswahl. Durch Hinzufügen von neuen Jar-Dateien mit weiteren Diagrammelementen zum Ordner "Jars" läßt sich die Anwendung um neue Diagrammelemente erweitern. Die darzustellenden Verknüpfungen sind in gleicher Weise organisiert. Sie müssen zusätzlich das Interface Relation implementieren, um in einem BeanContainer als Verknüpfung dargestellt werden zu können.

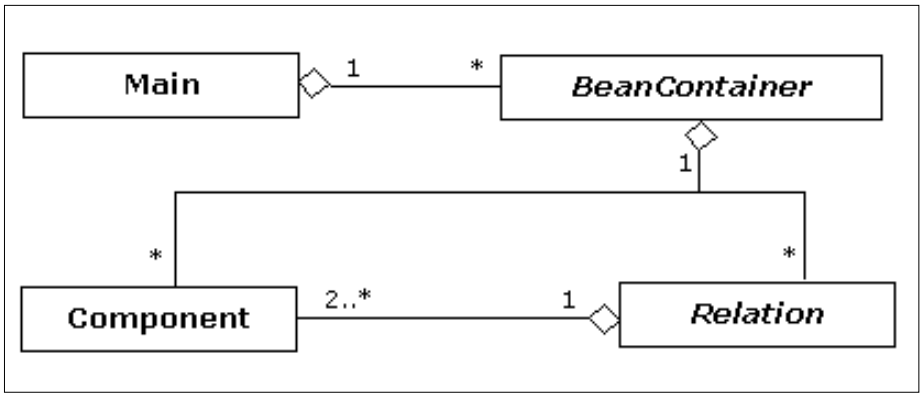


Abb. 4.1. : Architektur-Überblick

Die Klasse Main beinhaltet Referenzen auf eine beliebige Anzahl von BeanContainern. Das Interface BeanContainer definiert eine Klasse für Beans (=Komponenten) aufnehmende Container. Implementierende Klassen legen die Möglichkeiten des Benutzers fest, die Beans in diesem Container zu manipulieren. Der BeanContainer nimmt sowohl Komponenten als auch Verknüpfungen zwischen diesen Komponenten auf. Eine Verknüpfungsklasse wird in dem Interface Relation definiert. Implementierende Klassen legen die spezifischen Eigenschaften der Verknüpfung fest. Die Trennung zwischen Elementen und Verknüpfungen zwischen Elementen war notwendig, da Verknüpfungen eine andere Darstellung innerhalb eines BeanContainers erfordern (z.B. sollen Verknüpfungen keinen rechteckigen Bereich der Zeichenfläche im BeanContainer belegen).

2.2. Datenaustausch über den Infobus

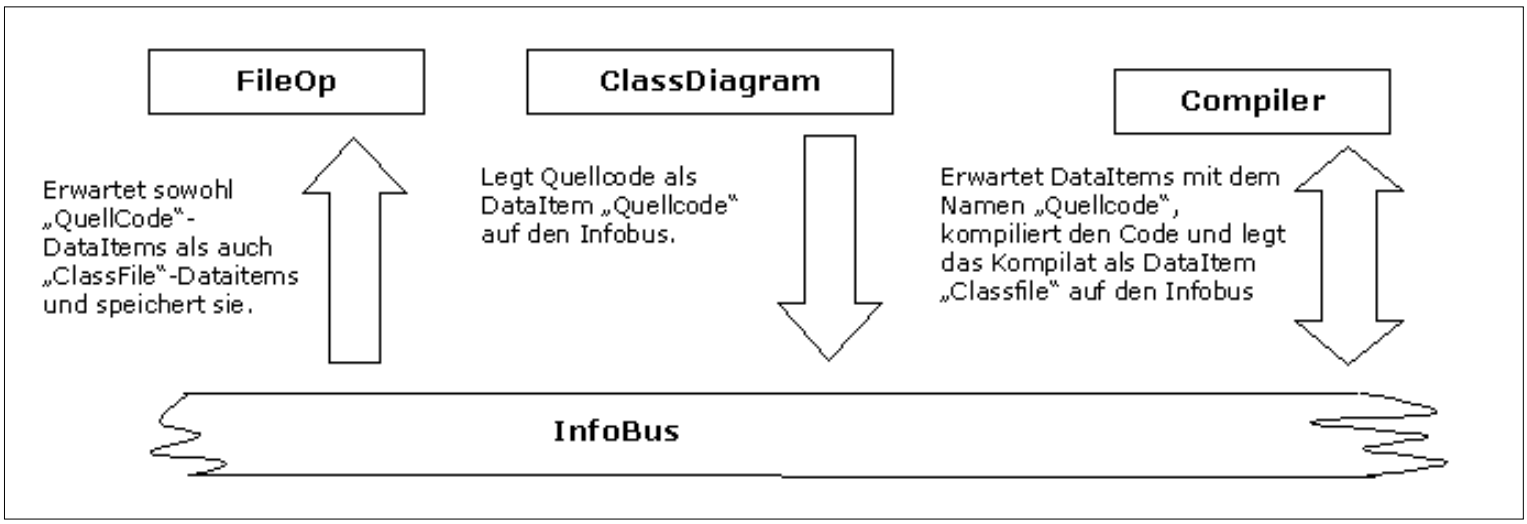


Abb 4.2. Beispiel zum Datenaustausch über den Infobus

Komponenten können über den InfoBus Daten austauschen, ohne voneinander zu wissen. Allein über den Namen des

