JADE Version Control

Version 2018

Contents

JADE Version Control

Introduction

JADE provides a variety of tools to assist with version control of JADE schemas.

- Deltas allow for methods to be checked in and out, enabling developers to lock methods while they are working on them.
- Patch versioning allows the setting of a patch version number that records all changes made to schema entities until a new patch version number is set.
- The JADE Git client allows for the use of the Git version control system to maintain a history of changes.

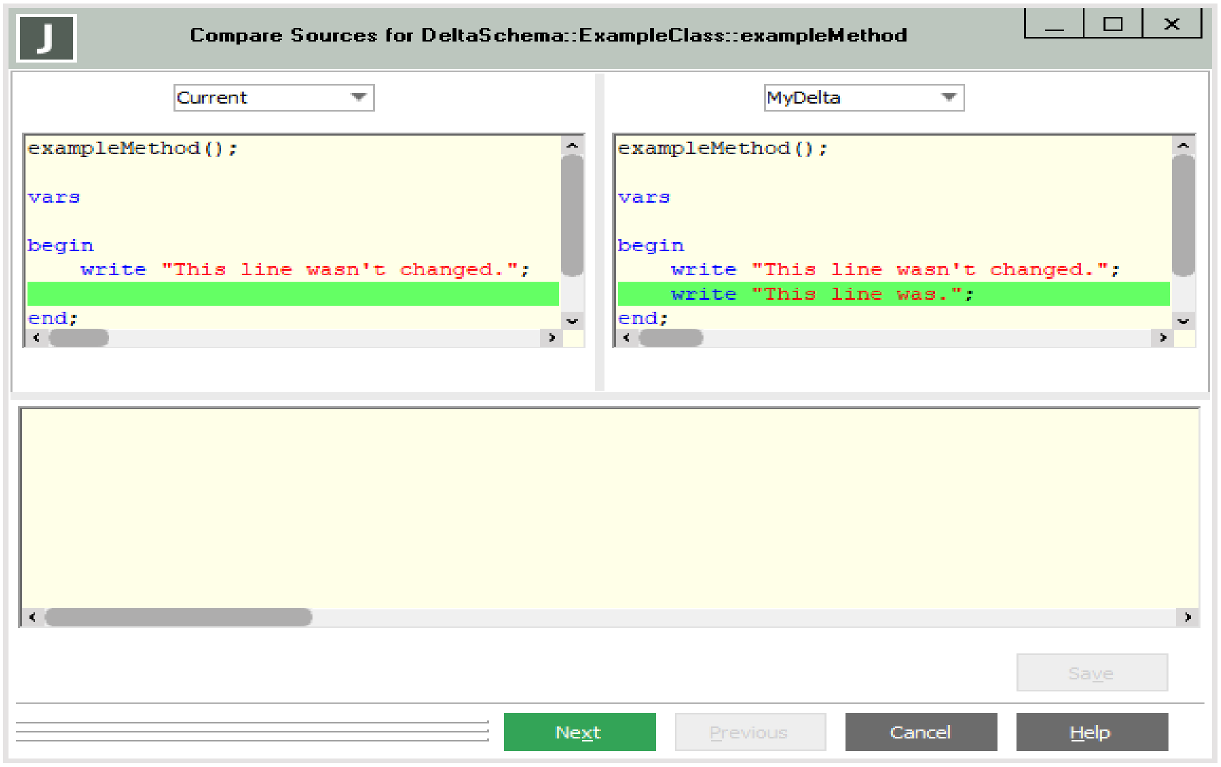
Deltas

Deltas are a tool used for managing changes to a JADE method so that when a method is checked out, any modifications can be reviewed. After review, the method can be accepted (with **Check In**) or reverted (with **Undo Checkout**).

A delta is essentially a list of changes to a method, and as such, is typically used to review changes during a development process; for example, ensuring that the scope of a change is limited to the intended scope so that no extraneous changes are unintentionally made during a method change.

The Compare Sources window provides this functionality, serving as an inbuilt diff tool and highlighting the parts of the method that have changed since checking out the method.

The following is an example of the Compare Sources window.


Compare Sources for DeltaSchema::ExampleClass::exampleMethod

Current

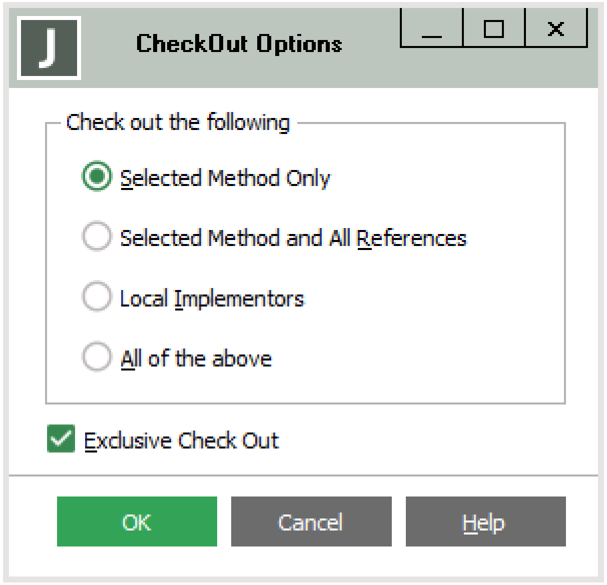
```
exampleMethod();

vars

begin
    write "This line wasn't changed.";

end;
```

MyDelta

```
exampleMethod();

vars

begin
    write "This line wasn't changed.";
    write "This line was.";
end;
```

Note A diff tool shows the differences between two text files. This becomes especially useful when there is a large amount of text and only a very small amount of it has been changed, as well as for verifying that no code has been changed unintentionally.

Deltas in JADE also provide the ability to lock methods while they are checked out. Checking the **Exclusive Check Out** check box when checking out a method prevents any other user from modifying the method until the method is checked back in or reverted.


CheckOut Options

Check out the following

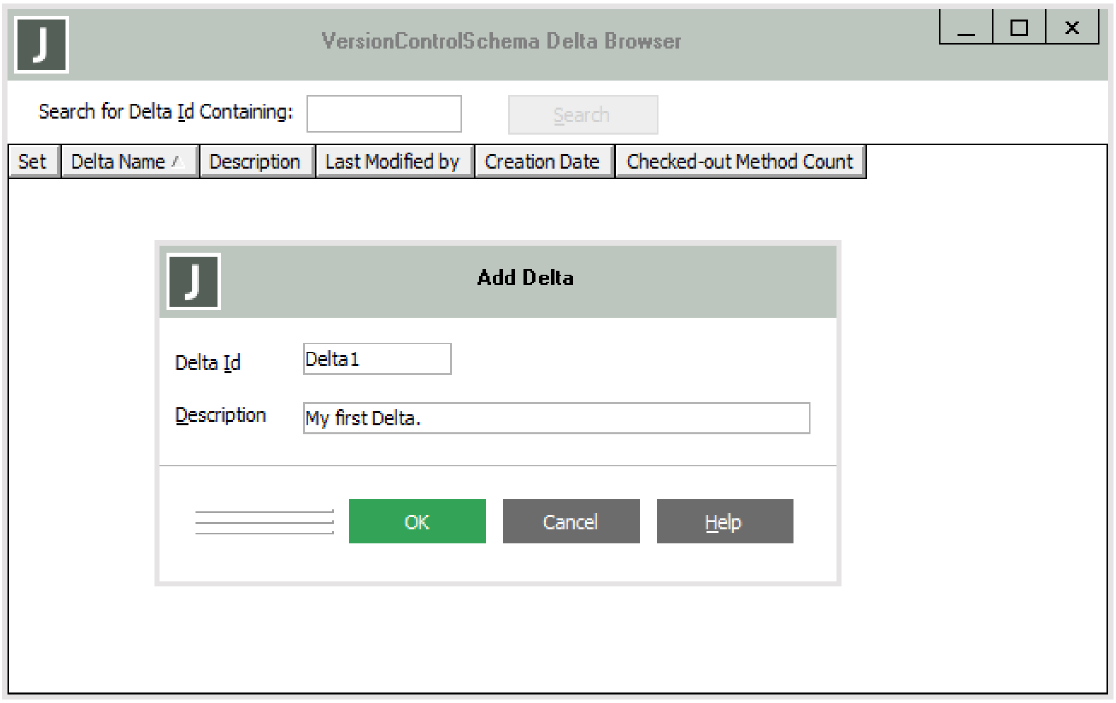
[✓ Selected Method Only]
Selected Method and All References
Local Implementors
All of the above

✓ Exclusive Check Out

Exercise 1 – Creating a Delta

In this exercise, you will create a delta and check out a method into it.

1. Create a new schema called **VersionControlSchema**.
2. Add a class called **ExampleClass** to the **VersionControlSchema**.
3. Add a method called **exampleMethod** to the **ExampleClass**. For now, you don't need to code anything in it.
4. Open the Delta Browser by selecting the **Deltas** command from the Browse menu or pressing CTRL+D.
5. Add a delta by selecting the **Add** command from the Delta menu or right-clicking in the Delta Browser and selecting **Add**.



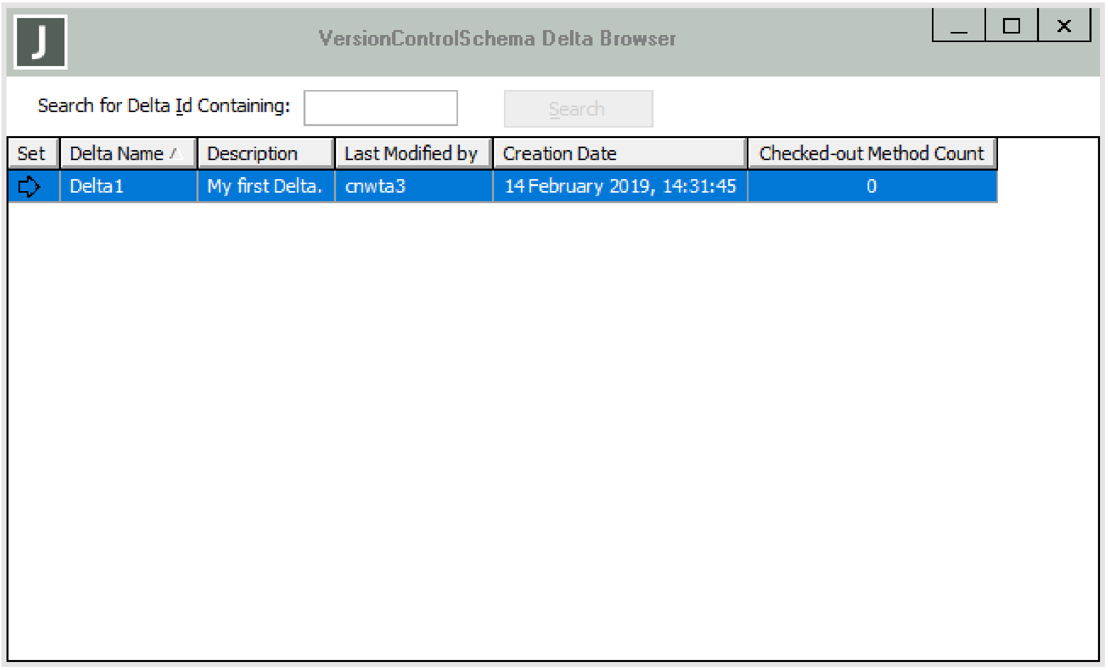
Delta Id: Delta1

Description: My first Delta.

Enter **Delta1** in the **Delta Id** text box and **My first Delta** in the **Description** text box.

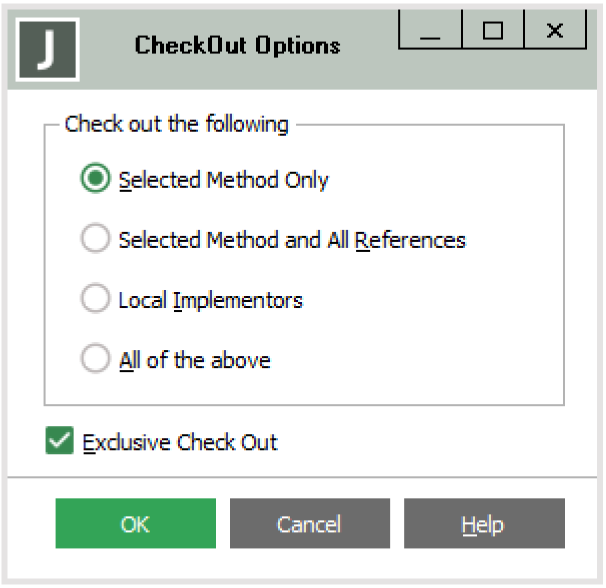
6. The delta is displayed in the list of deltas. Right-click on it and the select **Set**.

