

# BlueJ und Subversion

## Inhaltsverzeichnis

Teil 1 – Auschecken.....	3
Teamarbeits-Menü einschalten.....	3
Erstmalig ein BlueJ-Projekt aus einem Repository auschecken.....	3
Teil 2 – Erstes Arbeiten.....	5
Benutzername und Passwort eingeben.....	5
Aktualisieren (update) und Abgeben (commit).....	5
Grundsätzliche Reihenfolge beim Arbeiten.....	7
Teil 3 – Konflikte.....	8
Teil 4 – Der Server.....	10
Teil 5 – Workflow für die Lehrkraft.....	12
Tortoise SVN – Update, Commit.....	12
Tortoise SVN – Eine lokale Kopie des Repository.....	14
Teil 6 – Erfahrungen.....	15
Rückmeldung aus der Praxis.....	15
Typische Probleme.....	15
Teil 7 – Beispiele.....	17
Einfaches Tic-Tac-Toe mit Model-View-Controller.....	17
Größeres Programmierprojekt.....	18

Alles Material unter: <https://ddi.ifi.lmu.de/tdi/2016/svn-bluej>



# Teil 1 – Auschecken

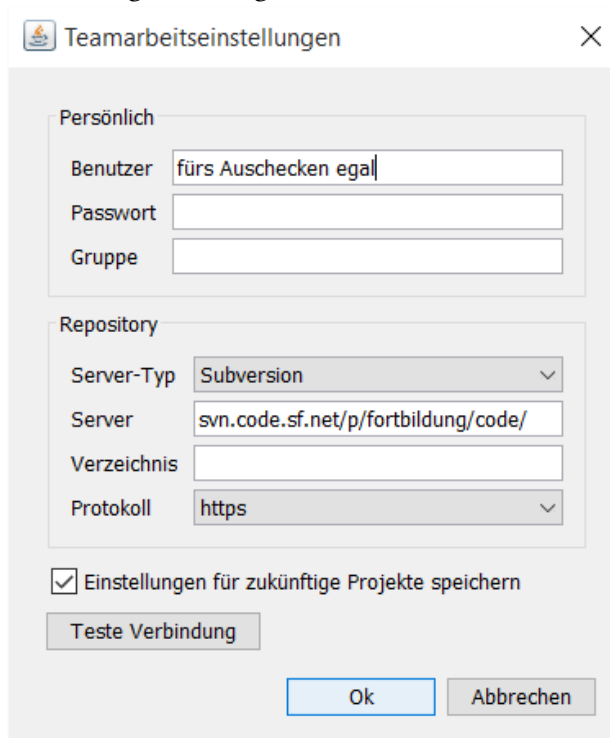
## Teamarbeits-Menü einschalten

Bevor man den in BlueJ integrierten SVN-Client benutzen kann, muss man ihn freischalten. Dazu muss man im Menü unter **„Werkzeuge > Einstellungen... > Interface“** ein Häkchen setzen bei „Teamarbeitswerkzeuge anzeigen“.

## Erstmalig ein BlueJ-Projekt aus einem Repository auschecken

Es ist egal, ob man an dieser Stelle mit einem bereits geöffneten BlueJ-Projekt arbeitet oder nicht, die neu heruntergeladenen Dateien werden auf jeden Fall als neues, eigenes BlueJ-Projekt gespeichert.

Nach Auswahl des Menüpunkts **„Werkzeuge > Teamarbeit > Arbeitskopie erstellen...“** muss man die Serverangaben eingeben:

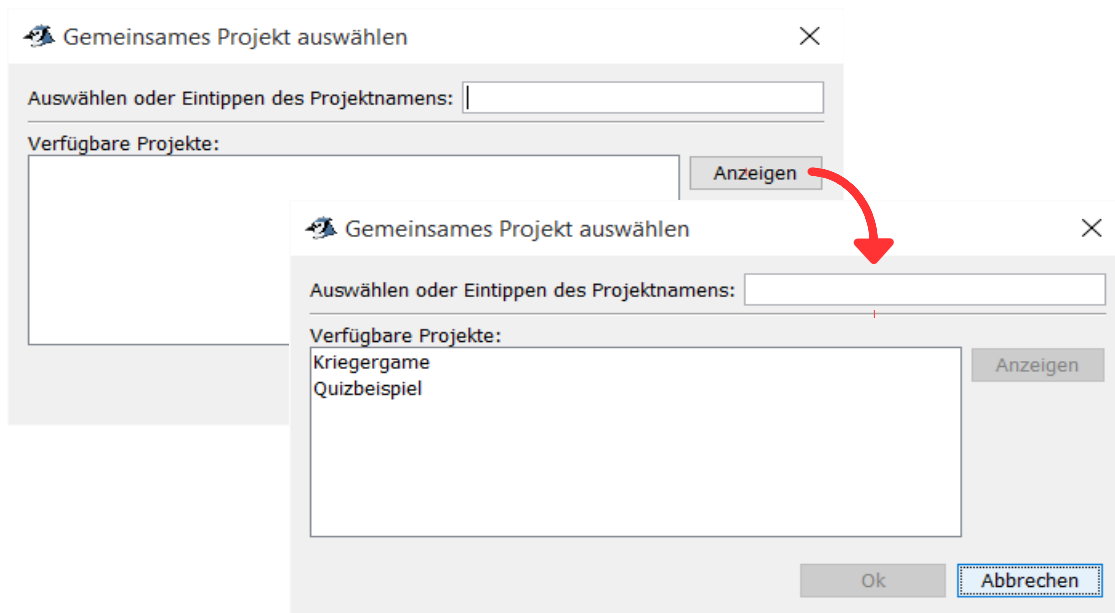


The screenshot shows the 'Teamarbeitseinstellungen' (Teamwork Settings) dialog box. It is divided into two main sections: 'Persönlich' (Personal) and 'Repository'. In the 'Persönlich' section, the 'Benutzer' (User) field is filled with 'fürs Auschecken egal', while the 'Passwort' (Password) and 'Gruppe' (Group) fields are empty. In the 'Repository' section, the 'Server-Typ' (Server Type) is set to 'Subversion', the 'Server' field contains 'svn.code.sf.net/p/fortbildung/code/', the 'Verzeichnis' (Directory) field is empty, and the 'Protokoll' (Protocol) is set to 'https'. Below these fields, there is a checkbox labeled 'Einstellungen für zukünftige Projekte speichern' (Save settings for future projects) which is checked. At the bottom of the dialog, there is a 'Teste Verbindung' (Test Connection) button and 'Ok' and 'Abbrechen' (Cancel) buttons.