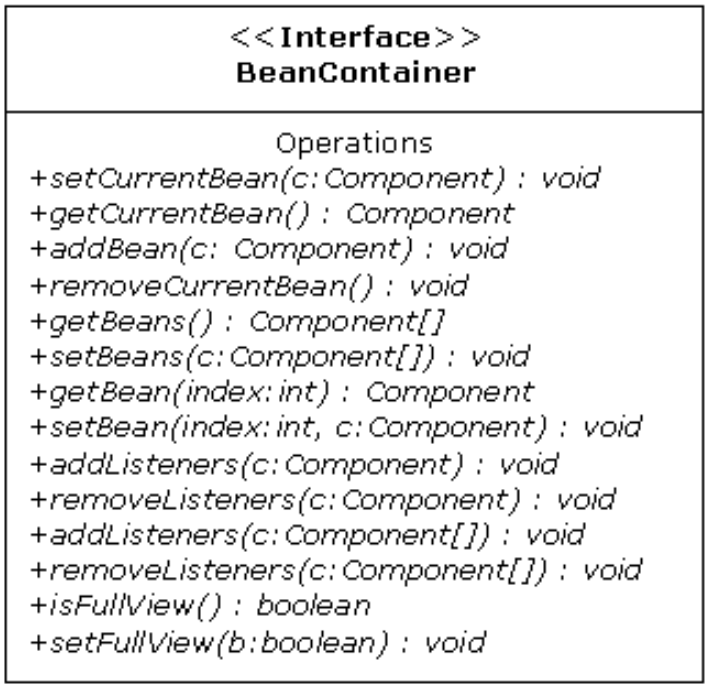
auszutauschenden DataItem muß Einigung bestehen. Die Komponenten, die an diesem Datenaustausch telnehmen möchten, müssen sich bei einer Instanz der Klasse InfoBus registrieren und entweder das Interface DataProducer (im Beispiel übernimmt ClassDiagram diese Rolle) oder das Interface DataConsumer (im Beispiel FileOp) oder beide Interfaces (im Beispiel Compiler) implementieren.

2.3. Einbinden neuer Funktionsmodule

Die Einbindung neuer Funktionsmodule über Remote Method Invocation (RMI) erfolgt in zwei Schritten. Auf Benutzerwunsch stellt die Anwendung eine Anfrage an den Server und empfängt eine Liste verfügbarer Module. Anhand des Modulnamens und einer Versionsnummer schlägt die Anwendung Module zum Download vor. Nach Auswahl der gewünschten Module werden die entsprechenden Objekte empfangen und als serialisierte Klassen gespeichert.