An arrow is then displayed in the **Set** column at the left of the delta name.


VersionControlSchema Delta Browser
Search for Delta Id Containing:
Search

Set	Delta Name	Description	Last Modified by	Creation Date	Checked-out Method Count
⇨	Delta1	My first Delta.	cnwta3	14 February 2019, 14:31:45	0

7. Navigate to the **exampleMethod**, right-click on it, and then select **Check Out** from the Delta menu.



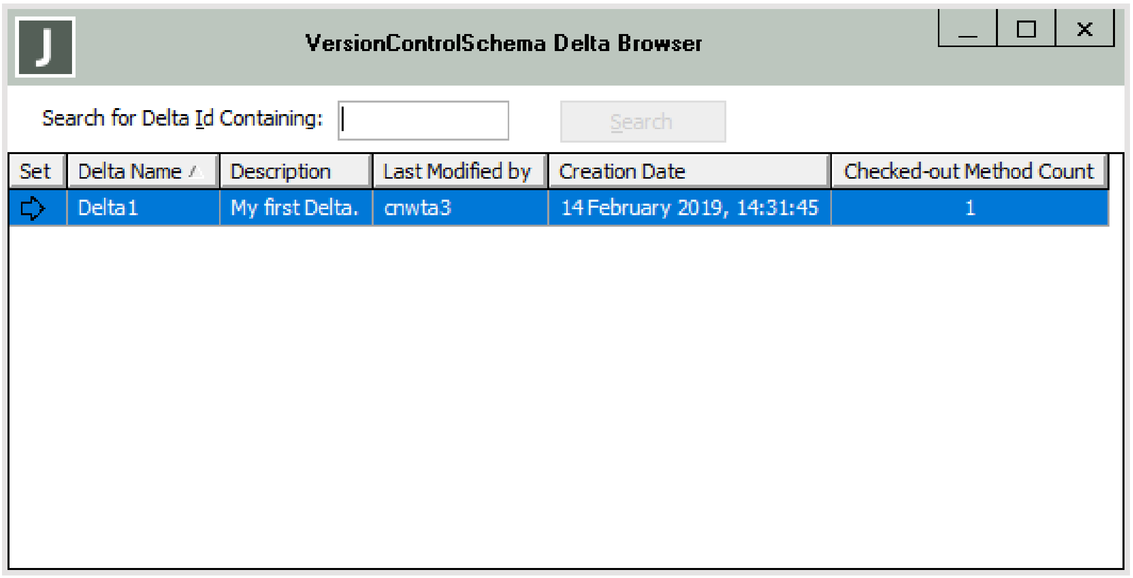
- [✓ Selected Method Only]
- Selected Method and All References
- Local Implementors
- All of the above

✓ Exclusive Check Out

8. Select the **Selected Method Only** option button and check the **Exclusive Check Out** check box, and then press **OK**.

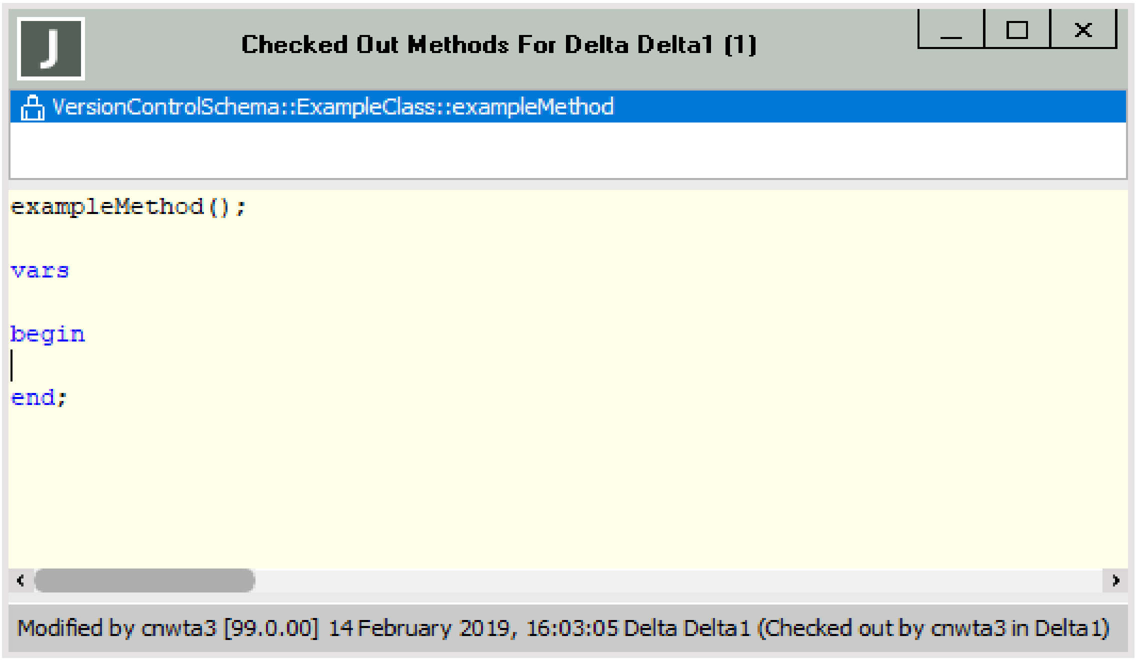
Note *Exclusive check out* means that the method will be locked, and other users cannot modify it until you check it back in or you undo the checkout.

9. Press Ctrl+D to reopen the Delta Browser. You will see that the checked-out method count is now set to **1**.


VersionControlSchema Delta Browser
Search for Delta Id Containing:
Search

Set	Delta Name	Description	Last Modified by	Creation Date	Checked-out Method Count
➪	Delta1	My first Delta.	cnwta3	14 February 2019, 14:31:45	1

10. Right-click the delta and then select **View Methods**. You will see a methods list containing all checked out methods (currently there is just the one).


Checked Out Methods For Delta Delta1 (1)
VersionControlSchema::ExampleClass::exampleMethod

```
exampleMethod();

vars

begin

end;
```

Modified by cnwta3 [99.0.00] 14 February 2019, 16:03:05 Delta Delta1 (Checked out by cnwta3 in Delta1)

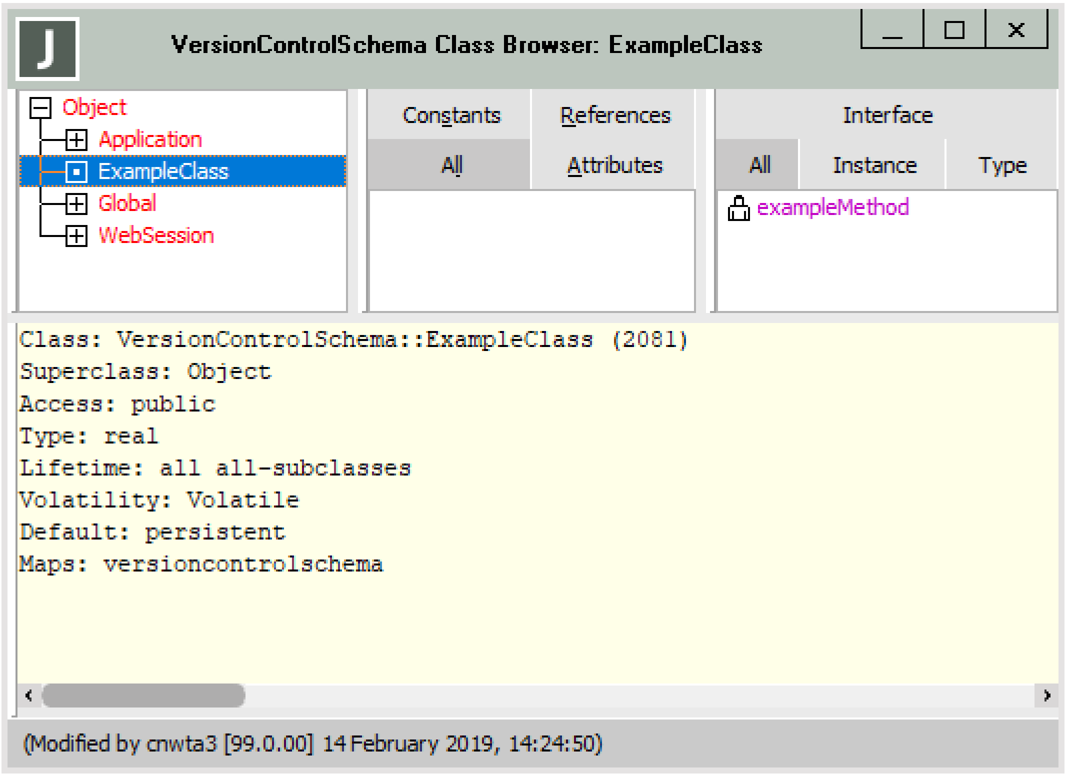
Exercise 2 – Checking in from a Delta

Note This exercise will be easiest if you open your JADE database in multiuser mode; that is, with a database server and two standard (fat) clients.

In this exercise, you will view the checked out method from another user's perspective, modify it, and then check the method back in from the delta.

1. Log on to your JADE database with a different name than that with which you created the delta. (If you are running in single user mode, close and reopen the database or start a standard client if you have a database server and a standard client running.)

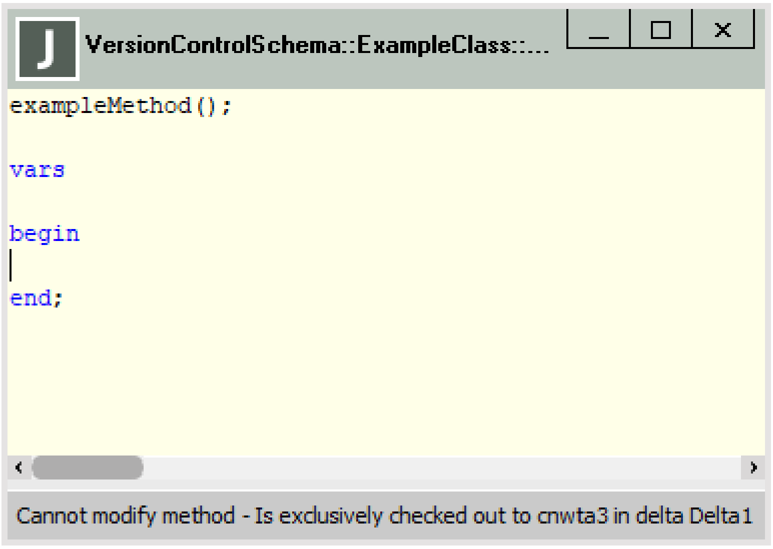
2. Navigate to the **exampleMethod** method. You will see that it is a violet color instead of the usual black.



```
Class: VersionControlSchema::ExampleClass (2081)
Superclass: Object
Access: public
Type: real
Lifetime: all all-subclasses
Volatility: Volatile
Default: persistent
Maps: versioncontrolschema
```

(Modified by cnwta3 [99.0.00] 14 February 2019, 14:24:50)

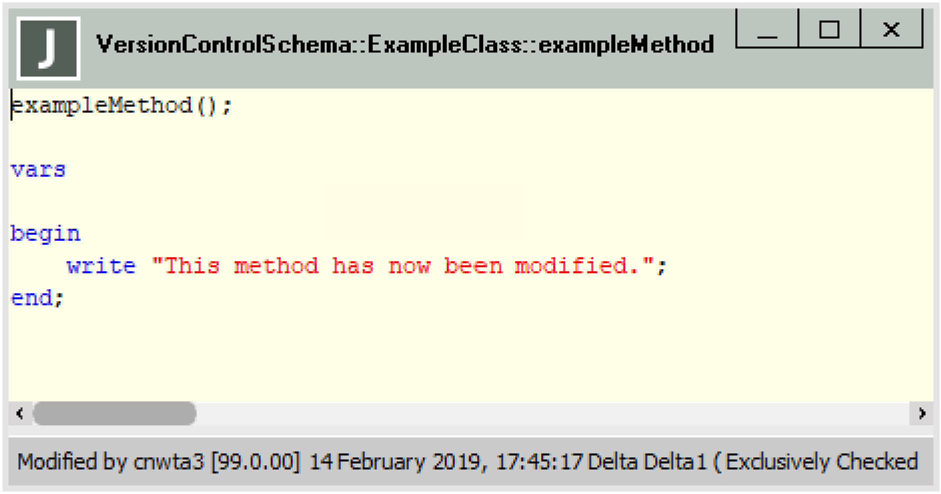
3. Attempt to modify the method. You will see that you cannot enter any text into the method and that the status line reads **Cannot modify method**.



```
exampleMethod();

vars

begin

end;
```