Das Passwort braucht man bei öffentlichen Repositories nur für das Hochladen; allerdings verlangt BlueJ auf jeden Fall die Eingabe eines (auch beliebigen) Benutzernamens.

Danach kann man sich mit dem Knopf „Anzeigen“ die unter dieser Adresse gespei-

cherten BlueJ-Projekte zeigen lassen:



Nach der Auswahl des BlueJ-Projekts, das man herunterladen möchte, wird man gebeten, einen Speicherort dafür anzugeben. Das heißt, die Dateien werden auf jeden Fall in einem neuen BlueJ-Projekt gespeichert.

Das Auschecken erfolgt üblicherweise nur einmal am Anfang. Danach lädt man nicht mehr das ganze Projekt neu vom Server herunter, sondern aktualisiert nur die auf dem Server geänderten Dateien oder fügt eigene Änderungen der Version auf dem Server hinzu.

Ausnahme: Wenn später einmal irgendetwas mit dem eigenen BlueJ-Projekt nicht funktioniert oder zu kompliziert wird, dann ist es manchmal einfacher, die Dateien wieder ganz neu vom Server auszuchecken und mit einem neuen BlueJ-Projekt weiterzuarbeiten. Man muss nur daran denken, das alte zu löschen.

Fehlerquelle: Beim Speichern reicht ein Laufwerksbuchstabe nicht, man muss einen Ordner angeben, in den dann der BlueJ-Projektordner gespeichert wird. Wenn man den Namen eines bereits existierenden BlueJ-Projekts angibt, kann das dazu führen, dass ein BlueJ-Projekt innerhalb eines anderen gespeichert wird (als *package*), was alles nur komplizierter macht.

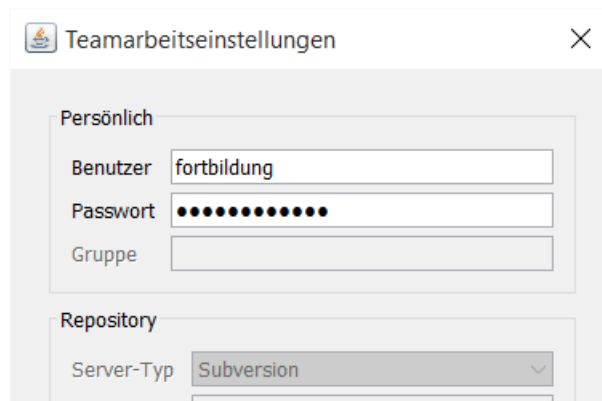
**Aufgabe:** Checken Sie beliebige Projekte aus dem Server aus, aber auf jeden Fall das BlueJ-Projekt „SVN Aufgabe 1“ für den nächsten Schritt.

# Teil 2 – Erstes Arbeiten

## Benutzername und Passwort eingeben

Checken Sie das BlueJ-Projekt „SVN Aufgabe 1“ aus.

Bisher waren Benutzername und Passwort egal, aber ab diesem Zeitpunkt müssen Sie konkrete Daten eingeben. Wählen Sie dazu im Menü **„Werkzeuge > Teamarbeit > Teamarbeitseinstellungen...“** und ergänzen Sie die folgenden Informationen:

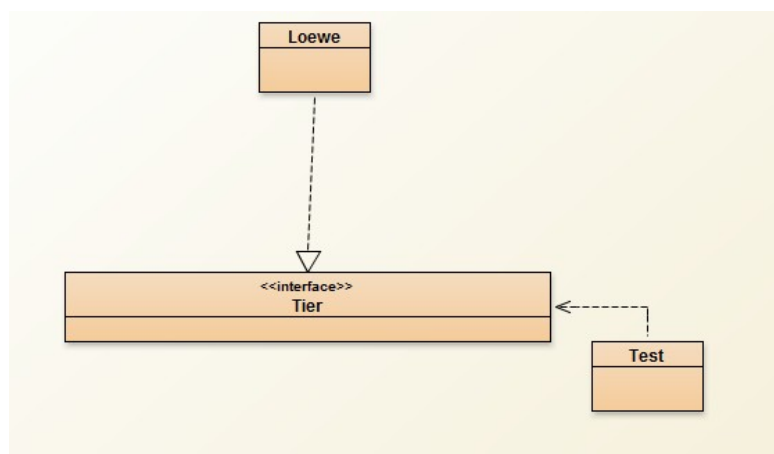


The screenshot shows the 'Teamarbeitseinstellungen' (Teamwork Settings) dialog box. It has two main sections: 'Persönlich' (Personal) and 'Repository'. In the 'Persönlich' section, the 'Benutzer' (User) field contains 'fortbildung', the 'Passwort' (Password) field is masked with dots, and the 'Gruppe' (Group) field is empty. In the 'Repository' section, the 'Server-Typ' (Server Type) dropdown menu is set to 'Subversion'.

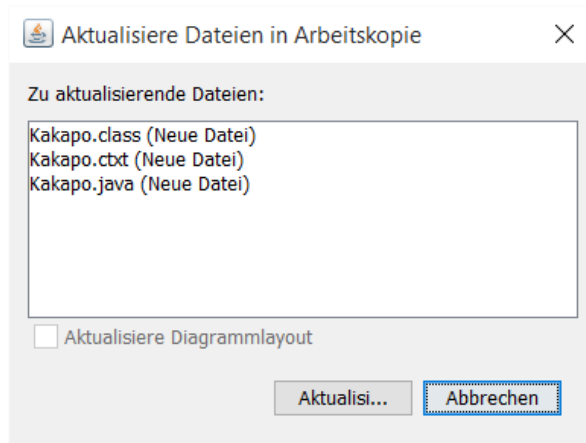
Der Benutzer ist „fortbildung“, das Passwort erfahren Sie mündlich.

## Aktualisieren (update) und Abgeben (commit)

Aufgabe: Legen Sie in „SVN Aufgabe 1“ eine neue Klasse an, die das Interface „Tier“ implementiert. Die Klasse „Loewe“ kann als Beispiel dienen. Sie machen es sich etwas einfacher, wenn Sie ein exotisches Tier wählen, weil Sie dadurch Konflikte mit anderen Projektmitarbeitern vermeiden.