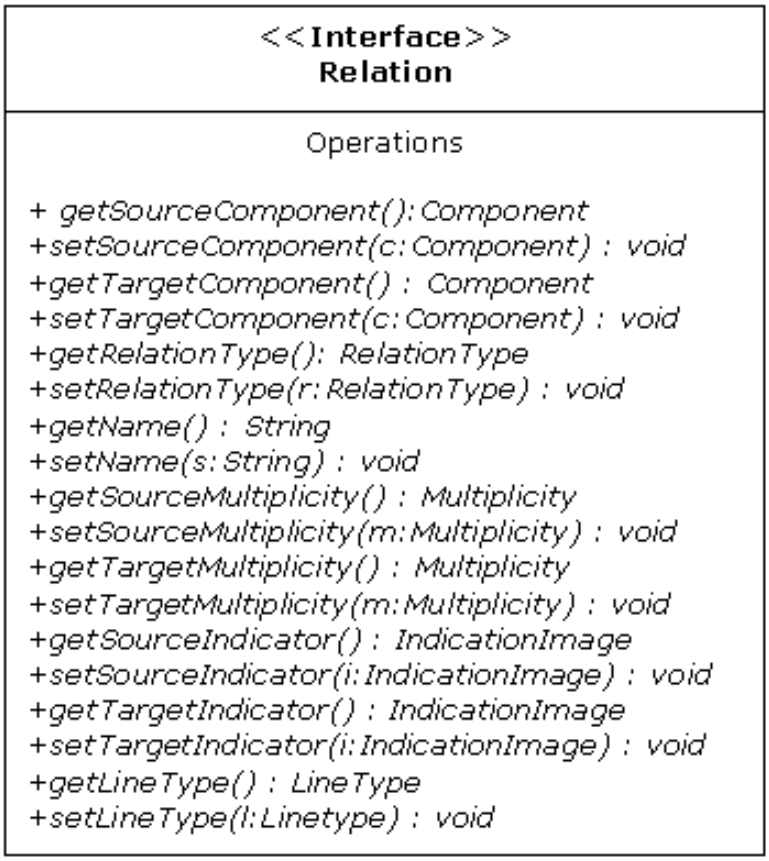
2.4. Das Interface BeanContainer

BeanContainer definiert eine Klasse für Beans aufnehmende Container. Implementierende Klassen legen die Möglichkeiten des Benutzers fest, die Beans in diesem Container zu manipulieren. Implementierende Klassen haben die Möglichkeit, verschiedene Arten von Beans unterschiedlich zu behandeln. Zur Unterscheidung von Beans werden Interfaces herangezogen. Die Klasse BeanPanel, die das Interface BeanContainer implementiert, nutzt z.B. das Interface Relation, um Verknüpfungen zwischen Beans zu erkennen und entsprechend darzustellen.
Der BeanContainer bestimmt zu jeder Zeit eine aktiven Bean, auf die sich weitergehende Aktionen wie z.B. Löschen, Verschieben, Vergrößern beziehen. Darüber hinaus sollten andere Komponenten Zugriff auf alle im BeanContainer enthaltenen Beans haben. Sowohl in visuellen als auch nichtvisuellen Umgebungen sollte der BeanContainer über EventListener mit den Beans verbunden sein, um sie steuern zu können. In visuellen Umgebungen würde z.B. ein Mausklick in eine Bean einen Wechsel der currentBean signalisieren. Es sollten sowohl einzelne Beans mit Listeners ausgestattet werden können als auch alle enthaltenen Beans gemeinsam, z.B. nach einer Entserialisierung des BeanPanel.



2.5. Das Interface Relation

Alle Verknüpfungen implementieren das Interface Relation. Das Interface Relation beinhaltet drei getter/setter-Paare zum Lesen/Schreiben der beiden miteinander verknüpften Komponenten und zum Lesen/Schreiben des Relationstyps.
Die Klasse IndicationImage bildet die Superklasse aller zu zeichnenden Endpunkte einer Verknüpfung. Die ableitenden Klassen sind Components, in deren paint()-Methoden der skalierbare Endpunkt beschrieben wird. Die Klasse LineType bildet die Superklasse aller zu zeichnenden Linien. Innerhalb der paint-methode der ableitenden Klassen wird das Aussehen der Linie beschrieben.



3. Modularität

3.1. Die Manager

Es ist denkbar, das Programm durch Austausch der Manager in ein komplett neues Programm zu transformieren. Allerdings spricht nichts dagegen, dann direkt ein neues Programm zu erstellen. Die Aufteilung in Manager erfolgte im Hinblick auf die Erweiterbarkeit der Anwendung durch weitere Module. Da die Module über den Infobus kommunizieren, muß nur Einigkeit über Name, Format und Zweck der ausgetauschten Datenpakete bestehen. Dann kann ein neues Modul die Funktionen eines bereits Vorhandenen nutzen. Die Schnttstellen der vorhandenen Manager sind in dieser Dokumentation enthalten.

3.2. Die Diagramme, Diagrammelemente und Verknüpfungen

Die Diagrammtypen der Anwendung sind in Jar-Dateien verpackt im Unterverzeichnis "diagrams" untergebracht, die der Diagrammelemente und Verknüpfungen in Jar-Dateien verpackt im Unterverzeichnis "jars". Die Diagrammelemente sind reinrassige Beans, die nur der Voraussetzung genügen müssen, serialisierbar zu sein und von der Klasse Component abzustammen. Die Diagrammtypen sind ebenfalls Beans, haben aber die zusätzliche Einschränkung, das Interface BeanContainer implementieren zu müssen. Dies ist notwendig, damit der DiagramManager die Diagrammtypen als Diagramme und nicht als Diagrammelemente behandelt. Die Verknüpfungen sind Beans, die zusätzlich das Interface Relation implementieren müssen, um sich von den Diagrammelementen abzugrenzen. Verknüpfungen werden im Diagramm z.B. anders dargestellt als Elemente.
Da die verschiedenen Diagrammtypen völlig unabhängig in der Darstellung der Diagrammelemente sind und die Diagrammelemente wiederum Beans, lassen sich beliebige Darstellungen mit beliebigen Elementen realisieren. So wäre es z.B. möglich, durch Implementierung eines neuen Diagrammtyps und neuer Diagrammelemente einen GUI-Builder zu realisieren. Die Verknüpfungen ließen sich dabei als Eventhandler betrachten.

3.3. Beans

Die feinste Granularität ergibt sich auf der Ebene der Beans. Da Beans in sich abgeschlossene Software-Komponenten sind, ist es möglich, in einer Jar-Datei eine Bean gegen eine andere auszutauschen oder zuvor angepaßte Beans in einer Jar-Datei zusammenzufassen. Im ersten Fall kann der Funktionsumfang der Bean erweitert oder modifiziert werden, ohne Einfluß auf die Gesamtanwendung zu nehmen, im zweiten Fall können Beans für den Benutzer auch nach ihrer Kompilierung angepaßt und dann im Programm genauso wie die Originalbeans verwendet werden.

4. Start des Programms

4.1. Laden der Manager

Nachdem Sie das Programm gestartet haben, sucht die Hauptklasse DETs im Unterverzeichnis "managers" nach Klassen, die der Anwendung die gewünschten Funtionen verleihen. Die Hauptklasse schafft die Voraussetzung zur Kommunikation mit den vorhandenen Managern, indem sie einen Kommunikationskanal, den Infobus, einrichtet. Ferner stellt die Hauptklasse eine Menüleiste und eine Werkzeugleiste zur Verfügung, die von den Managern zur Anzeige der von ihnen zur Verfügung gestellten Dienste genutzt werden kann.