Cannot modify method - Is exclusively checked out to cnwta3 in delta Delta1

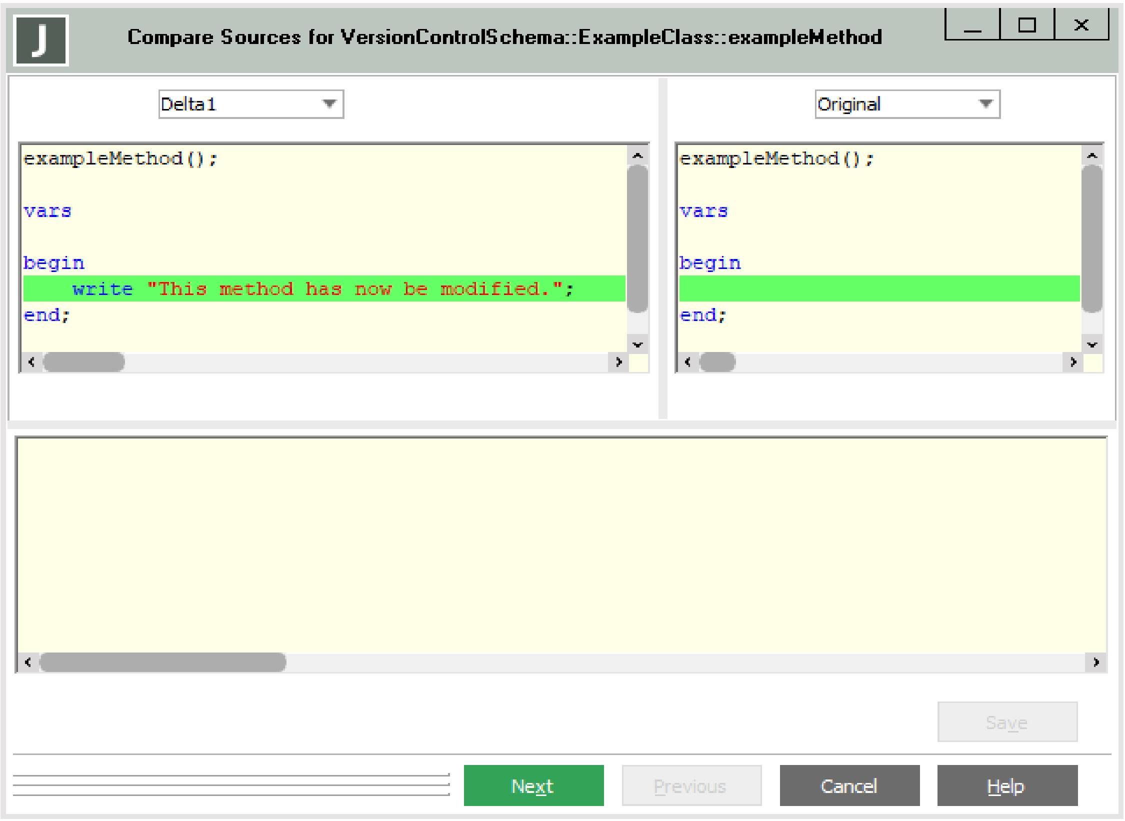
4. Open the database with your original user name and then modify the **exampleMethod** as follows.



```
exampleMethod();

vars

begin
    write "This method has now been modified.";
end;
```

Modified by cnwta3 [99.0.00] 14 February 2019, 17:45:17 Delta Delta1 (Exclusively Checked

5. Right-click the **exampleMethod** in the Class Browser and then select the **Compare** command from the Delta menu.


Compare Sources for VersionControlSchema::ExampleClass::exampleMethod
Delta1
Original
Save
Next
Previous
Cancel
Help

```
exampleMethod();

vars

begin
    write "This method has now be modified.";
end;
```

```
exampleMethod();

vars

begin

end;
```

The Compare Sources window shows the change or changes made to the method.

Tip This tool is useful when making changes to a large method, to ensure that you have changed only what you intended to.

6. Close the window, right-click the **exampleMethod** in the Class Browser, and then select the **Check in** command from the Delta menu. (Click **Yes** in the message box prompting you to confirm that you want to check in the method.)

7. Now that the method is checked back in, it is no longer locked. Open it in your other JADE client node to verify this.

Patch Versioning

Patch versioning allows for the setting of patch numbers for a JADE database and assigning a group of schema changes to that patch.

Bundling sets of changes into patches is a common software development technique, allowing for precision in the scope of changes to be deployed.

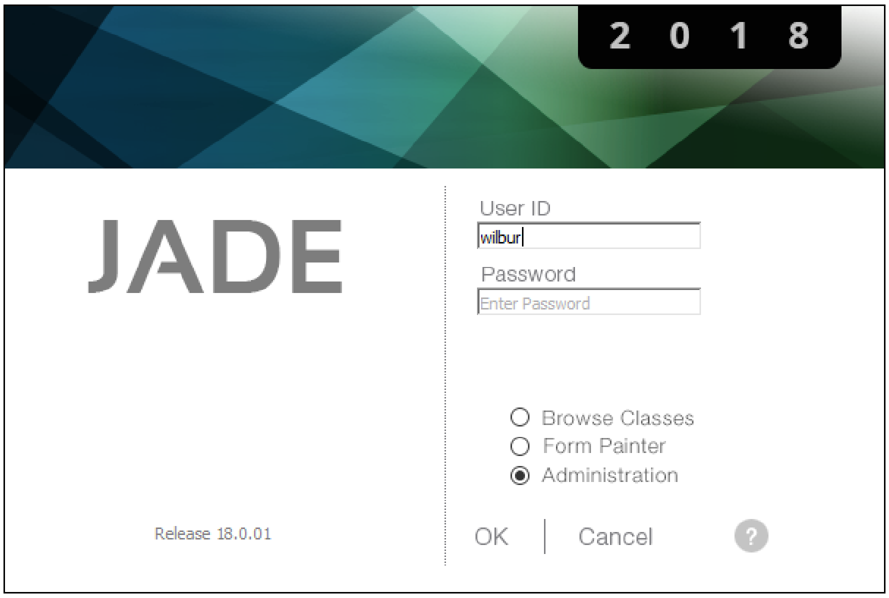
Note Patch versioning is completely distinct from deltas, and you can use either without the other. That said, a version control strategy often incorporates both.

One advantage of employing a patch versioning strategy is that it will cause a history of changes to be generated over the development of a solution. This history can be viewed from the JADE development environment using the Compare Sources window, allowing you to identify what changed, who changed it, and when it changed. This can be useful if a method's behavior changes unexpectedly.

Exercise 3 – Creating a Patch

In this exercise, you will create a patch for your database and add a number of changes to it.

1. Reopen your JADE database, but select the **Administration** option instead of the default **Browse Classes** option on the logon form.

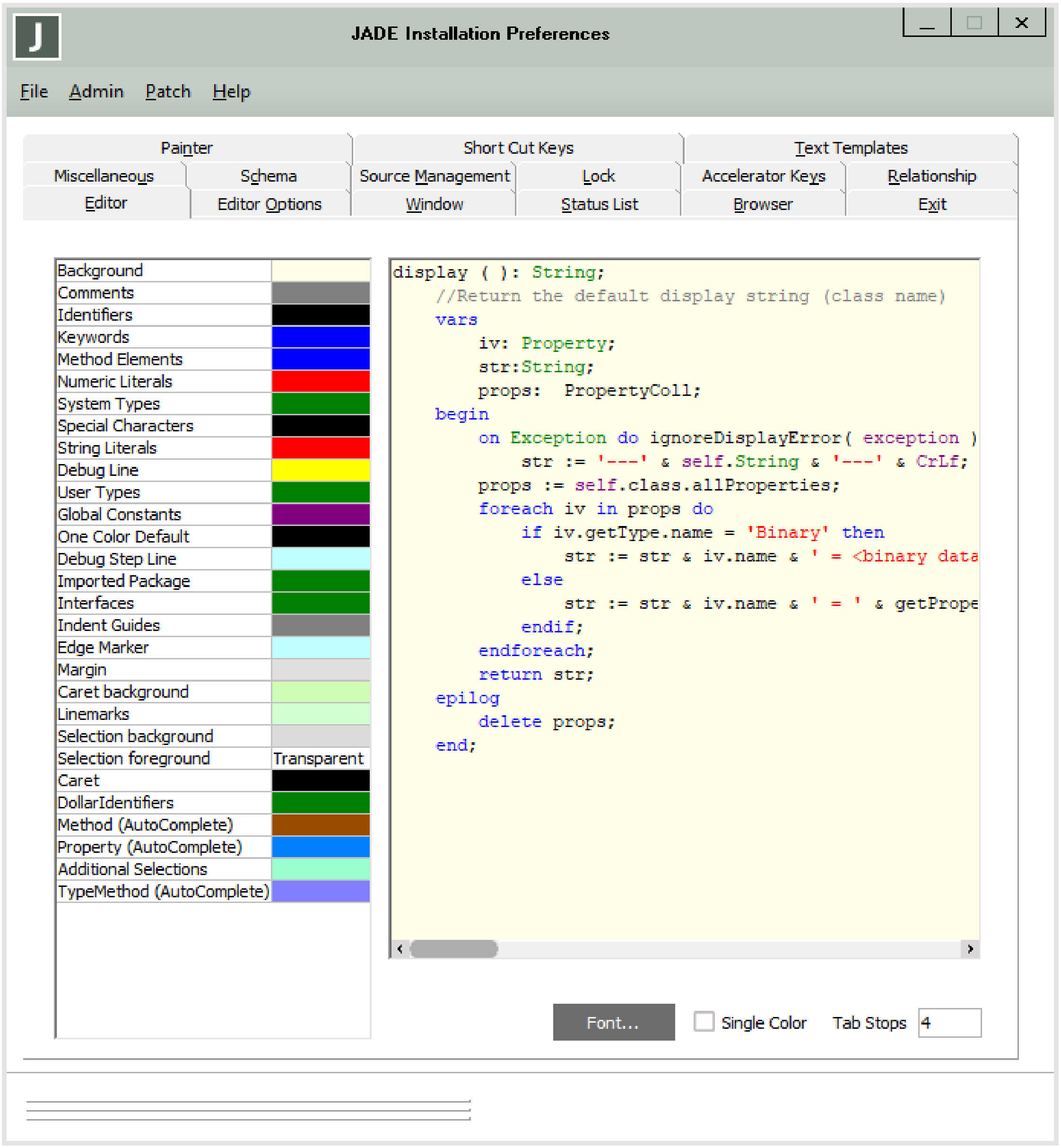


User ID: wilbur

Password: Enter Password

Browse Classes
Form Painter
[✓ Administration]

2. The Jade Installation Preferences dialog is then displayed. This is similar to the User Preferences dialog except that it has a **Painter** sheet and menus - specifically the *Patch* menu.