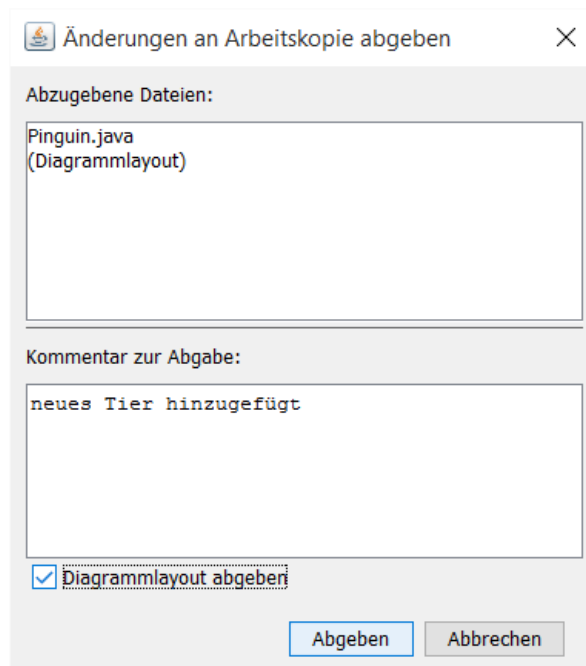


Schritt 1: Bevor Sie Ihre Klasse hochladen, sollten Sie schauen, ob es inzwischen vielleicht schon Beiträge anderer Projektmitarbeiter gibt. Wählen Sie dazu im seitlichen Menü unten „**Aktualisieren...**“ (englisch: „**Update...**“). Wenn inzwischen jemand anderes die Klasse „Kakapo“ angelegt hat (ein flugunfähiger Papagei auf Neuseeland), wird Ihnen folgender Dialog angezeigt:



Wenn Sie das Aktualisieren bestätigen, wird der Kakapo Ihrem BlueJ-Projekt hinzugefügt.

Schritt 2: Laden Sie jetzt Ihre eigenen Änderungen (Ihre Klasse „Pinguin“ zum Beispiel) hoch. Wählen Sie dazu im seitlichen Menü unten „**Abgeben...**“ oder „**Commit...**“. Darauf erscheint eine Liste der von Ihnen geänderten Dateien. Fügen Sie einen Kommentar hinzu und geben Sie die Änderungen ab:

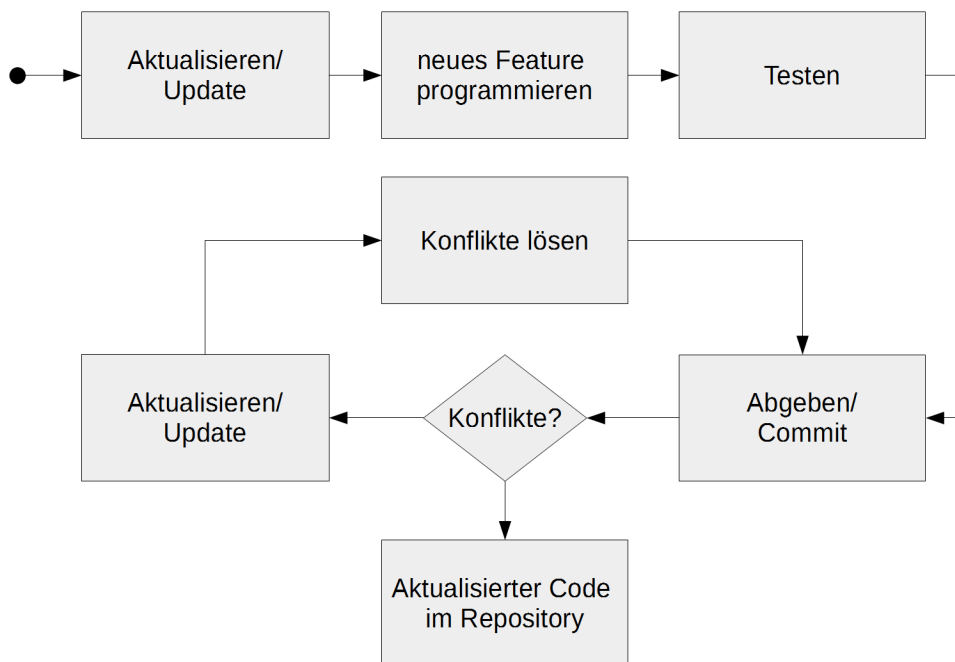


**Aufgabe: Führen Sie einige Updates und Commits durch. Wenn Sie auf Konflikte stoßen, ignorieren Sie sie bis zum nächsten Schritt.**

## Grundsätzliche Reihenfolge beim Arbeiten

Nach dem erstmaligen Auschecken des Projekts sieht das Arbeiten immer so aus:

1. Update, um zu sehen, ob sich am Server etwas geändert hat.
2. Eigenen Code ergänzen und testen.
3. Commit – der findet nur statt, wenn es keine Konflikte mehr gibt.



### Hilfreiche Regeln fürs Arbeiten mit Schülerinnen und Schülern:

1. Einen Commit nur dann durchführen, wenn alle Klassen im BlueJ-Projekt fehlerfrei übersetzt werden.
2. Um Konflikte zu vermeiden, sollte klar sein, welches Team oder welche einzelne Schülerin für welche Klasse oder Klassen zuständig sind. Und die Schüler müssen die Finger von den Klassen anderer Schüler lassen.

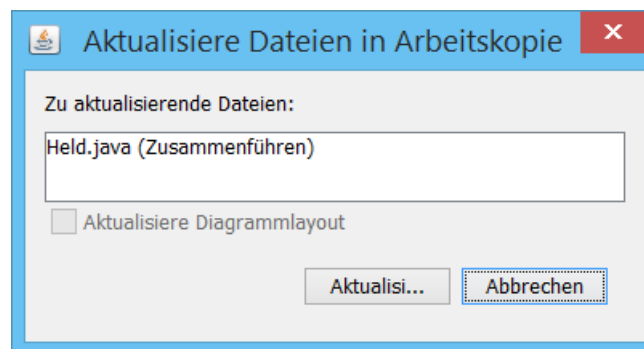
Wenn eine Datei ganz gelöscht wurde, kann sie *mit BlueJ* nicht mehr aus dem Repository ergänzt werden. Man muss dann das Projekt komplett neu auschecken und damit weitermachen, oder die fehlende Datei von dort ergänzen.

## Teil 3 – Konflikte

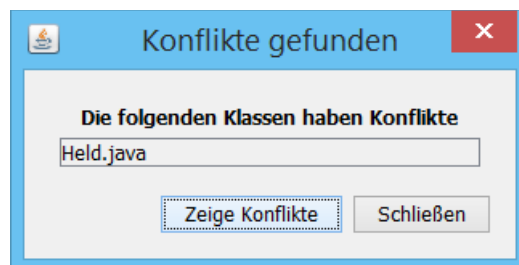
Wenn mehrere Bearbeiter unabhängig von einander an einer Klasse arbeiten, kommt es zu einem Konflikt. Das SVN-System kann einfache Konflikte selbstständig auflösen, bei den meisten muss aber ein Benutzer entscheiden, welche der in Konflikt stehenden Versionen gelten soll.