Zum Kern der Anwendung gehören der DiagramManager, der die komplette Steuerung der Diagramme übernimmt, der BeanManager, der für das Laden und die Instantiierung der Beans verantwortlich ist, der PropertyManager, der die Anzeige und Veränderung der Eigenschaften der aktiven Bean übernimmt, der FileManager, der das Speichern und Laden von Dateien vom Rest der Anwendung abschirmt und der TextManager, der dem Benutzer die Möglichkeit zur Verfügung stellt, vom Programm produzierten Text zu betrachten oder zu verändern.
Diese Manager sind nur über einen Kommunikationskanal, den sogenannten Infobus, miteinander und der Hauptklasse verbunden.

4.2. Der BeanManager

Der BeanManager ist für das Laden und Instantiieren der Beans verantwortlich. Zu diesem Zweck sucht er im Unterverzeichnis "jars" nach Dateien im Jar-Format und versucht, die darin gefundenen Beans zu laden. Pro gefundener jar-Datei wird ein Untermenü mit gleichem Namen wie die Jar-Datei im Menü "Beans" eingerichtet. Pro gefundener Bean in dieser Jar-Datei wird ein Menüpunkt mit gleichem Namen wie die Bean in diesem Untermenü eingetragen. Sofern die Beans ein Icon zur Verfügung stellen, wird auch ein Eintrag in der Werkzeugleiste vorgenommen.

4.3. Der DiagramManager

Der DiagrammManager übernimmt die komplette Steuerung und Verwaltung der Diagramme. Beim Start sucht der DiagramManager im Unterverzeichnis "diagrams" nach Dateien im Jar-Format. Pro gefundener jar-Datei wird ein Untermenü mit gleichem Namen wie die Jar-Datei im Menü "Diagramme" eingerichtet. Pro gefundener Bean in dieser Jar-Datei wird ein Menüpunkt mit gleichem Namen wie die Bean in diesem Untermenü eingetragen. Sofern die Beans ein Icon zur Verfügung stellen, wird auch ein Eintrag in der Werkzeugleiste vorgenommen. Der Unterschied zum Laden der Beans im BeanManager besteht in der unterschiedlichen Behandlung der Elemente und Diagramme im Programm.

4.4. Der PropertyManager, der FileManager und der TextManager

Diese Manager werden erst zur Laufzeit des Programms aktiv. Zum Start des Programms machen sie ihre Eintragungen in die Menüleiste des Programms, sofern sie Dienste direkt für den Benutzer anbieten.

5. Allgemeine Handhabung des Programms

5.1. Mit Diagrammen arbeiten

5.1.1. Ein neues Diagramm erzeugen

Um ein neues Diagramm in ApartJ zu erzeugen, wählen Sie die entsprechende Diagrammart aus dem Menü Diagramm aus oder drücken den entsprechenden Knopf in der Werkzeugleiste. Es öffnet sich daraufhin ein neues, leeres Fenster. In diesem Fenster können Sie nun die gewünschten Diagrammelemente plazieren und mit Verbindungen verknüpfen.

5.1.2. Ein Diagramm speichern

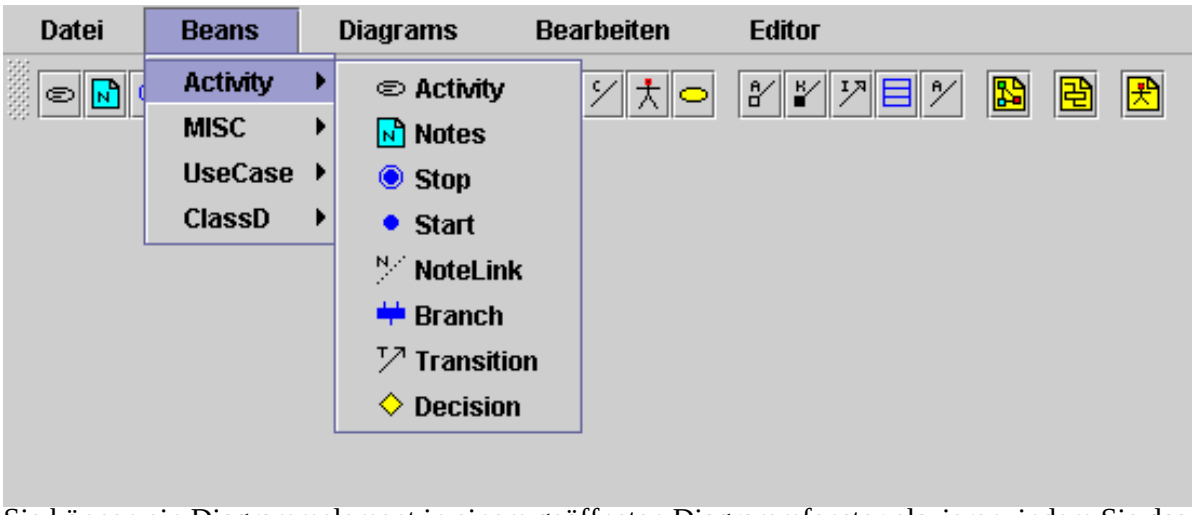
Sie können ein Diagramm in ApartJ speichern, indem Sie in ein geöffnetes Diagramm klicken und es damit zum aktiven Diagramm machen und dann aus dem Menü "Datei" den Menüpunkt "Diagramm speichern" auswählen. Ein Dialogfeld erscheint, in dem Sie Angaben über den Ort und den Namen des zu speichernden Diagramms machen können. Beim Speichern werden die vergebenen Namen automatisch um die Endung ".diag" erweitert.