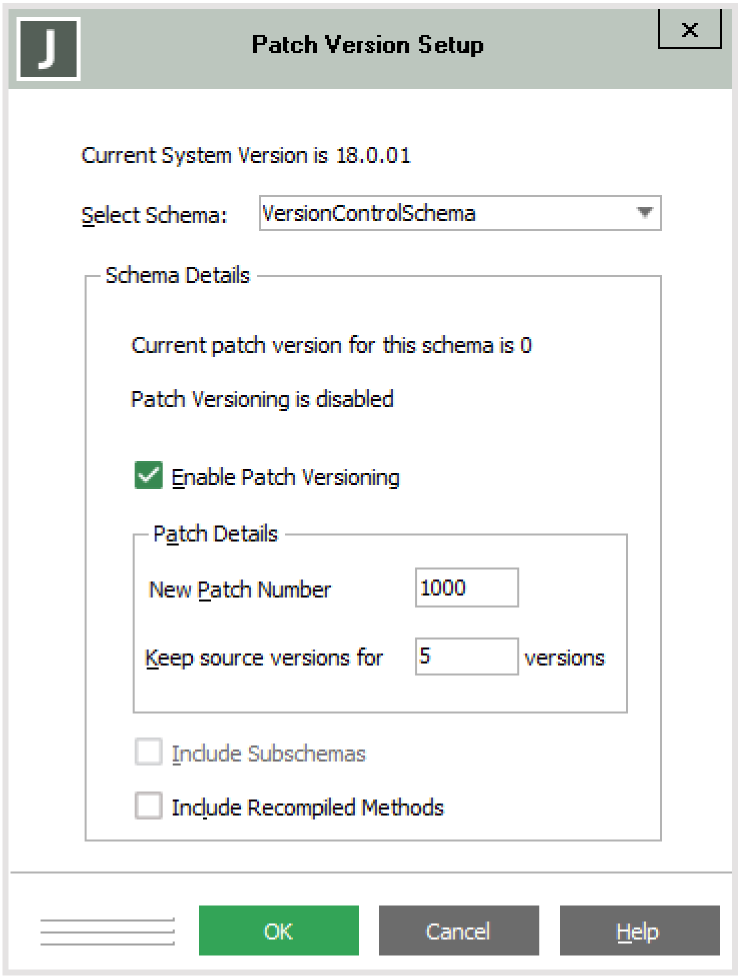
Background	⬜
Comments	▨
Identifiers	⬛
Keywords	🟦
Method Elements	🟦
Numeric Literals	🟥
System Types	🟩
Special Characters	⬛
String Literals	🟥
Debug Line	🟨
User Types	🟩
Global Constants	🟪
One Color Default	⬛
Debug Step Line	⬜
Imported Package	🟩
Interfaces	🟩
Indent Guides	▨
Edge Marker	⬜
Margin	⬜
Caret background	⬜
Linemarks	⬜
Selection background	⬜
Selection foreground	Transparent
Caret	⬛
DollarIdentifiers	🟩
Method (AutoComplete)	🟫
Property (AutoComplete)	🟦
Additional Selections	⬜
TypeMethod (AutoComplete)	🟪

```
display ( ): String;
    //Return the default display string (class name)
    vars
        iv: Property;
        str:String;
        props:  PropertyColl;
    begin
        on Exception do ignoreDisplayError( exception )
            str := '---' & self.String & '---' & CrLf;
        props := self.class.allProperties;
        foreach iv in props do
            if iv.getType.name = 'Binary' then
                str := str & iv.name & ' = <binary data
            else
                str := str & iv.name & ' = ' & getPrope
            endif;
        endforeach;
        return str;
    epilog
        delete props;
    end;
```

Single Color [ ]   Tab Stops 4

3. Select the **Set Patch Number** command from the Patch menu

Fill out the Patch Version Setup dialog that is then displayed as follows.



Current System Version is 18.0.01

Select Schema: VersionControlSchema

Schema Details

Current patch version for this schema is 0

Patch Versioning is disabled

[✓] Enable Patch Versioning

Patch Details

New Patch Number: 1000

Keep source versions for 5 versions

[ ] Include Subschemas

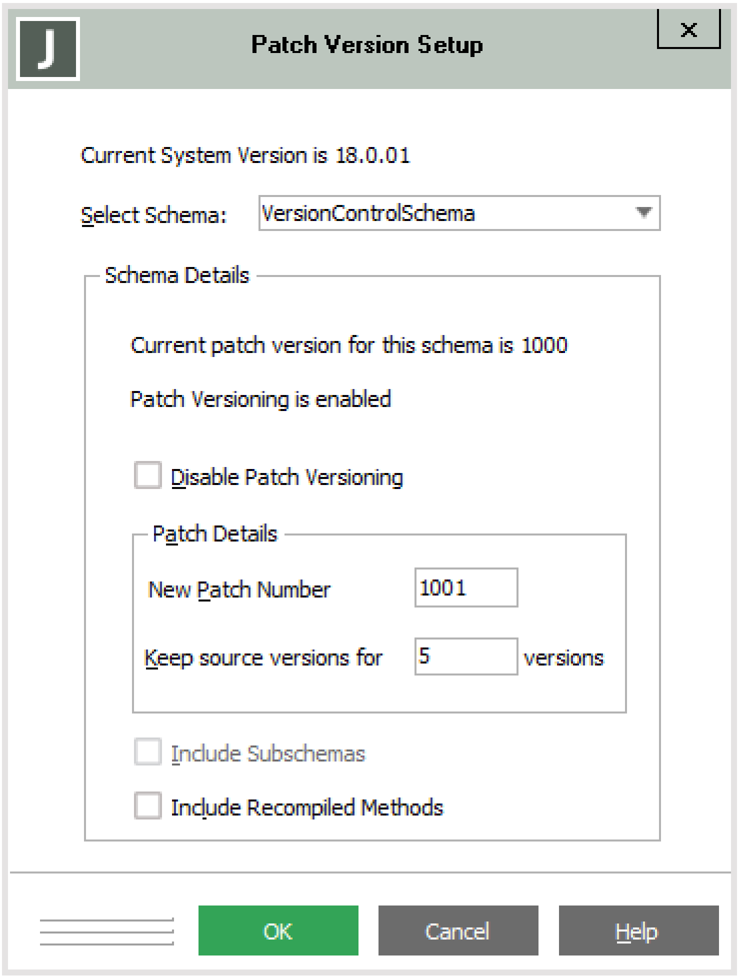
[ ] Include Recompiled Methods

4. Click **OK**, then reopen the database in Browse Classes mode.
5. Navigate to the exampleMethod and then modify it as follows.

```
exampleMethod();

vars

begin
    write "This method will have a patch history.";
end;
```

6. Save the **exampleMethod**, then close the database and reopen it in **Administration** mode (as you did in step 1 of this exercise).

In the Patch Version Setup dialog, change the new patch number to **1001**.


Patch Version Setup

Current System Version is 18.0.01

Select Schema: VersionControlSchema

Schema Details

Current patch version for this schema is 1000

Patch Versioning is enabled

☐ Disable Patch Versioning

Patch Details

New Patch Number: 1001

Keep source versions for 5 versions

☐ Include Subschemas

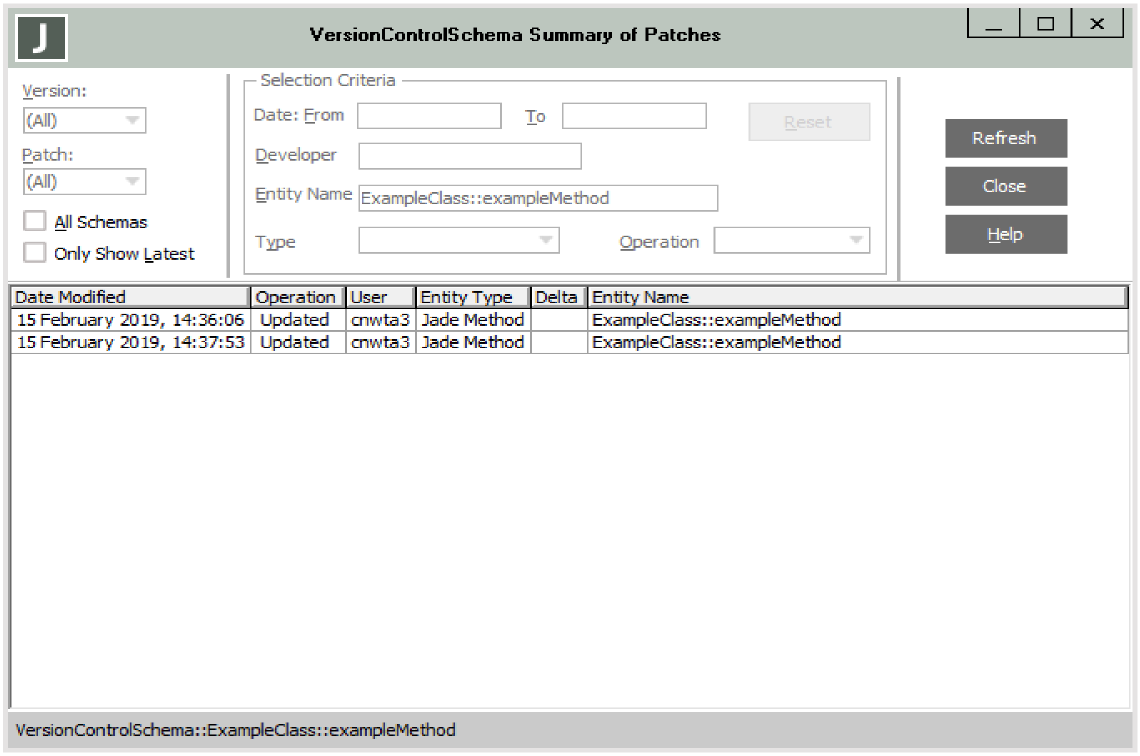
☐ Include Recompiled Methods

7. Click **OK**, then reopen the database in Browse Classes mode.
8. Navigate to the **exampleMethod** and modify it as follows.

```
exampleMethod();

vars

begin
    write "This is the latest patch";
end;
```

9. Right-click the **exampleMethod** then select **Show History**.

The Summary of Patches form is then displayed.



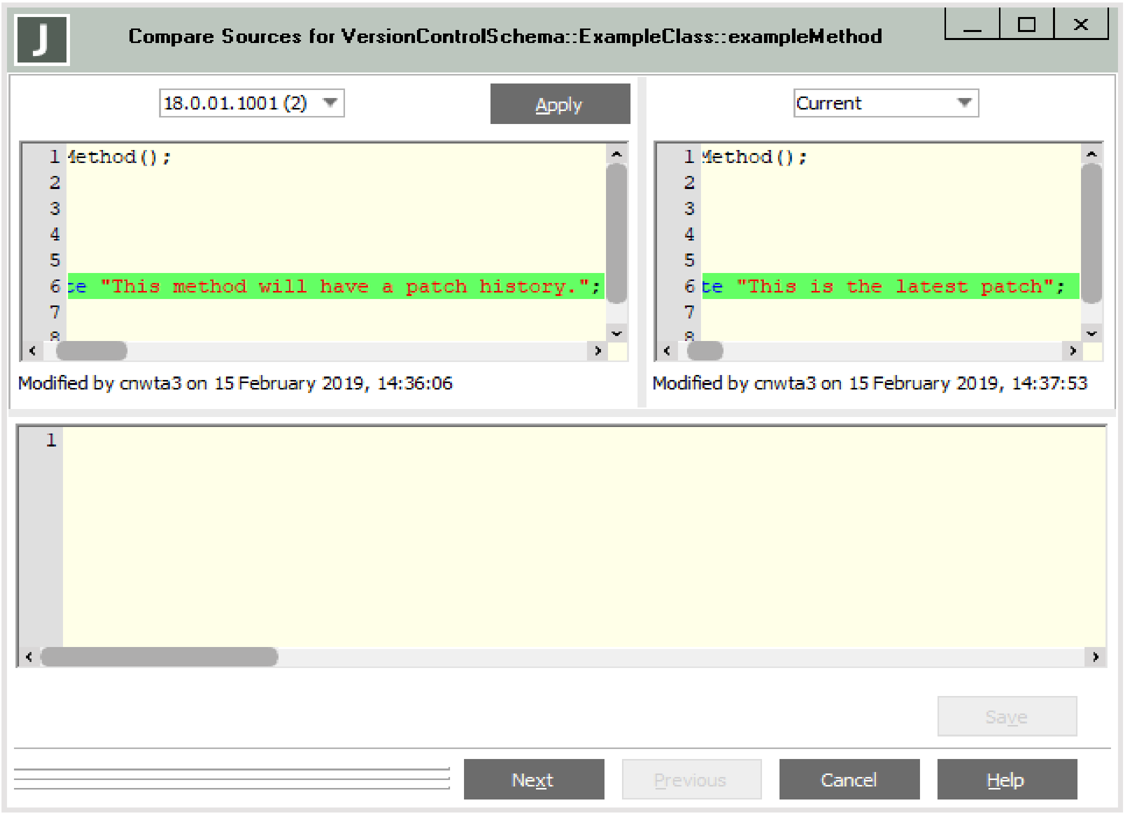
Entity Name: ExampleClass::exampleMethod

Date Modified	Operation	User	Entity Type	Delta	Entity Name
15 February 2019, 14:36:06	Updated	cnwta3	Jade Method		ExampleClass::exampleMethod
15 February 2019, 14:37:53	Updated	cnwta3	Jade Method		ExampleClass::exampleMethod

VersionControlSchema::ExampleClass::exampleMethod

Note This feature is available only while patch versioning is enabled.