**Aufgabe:** Checken Sie das Projekt „SVN Aufgabe 2“ aus und ergänzen Sie *eine* noch nicht vorhandene getter- *oder* setter-Methode für ein Attribut der Klasse „Held“. Führen Sie dann einen Commit durch (bzw. vorher ein eventuell eingefordertes Update).

Fall 1: Die entstehenden Konflikte lassen sich automatisch lösen, weil jede Änderung nur eine eigene, separate Methode betrifft. Dann zeigt BlueJ – nach dem Aktualisieren – folgende Meldung, mit der man das automatische Zusammenführen bestätigen kann:



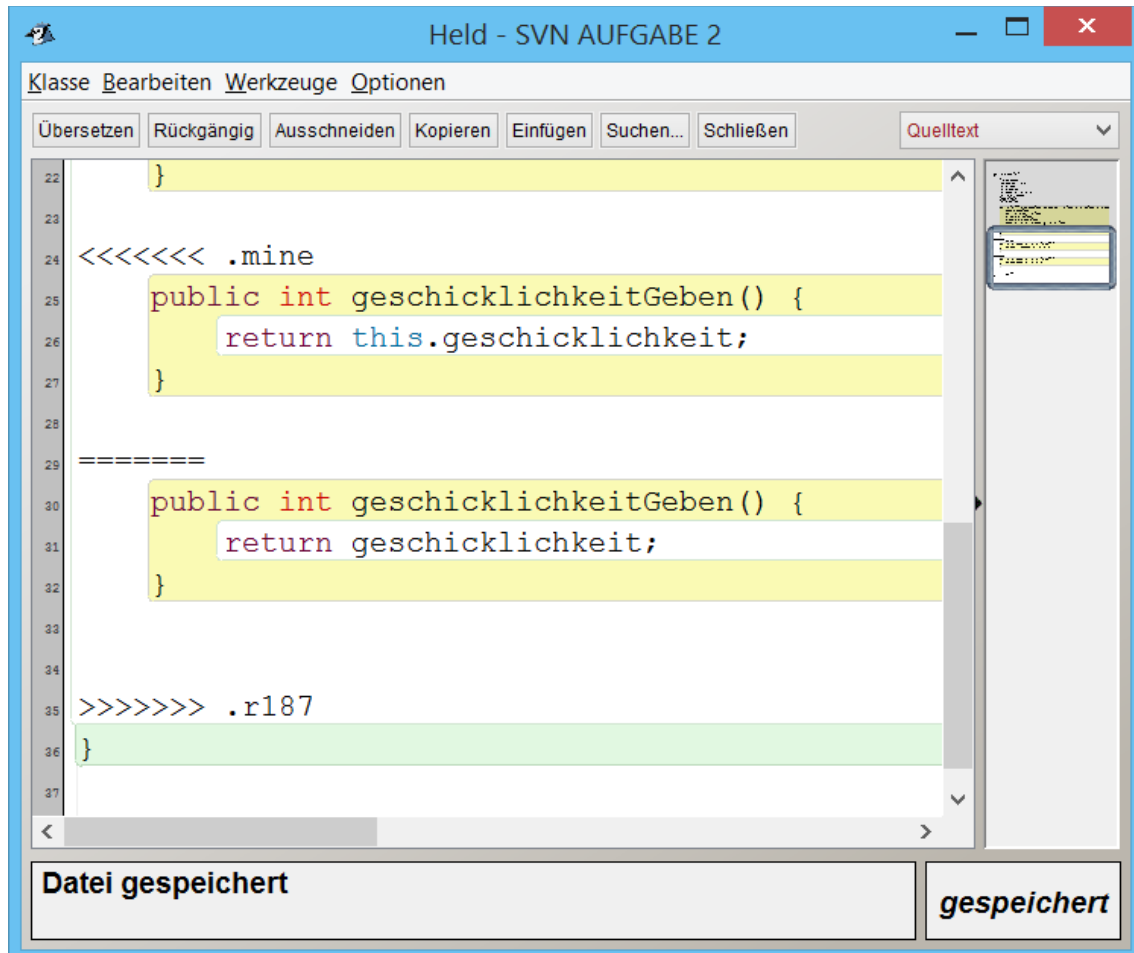
Fall 2: Der Konflikt lässt sich nicht automatisch lösen, weil Attribute der Klasse betroffen sind oder die Änderungen *eine* Methode betreffen. Dann meldet BlueJ den Konflikt so:



Daraufhin lässt man sich die Konflikte anzeigen. Die in Konflikt stehenden Codestellen werden beide in einem BlueJ-Editorfenster angezeigt. Von den Zeichen <<<<<< bis



===== steht die eine Fassung (die des aktuellen Benutzers), von den Zeichen ===== bis >>>>>> steht die andere Fassung (die aktuell auf dem Server befindliche). Der aktuelle Benutzer muss den Konflikt manuell lösen.



Im Rahmen eines Workshops sind Konflikte schwer zu lösen: Wenn es einen Konflikt gibt, und mehrere Teilnehmer versuchen gleichzeitig, ihn zu lösen, gibt das nur noch mehr Durcheinander.

Beim Arbeiten mit Schülerinnen und Schülern ist es besser, wenn sich die Gruppen auf einzelne Klassen spezialisieren, um Konflikte zu vermeiden.









# Teil 4 – Der Server

Man kann einen SVN-Server lokal im Netz installieren (etwa den Visual SVN Server <https://www.visualsvn.com/server/> für Windows), aber eigentlich möchte man einen über das WWW erreichbaren Server. Dafür gibt es verschiedene kostenlose Anbieter, die aber alle das Problem haben, dass a) die Projekte öffentlich einsehbar sind (Vorteil: Keine Registrierung zum Aktualisieren nötig, Nachteil: Urheberrecht und Datenschutz) und b) die Projektarbeiter alle eine Registrierung brauchen, wenn sie Commits durchführen sollen.

Unsere Projekte sind bei sourceforge.net gehostet. Ein SourceForge-Projekt kann man allerdings nicht so einfach wieder löschen, man muss eine Nachricht an einen Administrator schicken, der die Löschung einleitet. Die *Inhalte* des SourceForge-Projekts, also die BlueJ-Projekte, kann man natürlich jederzeit verändern.