5.1.3. Ein Diagramm laden

Sie können ein Diagramm in ApartJ laden, indem Sie aus dem Menü "Datei" den Menüpunkt "Diagramm laden" auswählen. In einem erscheinenden Dialogfeld können Sie das Diagramm auswählen, das Sie laden wollen. Diagramme in ApartJ haben die Endung ".diag" .

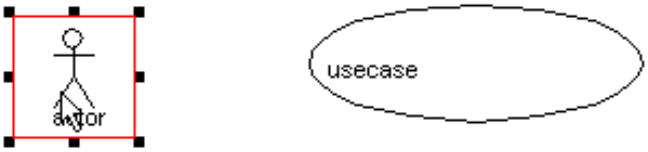
5.2. Mit Diagrammelementen arbeiten

5.2.1. Ein Diagrammelement in einem Diagramm plazieren



Sie können ein Diagrammelement in einem geöffneten Diagrammfenster plazieren, indem Sie das gewünschte Diagrammelement aus einer der Unterkategorien des Menüs "Beans" oder direkt aus der Werkzeugleiste auswählen und danach an der Stelle in einem geöffneten Diagrammfenster linksklicken, an der Sie das Diagrammelement plazieren wollen. Das Diagrammelement erscheint und ist das aktiv ausgewählte Diagrammelement.

5.2.2. Ein Diagrammelement zum aktiven Diagrammelement machen

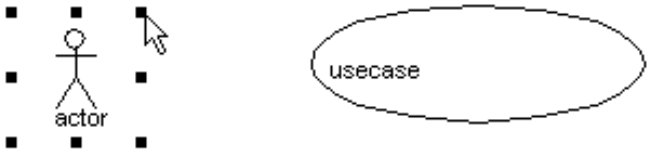


Sie können ein Diagrammelement zum aktiven Diagrammelement machen, indem Sie in dieses Diagrammelement linksklicken. Das aktiv ausgewählte Diagrammelement wird dann durch einen dünnen roten Rahmen und kleine schwarze Kästchen zum Vergrößern und Verkleinern als aktives Diagrammelement markiert.

5.2.3. Ein Diagrammelement im Diagramm verschieben

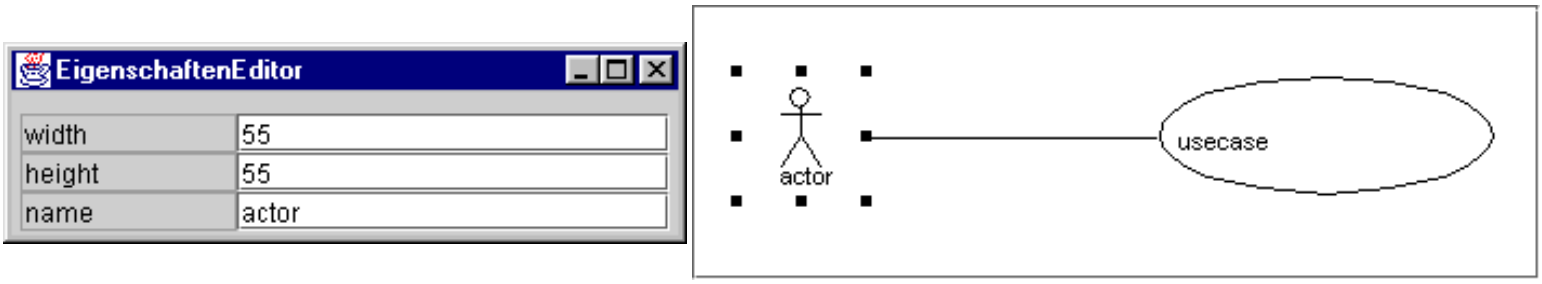
Sie können ein Diagrammelement in einem Diagramm verschieben, indem sie es linksklicken und die Maus gedrückt halten. Solange Sie die Maus gedrückt halten, folgt das Diagrammelement den Bewegungen der Maus. Sobald Sie die Maus loslassen, wird das Diagrammelement an der neuen Stelle plaziert.