10. Double-click the second item in the list, to display the Compare Sources window.


Compare Sources for VersionControlSchema::ExampleClass::exampleMethod
18.0.01.1001 (2)
Apply
Current
Save
Next
Previous
Cancel
Help

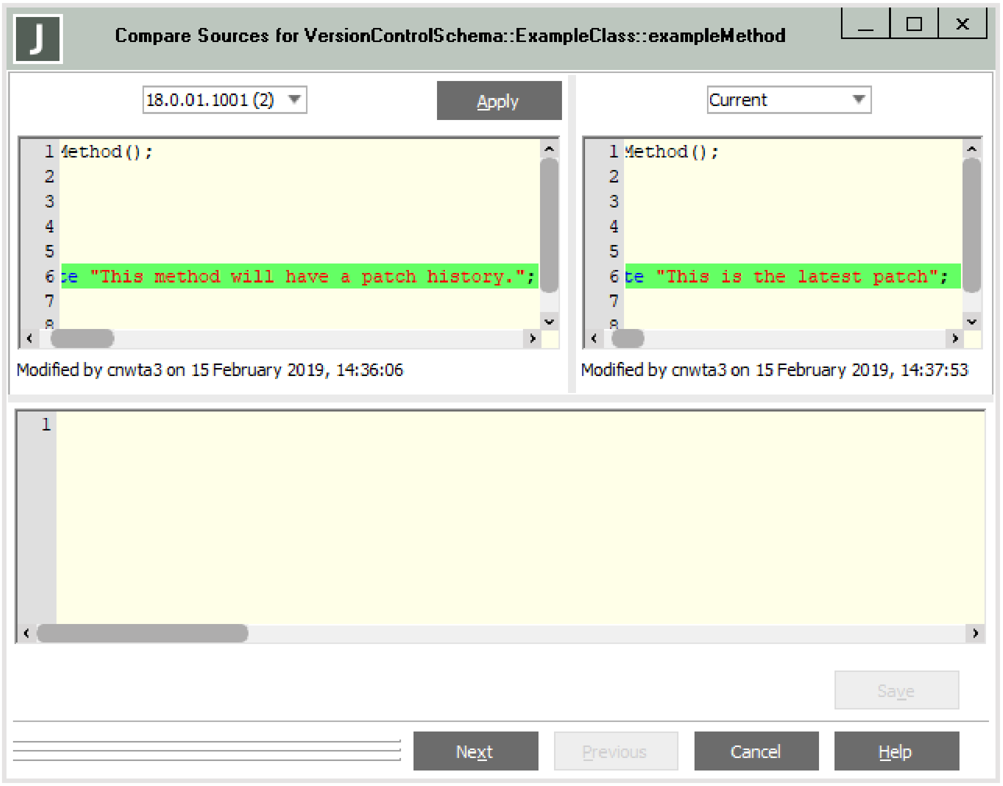
```
lethod();




te "This method will have a patch history.";
```

Modified by cnwta3 on 15 February 2019, 14:36:06

```
Method();




te "This is the latest patch";
```

Modified by cnwta3 on 15 February 2019, 14:37:53

The Compare Sources window shows the differences between the current version of the method and the version as it was when the patch was set, the user who modified the method, and a timestamp of when it was modified.


Compare Sources for VersionControlSchema::ExampleClass::exampleMethod

18.0.01.1001 (2)

```
1 Method();
2
3
4
5
6 te "This method will have a patch history.";
7
```

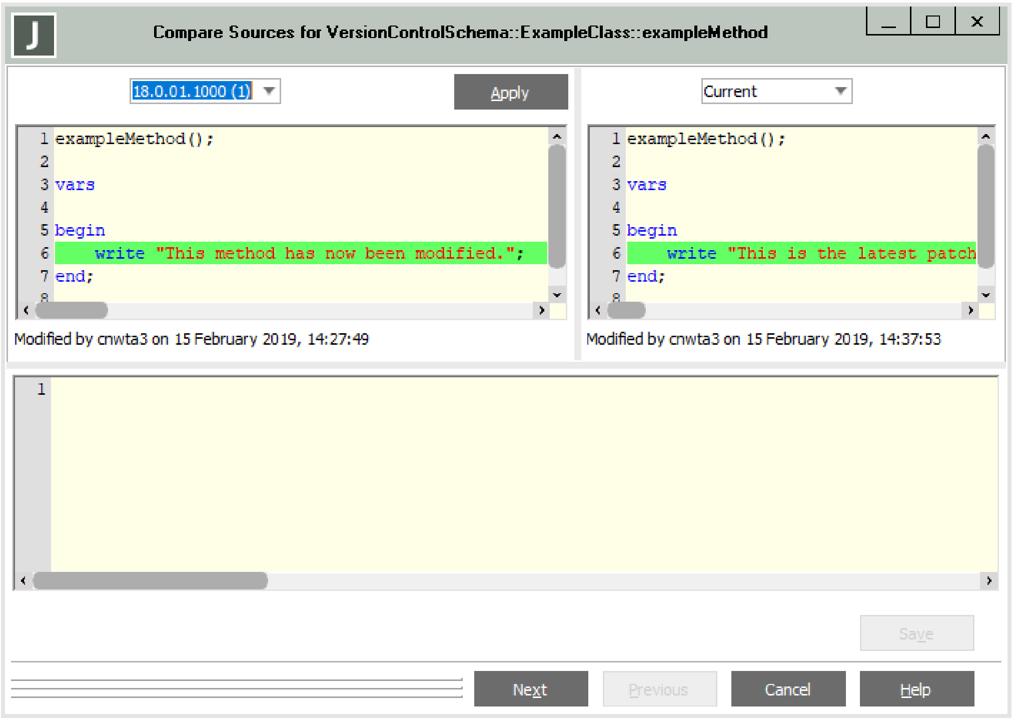
Modified by cnwta3 on 15 February 2019, 14:36:06

Current

```
1 Method();
2
3
4
5
6 te "This is the latest patch";
7
```

Modified by cnwta3 on 15 February 2019, 14:37:53

11. In the combo box for the code source on the left, select the **18.0.01.1000** patch.


Compare Sources for VersionControlSchema::ExampleClass::exampleMethod

18.0.01.1000 (1)

```
exampleMethod();

vars

begin
    write "This method has now been modified.";
end;
```

Modified by cnwta3 on 15 February 2019, 14:27:49

Current

```
exampleMethod();

vars

begin
    write "This is the latest patch
end;
```

Modified by cnwta3 on 15 February 2019, 14:37:53

You will see the differences between the first patch and the current version.

Regular patching allows for the maintenance of a history of patches, documenting how the software has changed over its lifetime. In addition, clicking **Apply** reverts the current source to the source of the selected patch.

Note The **18.0.01** prefix before the **1000** for the patch number is the version number of JADE itself; for example, **18.0.01** is the version number of the first release of JADE 2018.

Git Integration

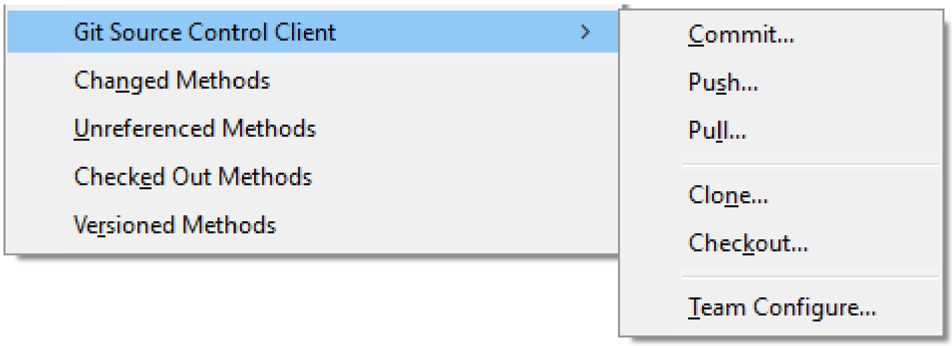
As of the 2018.0.01 release, JADE supports Git integration directly from the JADE development environment.

Git is a version control system in which the entire history of a repository is stored in each developer's checkout. This allows for greater protection of source code, because even if the remote repository is damaged, each user has the complete source along with the complete version history.

Git has several common terms, some of which are specific to the Git technology and some common to version control systems in general. These include the following.

Git Term	Meaning
Branch	A copy of a software's source code at a specific time, along with a history of all changes made up to that point. It is common to have multiple branches within a repository; for example, one of release quality and one with experimental changes or works in progress.
Repository	A collection of source branches. It is typically stored remotely using a service such as GitHub, with each developer also having a local copy on his or her device.
GitHub	A common provider of remote Git repository hosting.
Clone	To make a local copy of a remote repository.
Commit	To save changes to a local branch.
Checkout	To create a new branch. It is initially a copy of a specific branch but diverges as changes are made to each.
Pull	To update a local branch based on a remote branch.
Pull request	A request to merge changes from one branch into another. This typically involves discussion and review before confirmation of the merge and resolution of any conflicts.
Push	To update a remote branch based on a local branch.
Merge	Incorporating changes from one branch to another, while resolving any conflicts.
Conflict	When two branches that are to be merged modify the same code in different ways.

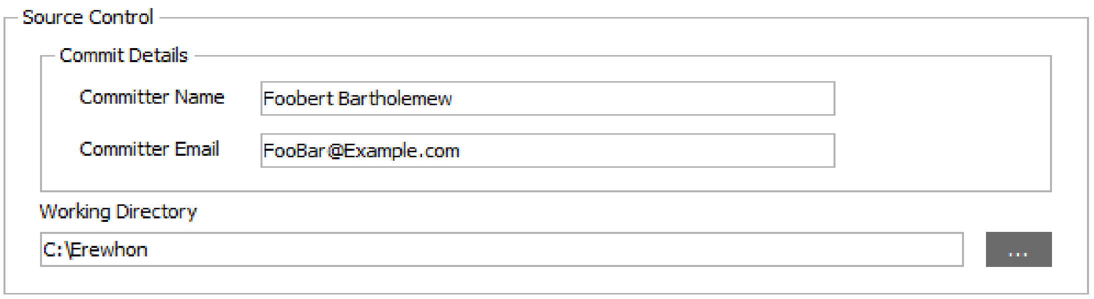
You can perform many of the functions in this table without leaving the JADE development environment, using the Git Source Control Client submenu of the Browse menu.



Exercise 4 – Cloning a Repository

In this exercise, you will clone the Erewhon repository from the Jade Software public GitHub and load it into your database.

1. Select the **Preferences** command from the Options menu.
2. On the **Source Management** sheet, fill out the Source Control group box as follows.



Source Control

Commit Details

Committer Name: Foobert Bartholemew

Committer Email: FooBar@Example.com

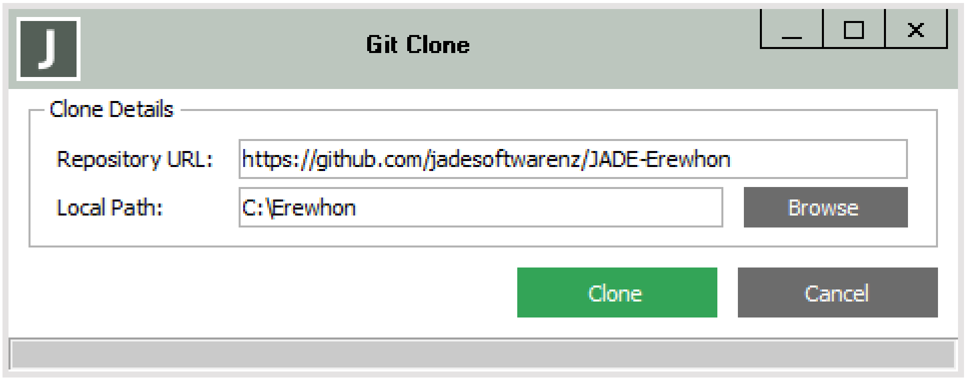
Working Directory: C:\Erewhon

- For the **Committer Name**, enter your name
- For the **Committer Email**, enter your e-mail address
- For the **Working Directory**, select **C:/Erewhon**

Note The name and e-mail address must be set. However, they are not validated unless you are attempting to push to the remote repository, which we will not be doing in this exercise.

3. Navigate to the **C:/** drive in your file system and add a new directory of **C:/Erewhon**. This will be the directory of your local repository.
4. Select the **Clone** command from the Git Source Control Client submenu of the Browse menu.