Vorgehen:

1. Registrierung bei SourceForge (einmalig) unter Angabe einer E-Mail-Adresse und eines Benutzernamens; das Passwort kann frei gewählt werden. Werden für Schüler Dummy-Accounts erstellt, besteht natürlich die Möglichkeit, dass sie E-Mail und Passwort ändern.
2. Anmelden bei SourceForge und Anlegen eines neuen Projekts. Dabei enthält die voreingestellte Auswahl als Versionskontrolle *nicht* Subversion, sondern das jüngere Git. Außerdem kann man wählen, ob man ein Wiki, ein Forum, ein Ticketsystem oder andere Features haben möchte:

<input checked="" type="checkbox"/> <b>Wiki</b>  Documentation is key to your project and the wiki tool helps make it easy for anyone to contribute.	<input type="checkbox"/> <b>Files &amp; Stats</b>  Use the largest free, managed, global mirror network to distribute your files, and follow the download trends that enable you to develop better software.
<input type="checkbox"/> <b>Git</b>  Git is a distributed version control system designed to handle everything from small to very large projects with speed and efficiency.	<input checked="" type="checkbox"/> <b>Tickets</b>  Bugs, enhancements, tasks, etc., will help you plan and manage your development.
<input checked="" type="checkbox"/> <b>Forums</b>  Collaborate with your community in your forum.	<input type="checkbox"/> <b>Blog</b>  Share exciting news and progress updates with your community.
<input checked="" type="checkbox"/> <b>SVN</b>  Enterprise-class centralized version control for the masses.	<input type="checkbox"/> <b>Mercurial</b>  Mercurial is a distributed source control management tool that efficiently handles projects of any size and offers an easy and intuitive interface.

3. Das SourceForge-Projekt ist zu Beginn ganz leer. Üblicherweise gibt es bei Subversion auf der obersten Ebene die Ordner „branches“, „tags“ und „trunk“; nur in letzterem werden die aktuellen Dateien gespeichert. Die anderen Ordner

betreffen Subversion-Features, die von BlueJ nicht genutzt werden. Man kann also auf sie verzichten. Dann muss man nur:

1. Ein neues BlueJ-Projekt anlegen.
2. **„Werkzeuge > Teamarbeit > Projekt gemeinsam nutzen...“** auswählen, Benutzername, Passwort und Serveradresse des SourceForge-Projekts eingeben. Das war's schon.
4. Will man dennoch mit den Ordnern „branches“, „tags“ und „trunk“ arbeiten, etwa um in Zukunft die Möglichkeiten anderer Subversion-Clients zu nutzen, muss man einen externen SVN-Client benutzen; mit BlueJ alleine geht das nicht.
  1. Kommandozeilen-SVN-Client bei Linux: Die bei SourceForge angegebenen Befehle eingeben.
  2. Mit einem anderen externen SVN-Client (etwa TortoiseSVN) eine temporäre lokale Kopie des – vorerst noch leeren – Repository anlegen. Dann die drei Order lokal anlegen und auf den Server hochladen („commit“). laden. Danach wird die temporäre Kopie gelöscht; die Ordner werden nur auf dem Server gebraucht.

**Aufgabe:** Legen Sie ein neues BlueJ-Projekt an und fügen Sie es mit **„Werkzeuge > Teamarbeit > Projekt gemeinsam nutzen...“** dem SourceForge-Projekt `svn.code.sf.net/p/fortbildung/code/` hinzu.

**Jeder Projektmitarbeiter braucht einen eigenen Benutzernamen und ein eigenes Passwort, eventuell Dummy-Accounts für Schüler anlegen.**

**Ein möglichst kurzer Projektname erleichtert das Eingeben der Serveradresse in BlueJ.**

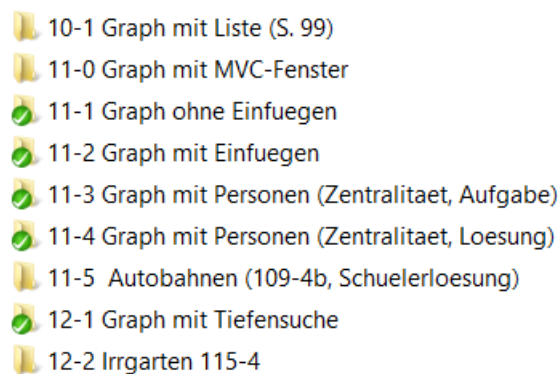
**Eine im Repository angelegte Datei (also auch Code) kann man nicht einfach wieder löschen. Grundsätzlich bleiben *alle* bisherigen Versionen von Textdateien erhalten – man soll bei Subversion auch auf frühere Versionen zurückgreifen können soll, auch wenn BlueJ diese Funktion nicht unterstützt.**

# Teil 5 – Workflow für die Lehrkraft

## Tortoise SVN – Update, Commit

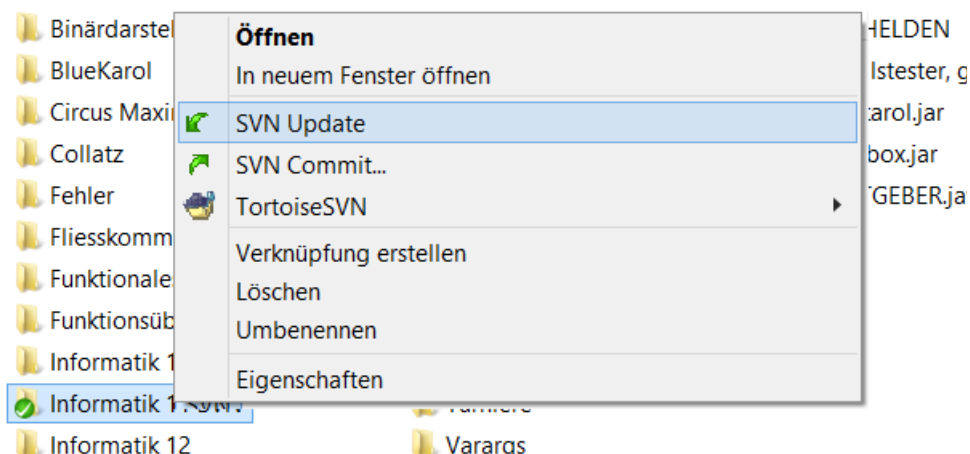
Wenn man als Lehrkraft mehrerer Projekte verwalten möchte, ist der in BlueJ integrierte SVN-Client umständlich, es empfiehlt sich die Verwendung eines eigenen SVN-Clients, für Windows z.B. TortoiseSVN (<http://tortoisesvn.net/>).