5.2.4. Ein Diagrammelement in seiner Größe verändern



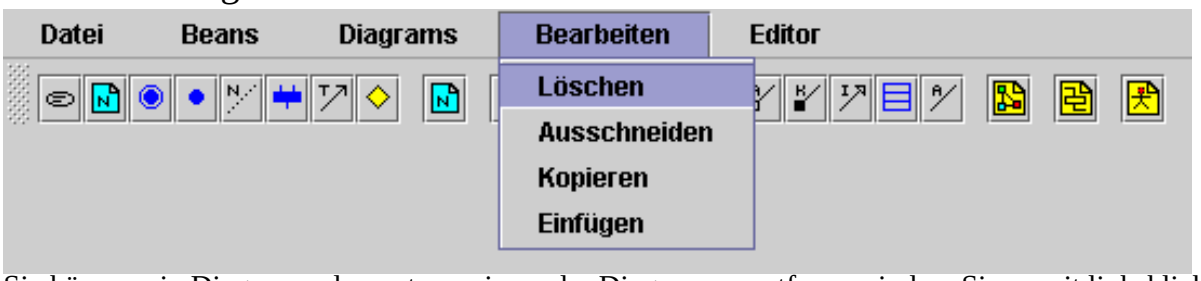
Sie können ein Diagrammelement in seiner Größe verändern, indem Sie es linksklicken und damit zum aktiven Diagrammelement machen. An den Rändern des Diagrammelements erscheinen dann schwarze Kästchen, die dem Ändern der Größe dienen. Befindet sich der Mauszeiger über einem der Kästchen, können Sie anhand des veränderten Mauszeigers feststellen, in welche Richtung Sie das Diagrammelment in seiner Größe verändern können. Durch Drücken und Festhalten der linken Maustaste über einem der Kästchen folgt dieses Kästchen (und damit die Größenveränderung) den Bewegungen der Maus. Sobald Sie die linke Maustaste loslassen, wird die neue Größe übernommen.

5.2.5. Eigenschaften des Diagrammelements anpassen



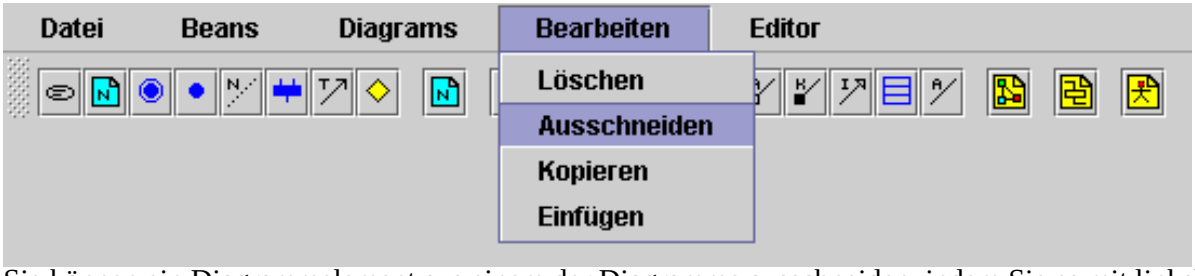
Sie können die Eigenschaften des Diagrammelements entweder direkt im Diagrammelement selbst (z.B. Größe, Ort der Plazierung) oder über den Eigenschafteneditor anpassen. Für weitere Informationen siehe [link Eigenschafteneditor].

5.2.6. Ein Diagrammelement entfernen



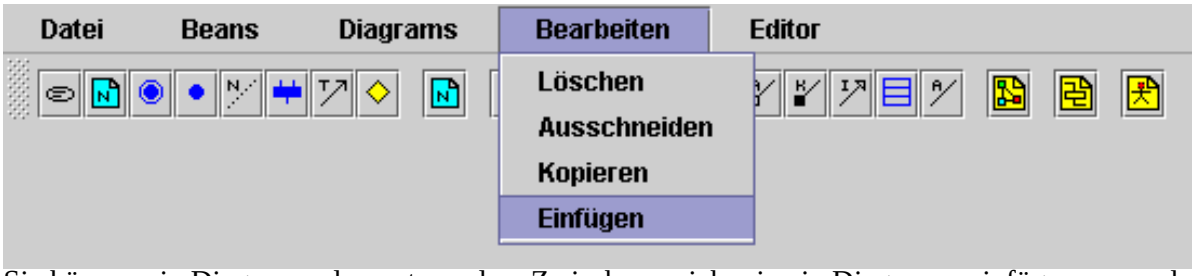
Sie können ein Diagrammelement aus einem der Diagramme entfernen, indem Sie es mit linksklicken und damit zum aktiven Diagrammelement machen und aus dem Menü "Bearbeiten" den Menüpunkt "Löschen" auswählen. Das Diagrammelement wird aus dem Diagramm gelöscht und mit ihm alle Verknüpfungen, die auf es verwiesen haben.

5.2.7. Ein Diagrammelement ausschneiden



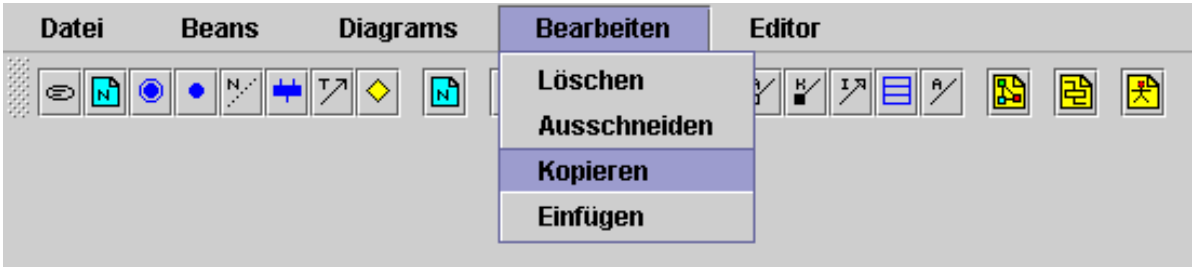
Sie können ein Diagrammelement aus einem der Diagramme ausschneiden, indem Sie es mit linksklicken und damit zum aktiven Diagrammelement machen und aus dem Menü "Bearbeiten" den Menüpunkt "Ausschneiden" auswählen. Das Diagrammelement wird aus dem Diagramm ausgeschnitten und mit ihm alle Verknüpfungen, die auf es verwiesen haben. Die Verknüpfungen, die auf das Diagrammelement verwiesen haben, werden gelöscht, das Diagrammelement wird zwischengespeichert, so daß Sie es durch den Befehl "Einfügen" im Menü "Bearbeiten" erneut in ein Diagramm einfügen können. Es wird solange zwischengespeichert, bis Sie durch erneutes Ausschneiden ein weiteres Diagrammelement zwischenspeichern.