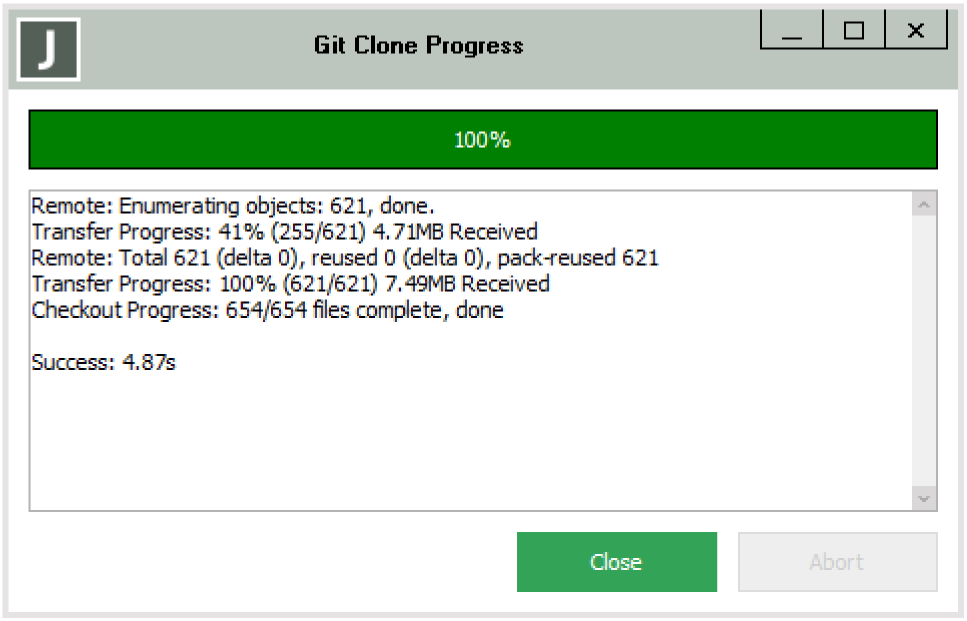
In the Git Clone dialog that is then displayed, enter the following.



Repository URL: https://github.com/jadesoftwarenz/JADE-Erewhon

Local Path: C:\Erewhon

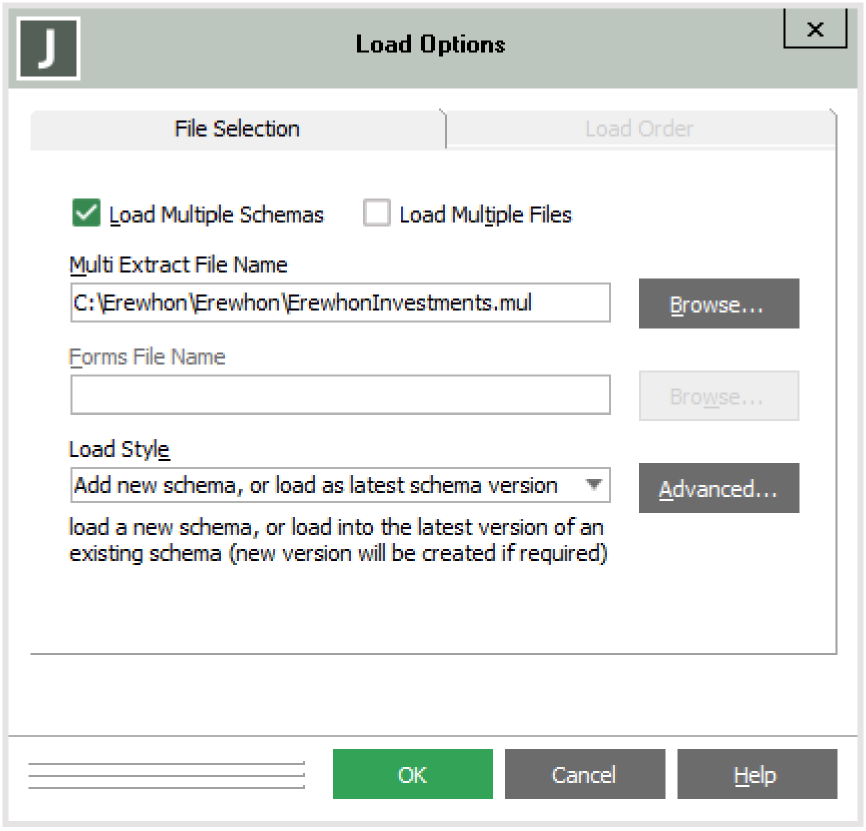
Click **Clone**. The Git Clone Progress dialog is then displayed.



```
Remote: Enumerating objects: 621, done.
Transfer Progress: 41% (255/621) 4.71MB Received
Remote: Total 621 (delta 0), reused 0 (delta 0), pack-reused 621
Transfer Progress: 100% (621/621) 7.49MB Received
Checkout Progress: 654/654 files complete, done

Success: 4.87s
```

5. Navigate to the **C:\Erewhon** folder. You will see that the Erewhon files have been cloned into the directory.
6. Select the **Load** command from the Schema menu.

7. On the Load Options dialog, check the **Load Multiple Schemas** check box, click **Browse** at the right of the **Multi Extract File Name**, and then select **ErewhonInvestments.mul** in the **C:\Erewhon\Erewhon** directory.



[✓] Load Multiple Schemas   [ ] Load Multiple Files

Multi Extract File Name: C:\Erewhon\Erewhon\ErewhonInvestments.mul

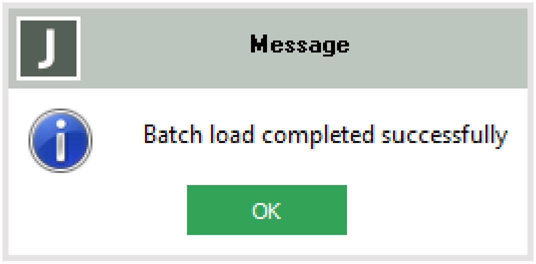
Forms File Name:

Load Style: Add new schema, or load as latest schema version

load a new schema, or load into the latest version of an existing schema (new version will be created if required)

Click **OK**.

The Erewhon schemas are large, so it may take around a minute to load, at which point the following message box is displayed.

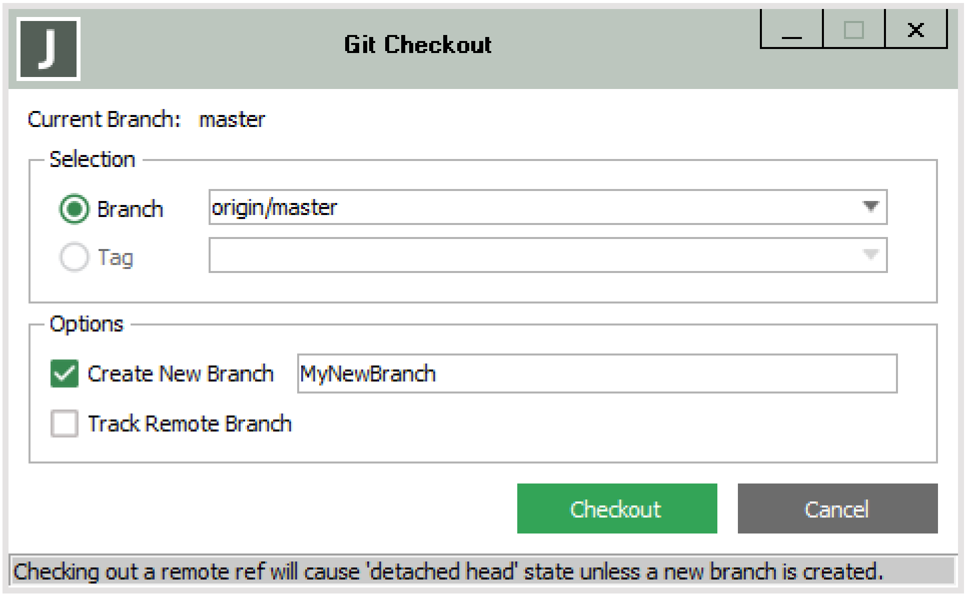


Batch load completed successfully

Exercise 5 – Checking Out a Branch in Git

In this exercise, you will check out a new local branch, modify this branch, and then merge it back into the main branch.

1. Select the **Checkout** command from the Git Source Control Client submenu of the Browse menu.



Current Branch: master

[✓ Branch] origin/master

Tag

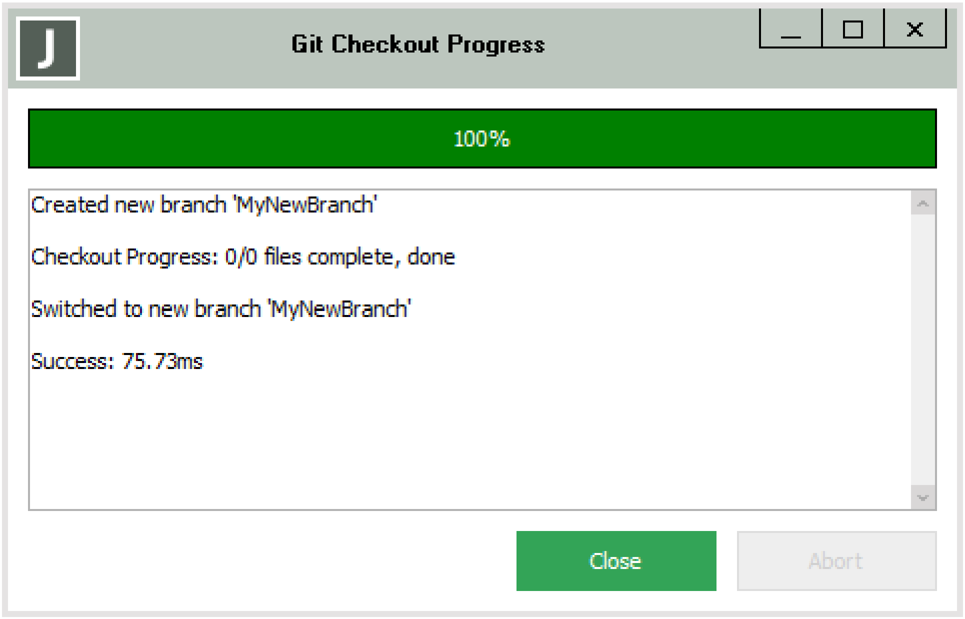
✓ Create New Branch MyNewBranch

Track Remote Branch

Checking out a remote ref will cause 'detached head' state unless a new branch is created.

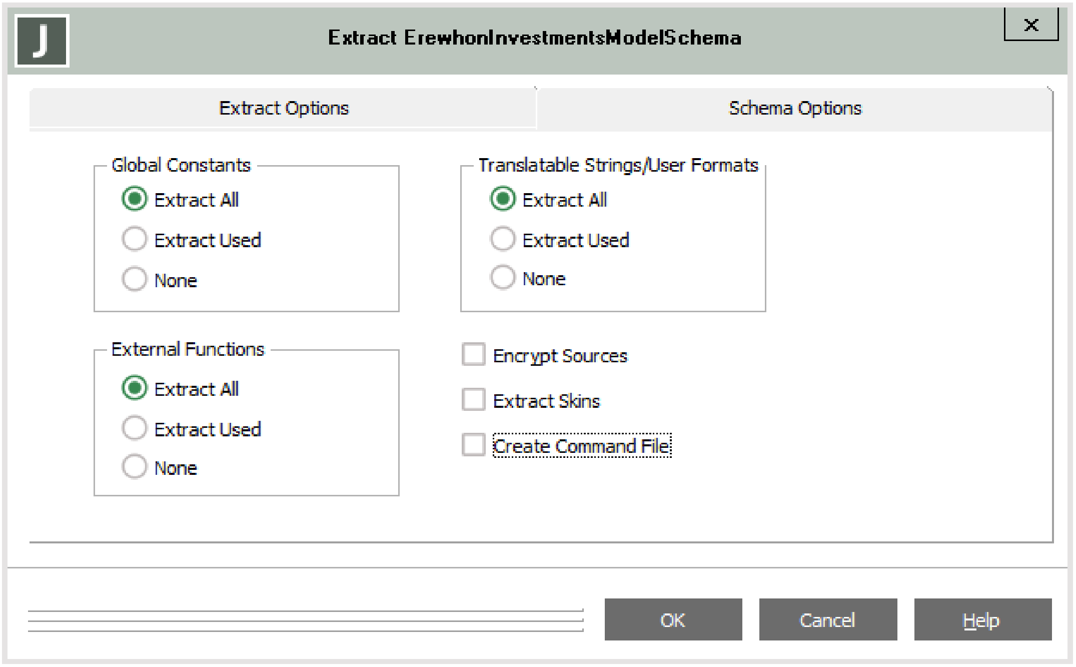
- Uncheck the **Track Remote Branch** check box
- Enter **MyNewBranch** in the **Create New Branch** text box
- Click **Checkout**

The following confirmation message box is displayed upon success.



```
Created new branch 'MyNewBranch'

Checkout Progress: 0/0 files complete, done

Switched to new branch 'MyNewBranch'

Success: 75.73ms
```

2. In the **ErewhonInvestmentsModelSchema**, create a JadeScript method called **newMethod**. This method does not need any code, as it is needed only to make an arbitrary change to the schema.
3. Right-click **ErewhonInvestmentsModelSchema** in the Schema Browser and then select **Extract**.
4. On the **Schema Options** sheet, uncheck the **Create Command File** check box.