Damit kann man z.B. einen Ordner mit verschiedenen gesammelten BlueJ-Projekten mit einem SVN-Repository verknüpfen.

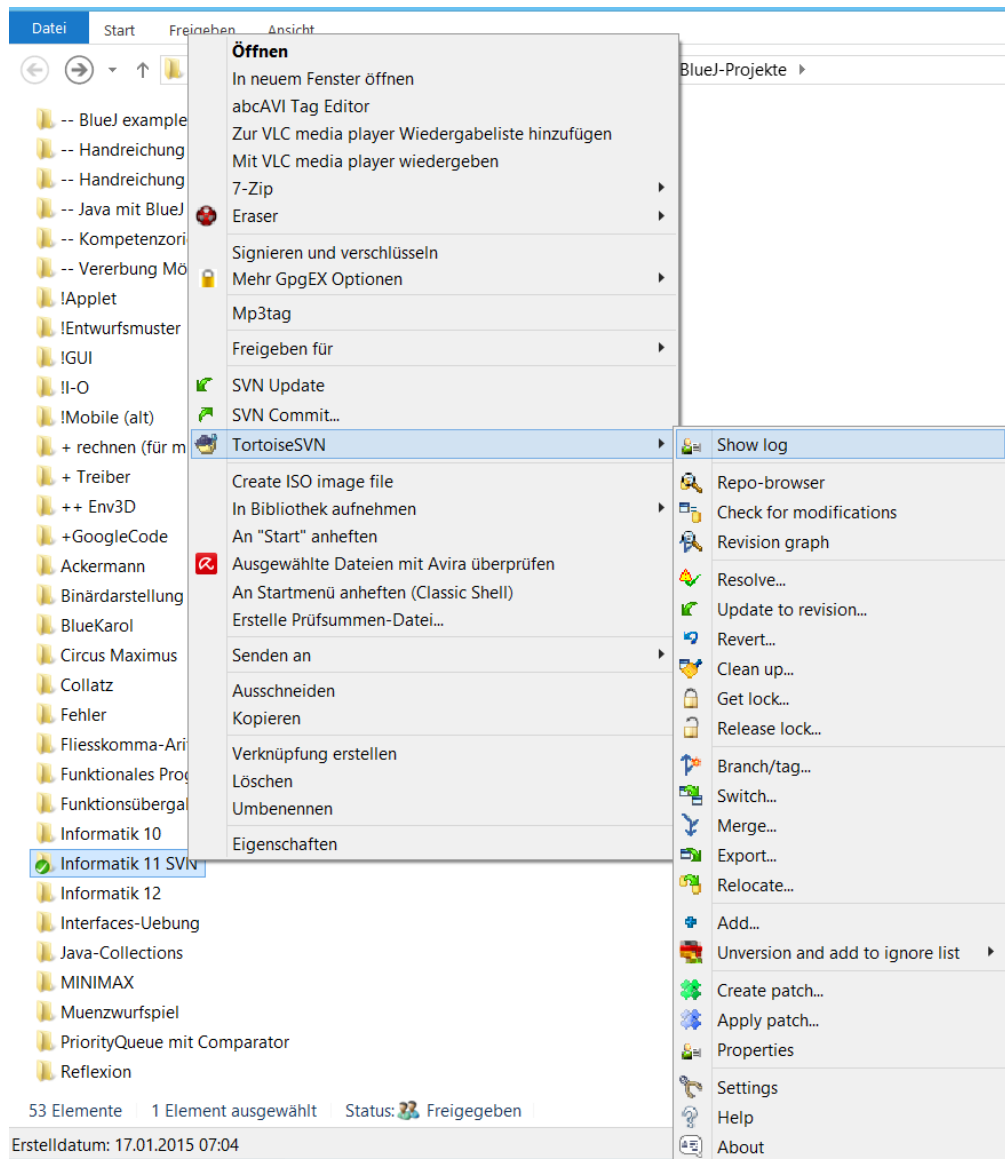


Nach Belieben markiert man einige der Verzeichnisse darin mit TortoiseSVN als „add to ignore list“, so dass diese Verzeichnisse eben nicht hochgeladen werden. Im Bild liegen nur die grünen Verzeichnisse auf dem Server und werden damit synchronisiert.

Man aktualisiert die BlueJ-Projekte jetzt nicht mehr von BlueJ aus (und kann das auch gar nicht), sondern über den eigenen Client, der sich in das Windows-Kontextmenü integriert:



Das Menü bietet noch weitere Möglichkeiten von Subversion, die für das Arbeiten in der Schule und mit BlueJ aber alle nicht nötig sind:



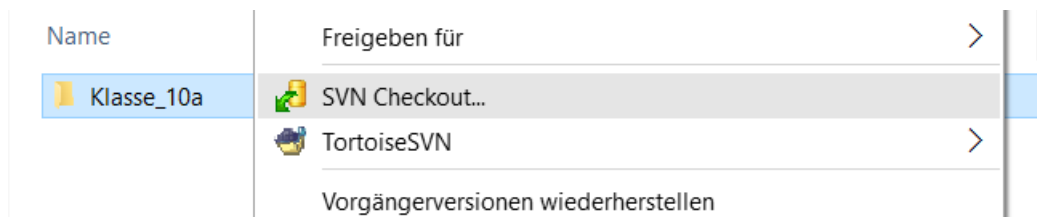
Wenn die Lehrkraft mit TortoiseSVN arbeitet, kann sie *nicht* wie die Schüler mit BlueJ arbeiten. Bei den Schülern hat jedes BlueJ-Projekt seine eigene .svn (ein verstecktes Verzeichnis mit SVN-Daten), bei der Lehrkraft gibt es *ein* .svn für das Verzeichnis mit den BlueJ-Projekten.

Will die Lehrkraft sich ein BlueJ-Projekt aus Schülersicht betrachten, muss sie das Projekt erst *aus BlueJ heraus* auschecken und kann das daraufhin neu gespeicherte Projekt dann auch mit BlueJ committen. Das ist dann aber *nicht* das Projekt im eigentlichen zentralen BlueJ-Ordner des Lehrers.