5.2.8. Ein Diagrammelement einfügen



Sie können ein Diagrammelement aus dem Zwischenspeicher in ein Diagramm einfügen, wenn der Zwischenspeicher nicht leer ist, indem Sie aus dem Menü "Bearbeiten" den Punkt "Einfügen" auswählen und dann an der Stelle im gewünschten Diagramm linksklicken, an der das Diagrammelement aus dem Zwischenspeicher plaziert werden soll.

5.2.9 Ein Diagrammelement kopieren



Sie können ein Diagrammelement kopieren, indem indem Sie es linksklicken und damit zum aktiven Diagrammelement machen und aus dem Menü "Bearbeiten" den Menüpunkt "Kopieren" auswählen. Eine exakte Kopie des Diagrammelements wird in den Zwischensspeicher kopiert, so daß Sie es durch den Befehl "Einfügen" aus dem Menü "Bearbeiten" in ein Diagramm einfügen können.

5.2.10 Ein Diagrammelement speichern

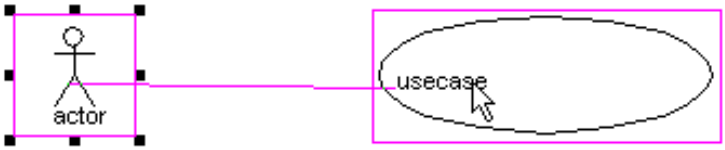
Sie können ein Diagrammelement für spätere Verwendungen auf Ihrer Festpatte speichern, indem Sie das Diagrammelement linksklicken und damit zum aktiven Diagrammelement machen und aus dem Menü "Datei" den Menüpunkt "Bean speichern" auswählen. Ein Dialogfeld erscheint, in dem Sie Angaben über den Ort und den Namen des zu speichernden Diagrammelements machen können. Beim Speichern werden die vergebenen Namen automatisch um die Endung ".bea" erweitert.

5.2.11. Ein Diagrammelement laden

Sie können ein Diagrammelement laden, indem Sie aus dem Menü "Datei" den Menüpunkt "Bean laden" auswählen. Ein Dialogfenster erscheint, in dem Sie das zu ladende Diagrammelement auswählen können. Die Namen der Diagrammelemente haben die Endung ".bea". Nachdem das Diagrammelement geladen ist, können Sie es jetzt durch Linksklicken in ein Diagramm in diesem Diagramm plazieren.

5.3. Mit Verknüpfungen arbeiten

5.3.1. Eine Verknüpfung in ein Diagramm einfügen



Sie können eine Verknüpfung zwischen zwei Diagrammelementen in ein Diagramm einfügen, indem Sie die Verknüpfung aus einer der Unterkategorien des Menüs "Beans" oder direkt aus der Werkzeugleiste auswählen. Nun linksklicken Sie das Diagrammelement, das als Quelle der Verknüpfung dienen soll und halten die Maustaste gedrückt. Das Diagrammelement wird mit einem violetten Rahmen versehen. Solange Sie die linke Maustaste gedrückt halten, folgt ein violettes Band ihrem Mauszeiger, um anzuzeigen, wo die Verknüpfung gezogen wird. Wenn Sie nun bei gedrückter linker Maustaste den Mauszeiger über ein Diagrammelement bewegen, welches als Ziel der Verknüpfung dienen kann, wird dieses Diagrammelement ebenfalls mit einem violetten Rahmen versehen. Wenn Sie nun die linke Maustaste loslassen, wird die gewünschte Verknüpfung zwischen den beiden ausgewählten Diagrammelementen gezeichnet.

5.3.2. Eine Verknüpfung zur aktiven Verknüpfung machen

Sie können eine Verknüpfung zur aktiven Verknüpfung machen, indem Sie neben die Linie der Verknüpfung linksklicken. Die Verknüpfung wird in rot gezeichnet, um anzuzeigen, daß diese Verknüpfung jetzt die aktive Verknüpfung ist. Im Gegensatz zu Diagrammelementen wird die aktive Verknüpfung nicht mit schwarzen Kästchen zur Veränderung der Größe ausgestattet, da die Größe der Verknüpfung auf der Größe und Position der verbundenen Diagrammelemente beruht.

5.3.3. Verlauf einer Verknüpfung anpassen durch Aufspaltung der Linie

Sie können den Verlauf einer Verknüpfung anpassen, indem Sie die Maus über die Linie der Verknüpfung bewegen, bis sich der Mauszeiger in ein Fadenkreuz verwandelt. Wenn Sie nun linksklicken, wird die Linie der Verknüpfung gespalten und ein quadratischer Knickpunkt erscheint an dem Punkt auf der Linie, an dem Sie geklickt haben. Durch Drücken und Festhalten der linken Maustaste, während Sie sich über diesem Knickpunkt befinden, folgt der Knickpunkt den Bewegungen des Mauszeigers. Sobald Sie die linke Maustaste loslassen, wird die veränderte Linie plaziert.