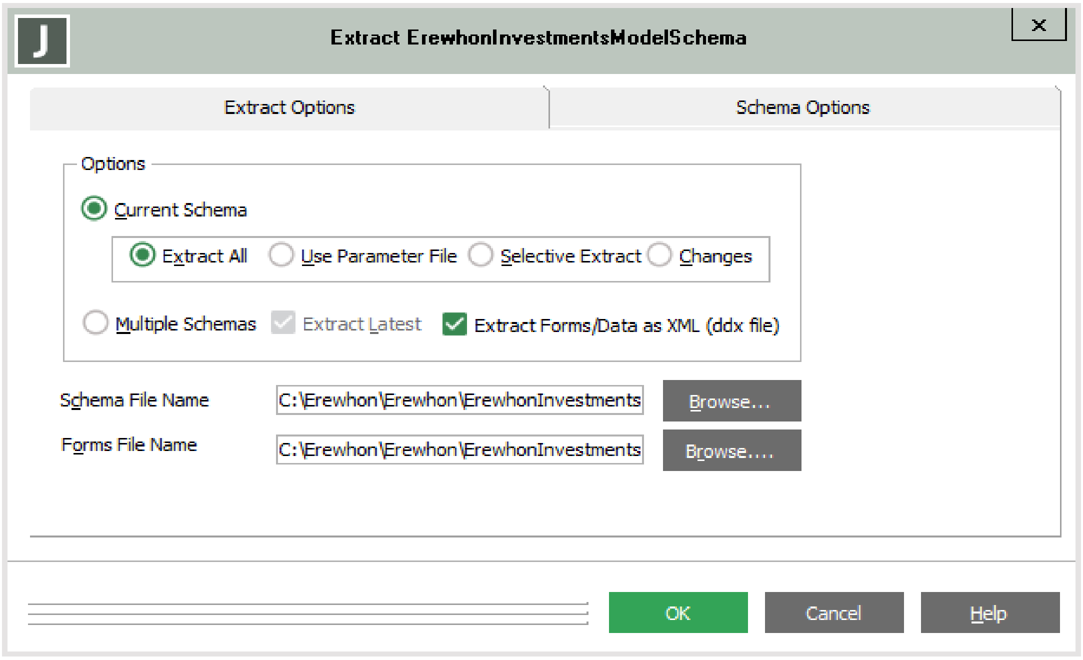
Global Constants: [✓ Extract All] Extract Used None

Translatable Strings/User Formats: [✓ Extract All] Extract Used None

External Functions: [✓ Extract All] Extract Used None

☐ Encrypt Sources
☐ Extract Skins
☐ Create Command File

5. On the **Extract Options** sheet, check the **Extract Forms/Data as XML (ddx file)** check box, click **Browse** at the right of the **Schema File Name** text box, and then select **C:\Erewhon\Erewhon** as the target directory. Click **OK**.



Options: [✓ Current Schema] Multiple Schemas

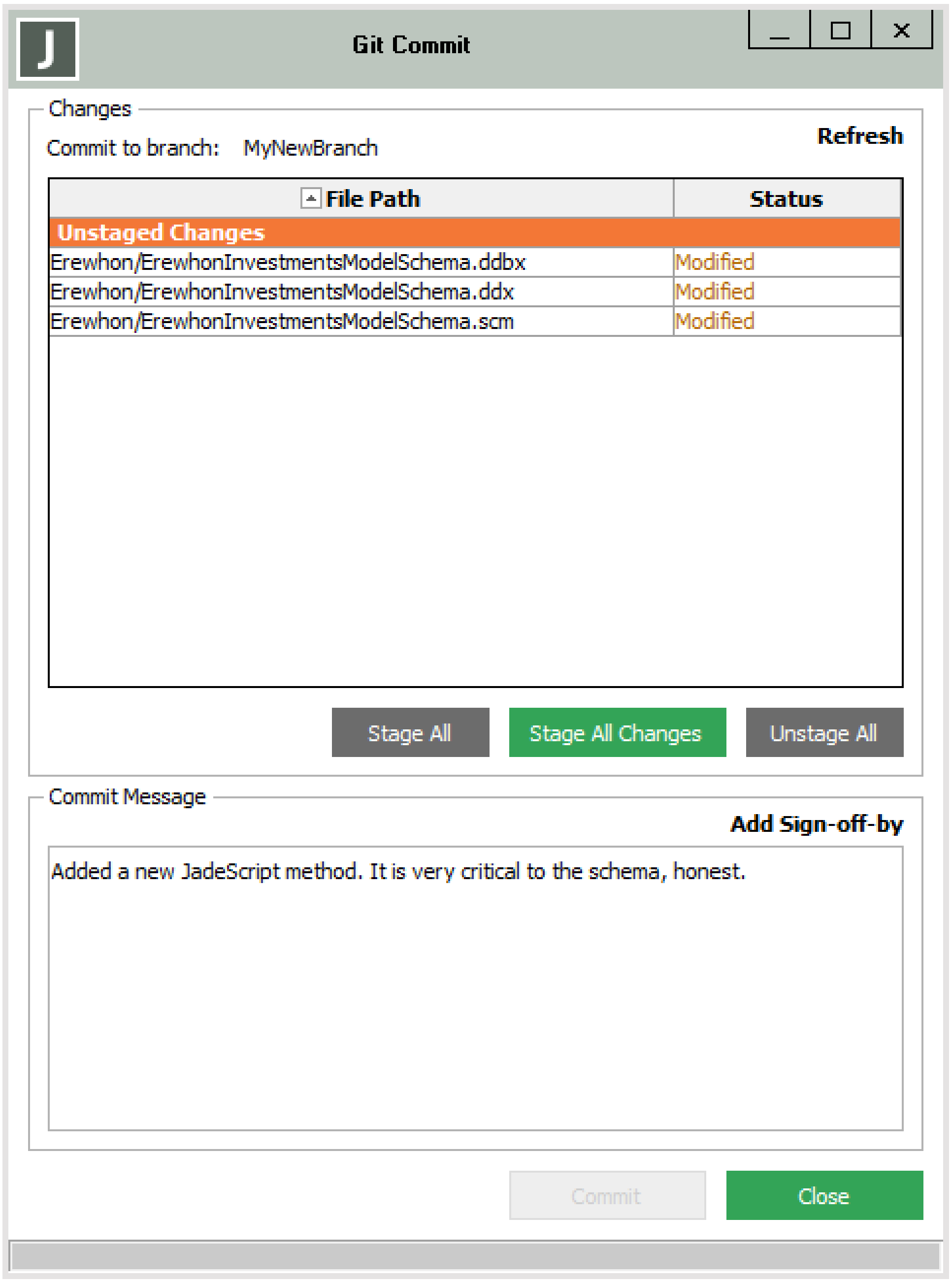
[✓ Extract All] Use Parameter File Selective Extract Changes

☑ Extract Latest ☑ Extract Forms/Data as XML (ddx file)

Schema File Name: C:\Erewhon\Erewhon\ErewhonInvestments

Forms File Name: C:\Erewhon\Erewhon\ErewhonInvestments

6. Select the **Commit** command from the Git Source Control Client submenu of the Browse menu.



Commit to branch: MyNewBranch

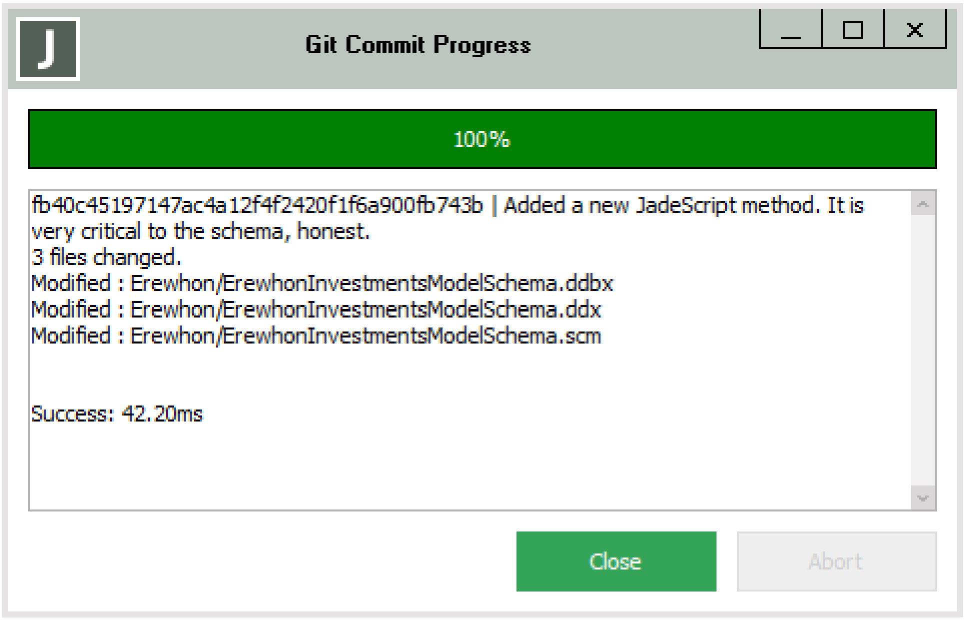
File Path	Status
Unstaged Changes	
Erewhon/ErewhonInvestmentsModelSchema.ddbx	Modified
Erewhon/ErewhonInvestmentsModelSchema.ddx	Modified
Erewhon/ErewhonInvestmentsModelSchema.scm	Modified

Added a new JadeScript method. It is very critical to the schema, honest.

The Git Commit dialog displays the files that have changed since the last commit action and allows you to stage some or all changes and then commit them. Right-clicking any of the unstaged changes allows you to stage the specific file, and clicking **Stage All** or **Stage All Changes** selects all files for the commit. When you are happy with the files to be committed, enter a commit message describing what has changed and why. The **Commit** button is then enabled.

Note The difference between **Stage All** and **Stage All Changes** is that the **Stage All Changes** action stages only those files that already exist in the repository and have been changed, whereas **Stage All** includes files that are newly created and do not yet exist in the repository.

7. Stage all changes, enter any message you like as the commit message, then click **Commit**.

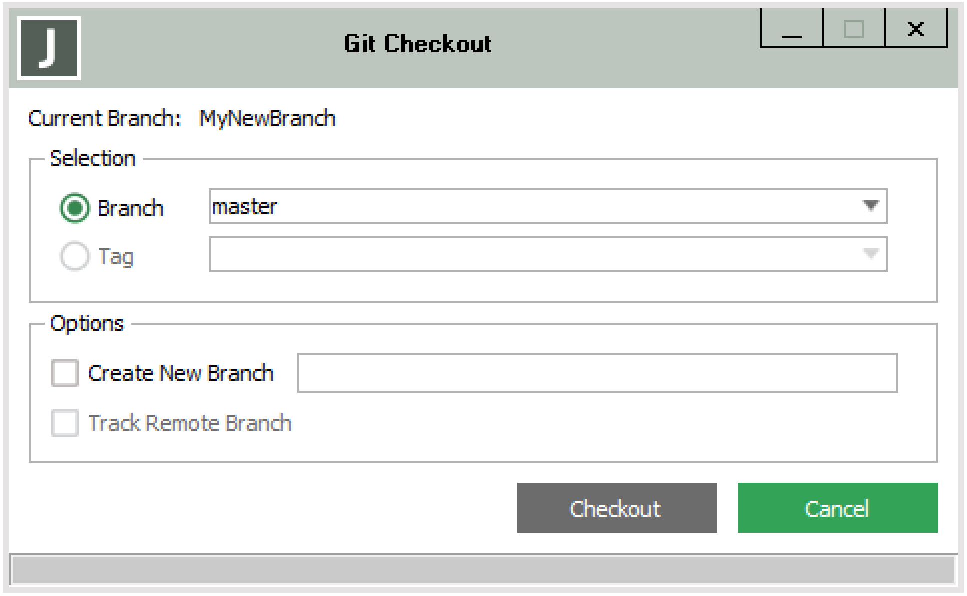


```
fb40c45197147ac4a12f4f2420f1f6a900fb743b | Added a new JadeScript method. It is
very critical to the schema, honest.
3 files changed.
Modified : Erewhon/ErewhonInvestmentsModelSchema.ddbx
Modified : Erewhon/ErewhonInvestmentsModelSchema.ddx
Modified : Erewhon/ErewhonInvestmentsModelSchema.scm


Success: 42.20ms
```

Exercise 6 – Back to the Master Branch

In this exercise, you will check out back to the master branch, effectively reverting the changes made while in the **MyNewBranch** branch.