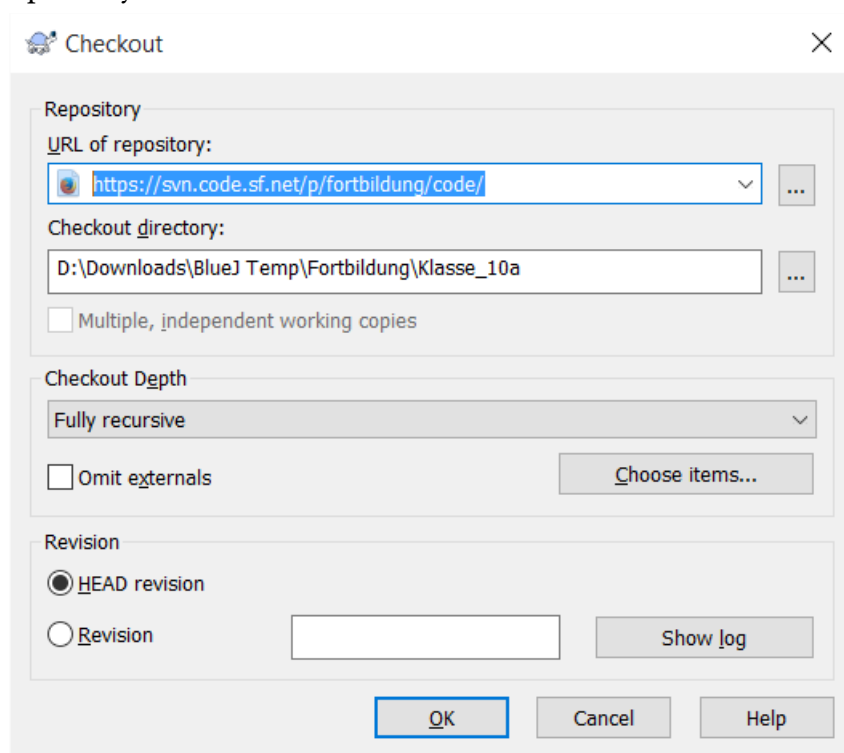
## Tortoise SVN – Eine lokale Kopie des Repository

Ganz am Anfang steht das Anlegen einer lokalen Kopie des Repository:

- Optional: Legen Sie ein leeres Repository auf einem Server an.
- Legen Sie ein Verzeichnis an für alle Projekte, die Sie zum Beispiel mit einer Schulklasse nutzen wollen. etwa „Klasse\_10a“.
- Wählen Sie für das Verzeichnis im Kontextmenü „SVN Checkout...“



- Geben Sie danach die URL des eben angelegten oder bereits existierenden Repository an:



- Danach werden alle Verzeichnisse und Dateien im Repository – also in der Regel alle BlueJ-Projekte dort – in das gewählte Verzeichnis übertragen.
- Ab jetzt kann das gewählte Verzeichnis mit dem Repository synchronisiert werden. Das geschieht über „SVN Update“ beziehungsweise „SVN Commit...“ im Kontextmenü.

# Teil 6 – Erfahrungen

## Rückmeldung aus der Praxis

- Kommt bei Schülern sehr gut an.
- Die Aktualisierung bestehender BlueJ-Projekte ist leichter als sie etwa bei Moodle aktuell zu halten
- Am Anfang unbedingt im Computerraum gemeinsam machen; zu Hause kommt kaum einer allein zurecht
- Gelegentlich gibt es ein unerklärtes *write lock* auf lokalem Projekt. Dann einfach neu auschecken und den bisher verwendeten BlueJ-Projekt-Ordner löschen.
- Interessierte Schüler gehen ins Repository und blättern dort. („Wer hat die Datei gelöscht?“)
- Für Projektarbeit enorm hilfreich. Der Austausch und die Koordination gehen sehr viel schneller und fehlerfreier als beim manuellen Dateienaustausch zwischen Teammitgliedern. Im Computerraum verständigt man sich durch kurzes Zwischenrufe „Habt ihr schon Update gemacht“ usw.
- Oft arbeiten Schüler mit dem Account eines anderen Schülers, weil die Ab-/Anmeldung ihnen zu umständlich erscheint oder nicht alle ihre Zugangsdaten dabei haben.
- Es ist vielleicht hilfreich, wenn Schülerinnen und Schüler erst einige Zeit lang nur BlueJ-Projekte auschecken (Aufgaben, Musterlösungen), und erst, wenn sie damit vertraut sind, Commits durchführen.
- Schülerinnen und Schüler denken selten daran, einem Commit einen Kommentar hinzuzufügen.

## Typische Probleme

- Hauptproblem: BlueJ führt keine Synchronisation durch oder erhält keine Verbindung zum Server. Grund: Je nach Version benutzt BlueJ eine SVNkit-Library, die Bugs enthält. Mit der BlueJ-Version 3.17 (Februar 2016) sollte alles funktionieren. (Externe SVN-Tools neuerer Generation könnten allerdings Probleme haben.)
- Eingabe der korrekten Daten in BlueJ (Benutzername, Passwort) für manche schwierig, das Passwort wird vergessen oder nicht richtig aufgeschrieben –

Schüler müssen sich ihre Zugangsdaten aufschreiben und jedesmal mitbringen.

- Tatsächlich werden Schüler immer die Zugangsdaten ihrer Mitschüler nutzen. Das hat den Nachteil, dass man eventuelle Zugangsprobleme erst spät feststellt.
- Manche Schüler hatten Schwierigkeiten, zu Hause die Teamarbeitseinstellungen in BlueJ einzuschalten, weil sie den Menüpunkt nicht gefunden haben, eventuell ist das Interface bei BlueJ auch auf englische Sprache eingestellt.
- Die Arbeit mit dem in BlueJ integrierten und einem externen SVN-Client muss getrennt bleiben. Wenn der externe Client eine .svn-Datei in einem BlueJ-Projekt entdeckt, führt das zu Problemen.
- Gelegentlich kommt es aus irgendwelchen Gründen zu SVN-Fehlermeldungen des BlueJ-Clients (wie einem *write lock*, einem von BlueJ nicht genutzten und ansprechbaren SVN-Feature). Dann einfach das Projekt ganz neu auschecken, das alte lokale löschen und eventuelle Codeergänzungen noch einmal durchführen.
- Enthält ein Verzeichnis Bilddateien, fügt Windows oft eine versteckte Datei „thumbs.db“ hinzu, die dann mit synchronisiert wird.

#### **Zur Erinnerung und als Übersicht:**

**Checkout** – macht man meist einmalig und lädt dabei alle Inhalte aus einem Repository in ein lokales Verzeichnis.

**Update** – führt man regelmäßig und insbesondere vor Beginn eigenen Arbeitens durch. Dabei werden lokale Fassungen eventuell durch aktuellere aus dem Repository ersetzt. Kann zu Konflikten führen.