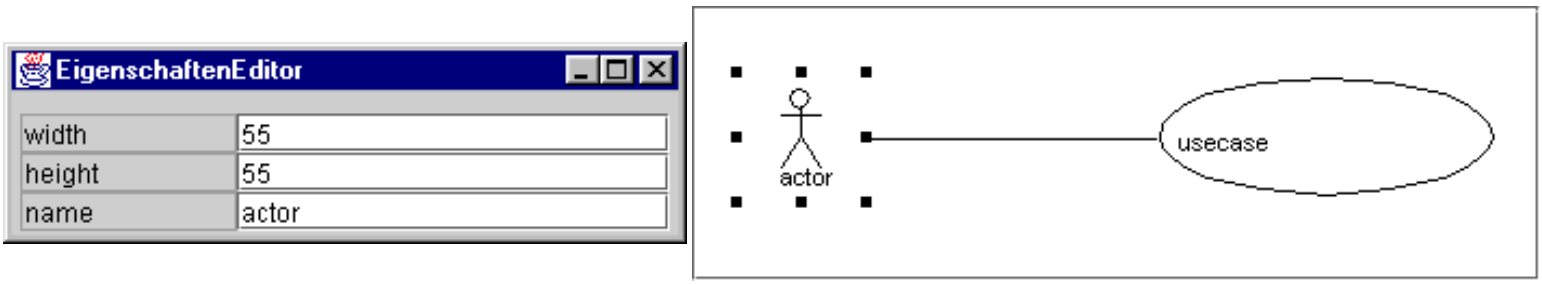
5.3.4. Verlauf einer Verknüpfung anpassen durch Zusammenführung von Liniensegmenten

Sie können die von Ihnen gesetzten Knickpunkte in der Linie einer Verknüpfung aus der Linie entfernen, indem Sie über einem Knickpunkt die rechte Maustaste drücken. Die Linie wird dabei automatich angepaßt.

5.3.5. Verhalten einer Verknüpfung bei Veränderung des Orts oder der Größe der verknüpften Diagrammelemente

Eine Verknüpfung paßt sich bei Veränderung des Orts oder der Größe der miteinander verknüpften Diagrammelemente automatisch den geänderten Verhältnissen an.

5.4. Mit dem Eigenschafteneditor arbeiten



Der Eigenschafteneditor zeigt die anpaßbaren Eigenschaften des akiven Diagrammelements bzw. der aktiven Verknüpfungen an. Im folgenden werden sowohl Diagrammelemente als auch Verknüpfungen unter dem Oberbegriff Beans zusammengefaßt. Um die Eigenschaften einer Bean anzupassen, wählt man die zu ändernde Eigenschaft aus den vorhandenen aus und ändert deren Wert auf der rechten Seite. Je nach Art der Eigenschaft können sich Dialoge öffnen, die als Eingabe weitere Informationen erwarten.

6. Download des Quelltexts und eigenes Kompilieren

6.1. Voraussetzungen

Voraussetzung zum Kompilieren und Funktionieren der Anwendung ist eine funktionierende Installation des JDK1.2. Die Anwendung läuft nicht mit JDK1.1.* und dem Swing-Paket!!!

6.2. Download

Klicken Sie hier, um sich die aktuelle Version des Quelltexts herunterzuladen. Sie können den Quelltext sowohl als tar-Datei (Linux) als auch als selbstextrahierende zip-Datei (Windows) herunterladen.

[Button Win] [Button Linux] (Noch in Arbeit)

6.2. Entpacken des Archivs

Wechseln Sie in das Verzeichnis, in das Sie das Archiv entpacken möchten und geben Sie unter Linux folgenden Befehl in die Shell ein:
tar -xvzf DET.tar.gz
Unter Windows starten Sie die selbstextrahierende Datei und folgen Sie den Anweisungen auf dem Bildschirm.

Im aktuellen bzw. angegebenen Verzeichnis wird das Verzeichnis DET erstellt, in dem sich die Unterverzeichnisse "beans", "classes" und "source" befinden. Im Verzeichnis "source" befinden sich die Quellen der Anwendung, im Verzeichnis "beans" befinden sich die Quellen der verwendeten Beans in eigenen Unterverzeichnissen zusammen mit den jeweils benötigten Ressourcen, das Verzeichnis classes sollte bis auf die leeren Verzeichnisse diagrams und jars leer sein.

6.3. Kompilieren der Anwendung

Da die Anwendung modular aufgebaut ist, sind mehrere Schritte notwendig, um die Gesamtanwendung zu kompilieren. Verwenden Sie deshalb das für Ihre Plattform passende Skript aus dem Verzeichnis "source". Für Windows sollte das Skript mit dem Namen "compile.bat" verwendet werden, für Linux das Skript mit dem Namen "compile.sh".

6.4. Starten des Programms

Wechseln Sie in das Verzeichnis "classes" und starten sie das für Ihre Plattform passende Startskript, für Windows "run.bat", für Linux "run.sh".