1. Select the **Checkout** command from the Git Source Control Client submenu of the Browse menu.



Current Branch: MyNewBranch

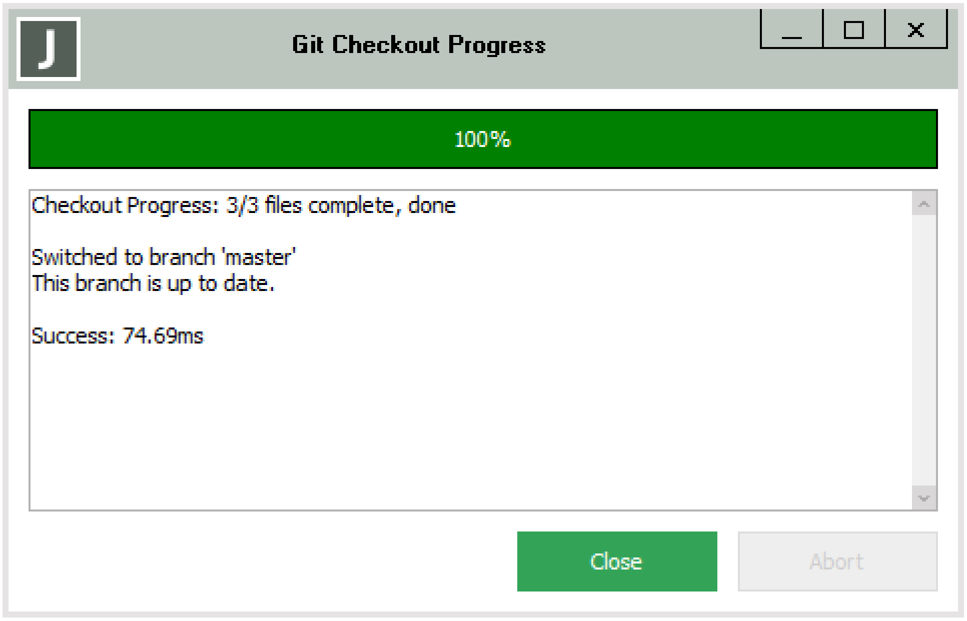
[✓ Branch] master

Tag

Create New Branch

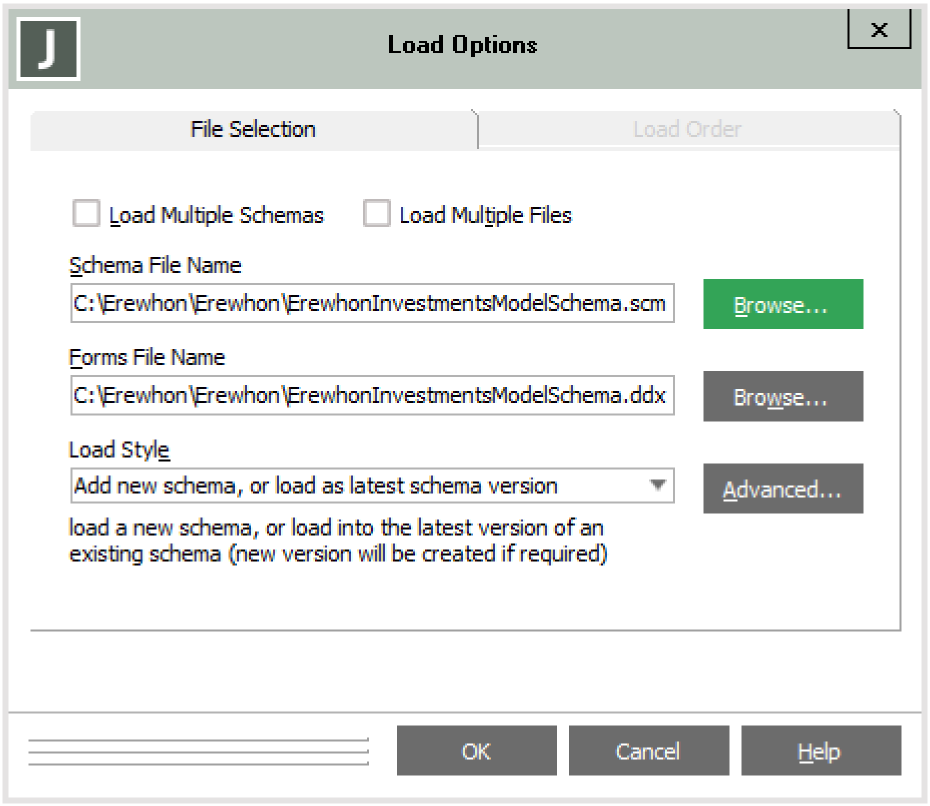
Track Remote Branch

2. Click **Checkout**, to switch back to master.



```
Checkout Progress: 3/3 files complete, done

Switched to branch 'master'
This branch is up to date.

Success: 74.69ms
```

3. Select the Load command from the Schema menu.



Load Multiple Schemas ☐ Load Multiple Files ☐

Schema File Name: C:\Erewhon\Erewhon\ErewhonInvestmentsModelSchema.scm

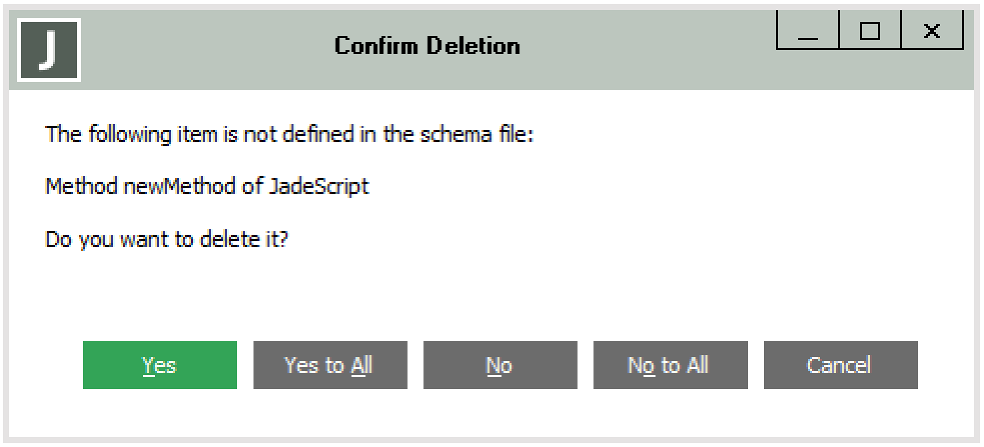
Forms File Name: C:\Erewhon\Erewhon\ErewhonInvestmentsModelSchema.ddx

Load Style: Add new schema, or load as latest schema version

load a new schema, or load into the latest version of an existing schema (new version will be created if required)

Click **Browse** at the right of the **Schema File Name** text box, navigate to the **ErewhonInvestmentsModelSchema.scm** file in **C:\Erewhon\Erewhon**, and then click **OK**.

4. Part-way through the load process, the following message box is displayed. Click **Yes**, to remove the JadeScript **newMethod** method from the schema.



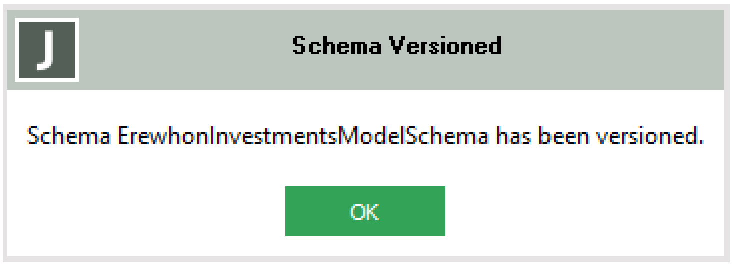
Confirm Deletion

The following item is not defined in the schema file:

Method newMethod of JadeScript

Do you want to delete it?

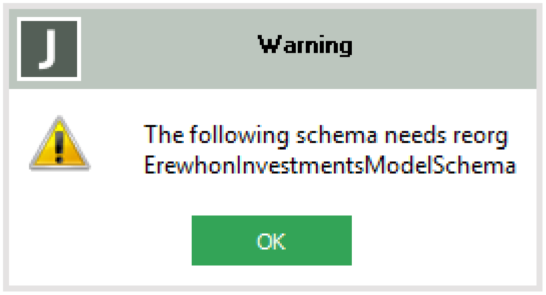
This change causes the schema to be versioned.



Schema Versioned

Schema ErewhonInvestmentsModelSchema has been versioned.

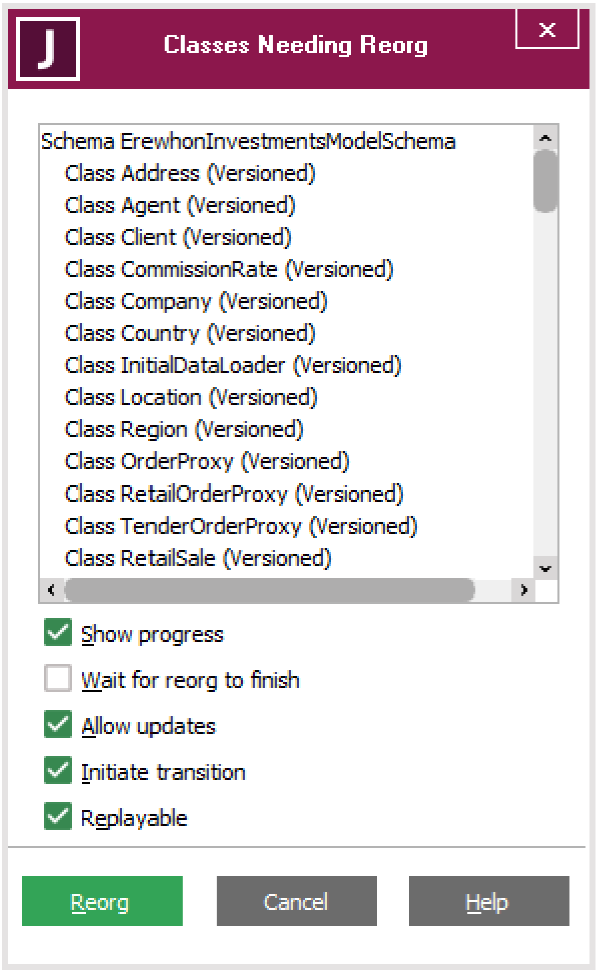
Clicking **OK** results in the display of the following message box.



Warning

The following schema needs reorg
ErewhonInvestmentsModelSchema

5. Click the **Reorg** toolbar button in the Class Browser and click **Reorg** in the Classes Needing Reorg dialog that is then displayed.



Schema ErewhonInvestmentsModelSchema
Class Address (Versioned)
Class Agent (Versioned)
Class Client (Versioned)
Class CommissionRate (Versioned)
Class Company (Versioned)
Class Country (Versioned)
Class InitialDataLoader (Versioned)
Class Location (Versioned)
Class Region (Versioned)
Class OrderProxy (Versioned)
Class RetailOrderProxy (Versioned)
Class TenderOrderProxy (Versioned)
Class RetailSale (Versioned)

[✓] Show progress
[ ] Wait for reorg to finish
[✓] Allow updates
[✓] Initiate transition
[✓] Replayable

6. When the reorganization completes, navigate to the **JadeScript** class within **ErewhonInvestmentsModelSchema**.

   You will see that the **newMethod** method has been removed; that is, the schema has reverted to the state it was when you switched from the master branch to the **MyNewBranch** branch.