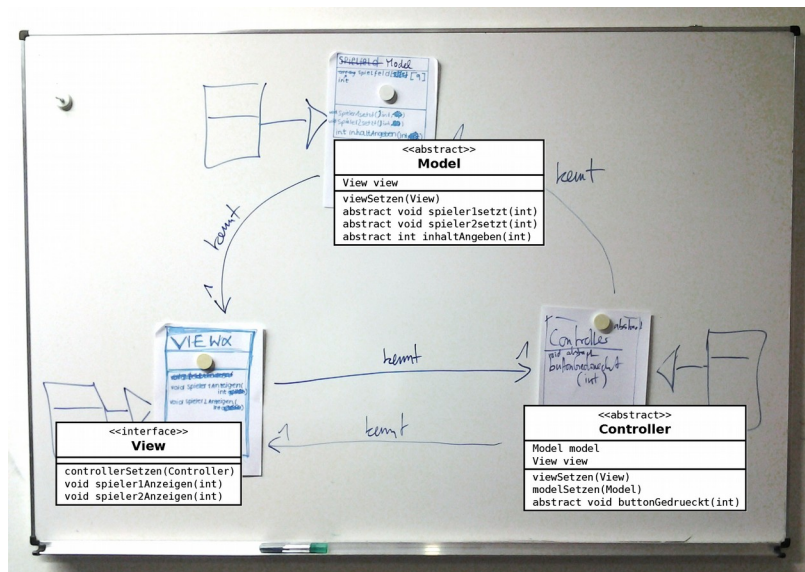
**Commit** – führt man unmittelbar nach dem eigenen Arbeiten durch. Wenn inzwischen neuere Fassungen im Repository liegen, wird man stattdessen zu einem Update und anschließender Konfliktlösung gezwungen. Danach kann man den Commit noch einmal versuchen.



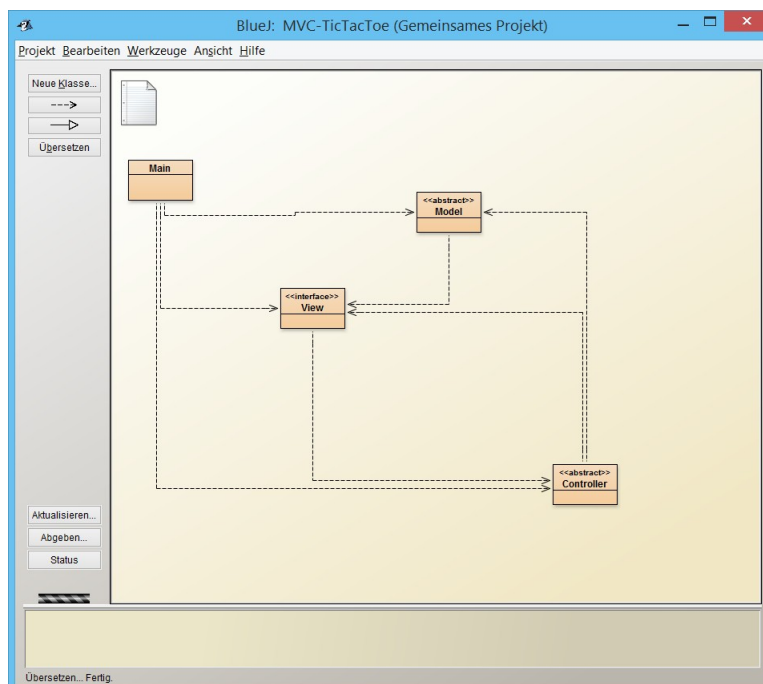
# Teil 7 – Beispiele

## Einfaches Tic-Tac-Toe mit Model-View-Controller

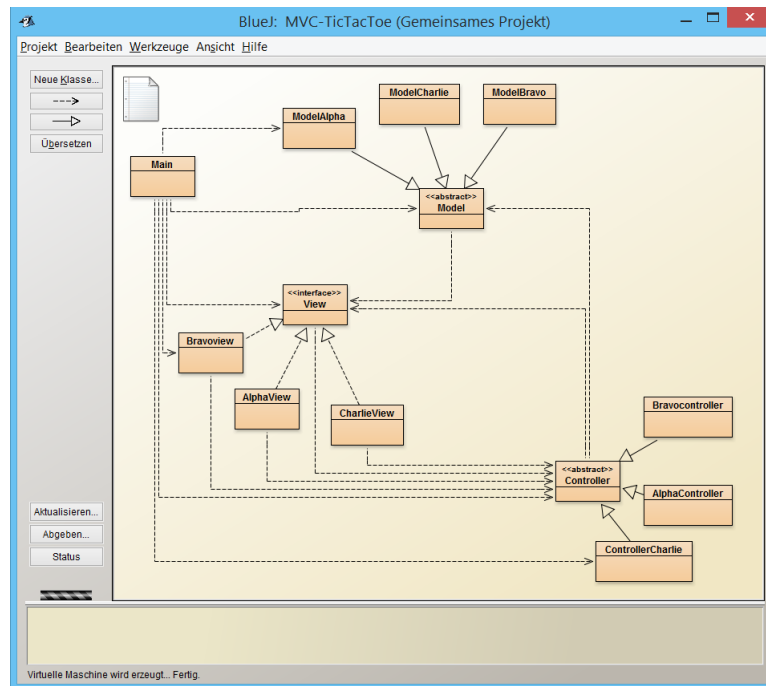
Als Fingerübung vor der eigentlichen Projektarbeit sollte ein Tic-Tac-Toe-Spiel programmiert werden. Dazu wurde in einer Stunde ein Klassendiagramm erstellt, angefangen vom Model (was muss gespeichert werden?) über den View (was muss der erfahren, um den Spielstand darstellen zu können?) bis hin zum Controller.



Dann wurden drei Teams eingeteilt (Alpha, Bravo, Charlie); in jedem Team waren jeweils zwei bis drei Schülerinnen und Schüler für die konkrete Umsetzung der drei abstrakten Klassen zuständig. So konnten über zwanzig Schüler weitgehend unabhängig von den anderen arbeiten.



Am Schluss sah das Projekt so aus, alle drei Model-, View- und Controller-Implementierungen konnten untereinander ausgetauscht werden:



## Größeres Programmierprojekt

Danach folgten acht Wochen Programmierprojekt in drei Gruppen zu jeweils 7-8 Mitgliedern. Jede Gruppe arbeitete in einem eigenen BlueJ-Projekt.



