**Synopsis**

The learning objectives of this seminar are same as that of the lecture for this week. Please, if you have not done yet, watch the lecture before you attempt this seminar (or do both in parallel).

This laboratory session is intended to enable you to design following UML diagrams in Violet UML Editor.

1. Sequence diagram

2. Object diagram

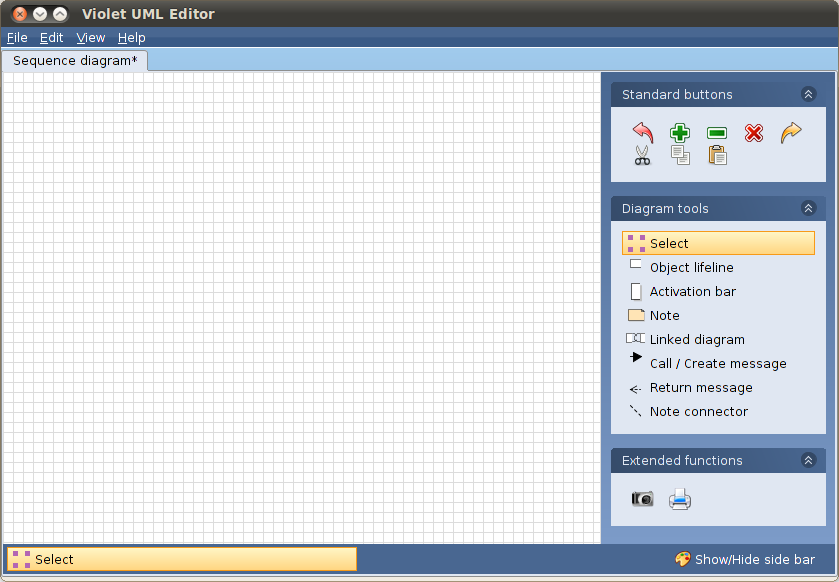
*Note: If you did not attend the seminar last week, please see the instruction to download and use Violet UML editor at the end of this file.*

# **Using Violet for Sequence Diagrams**

This section describes most of the sequence diagram operations that you can perform in Violet.

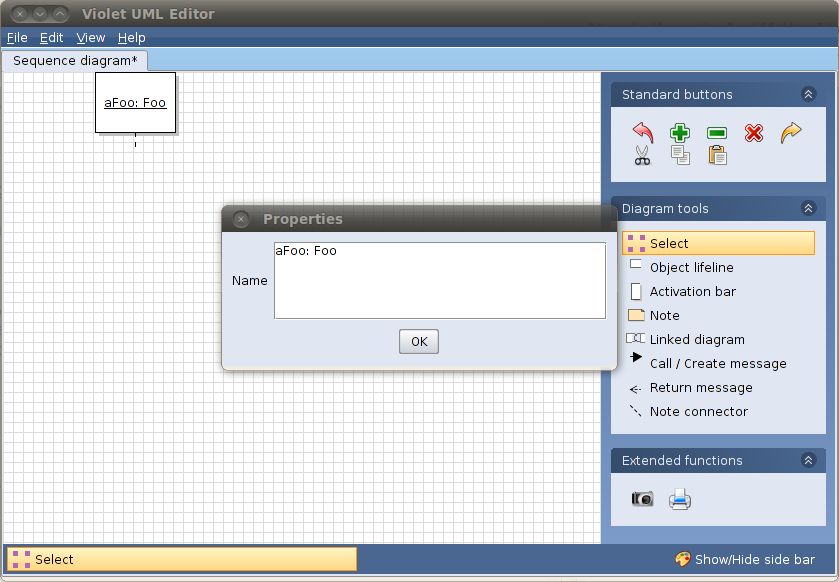
#### Creating a New Sequence Diagram

After launching Violet, either click sequence diagram on the opening screen or use the File → New menu option to get:



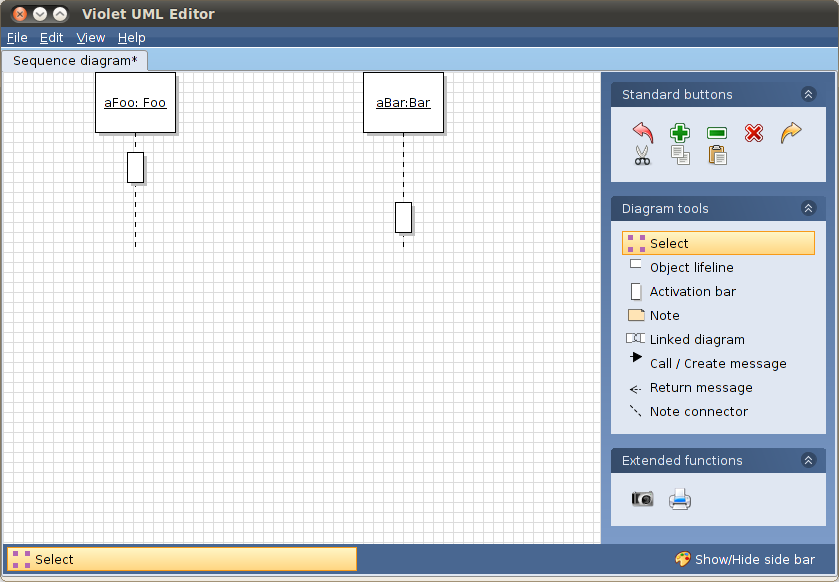
#### Creating an Object Lifeline

Under Diagram tools select Object lifeline and click in the drawing area to create an empty box. Right click on the box to get a dialog that accepts the object's name. In this case an object named aFoo of type Foo is being created:



#### Creating Activation Bars

An activation bar represents a method call on an object. To create one select the Activation bar tool and click on an object's lifeline. Below two objects with activation bars have been created. Note that at this point it is assumed that aFoo's method exits before aBar's method starts.

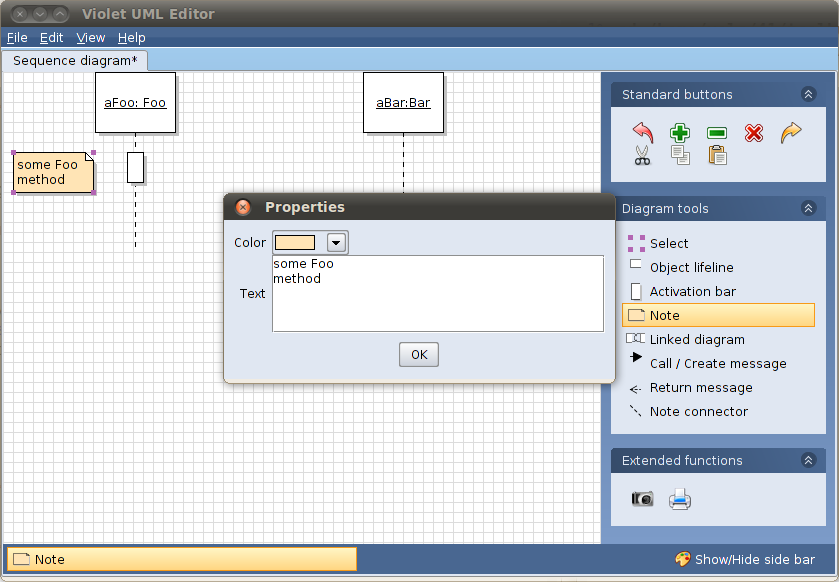


#### Diagram Notes

Sequence diagram notes provide information and make diagrams easier to read. They are created in the same way class diagram notes are created.

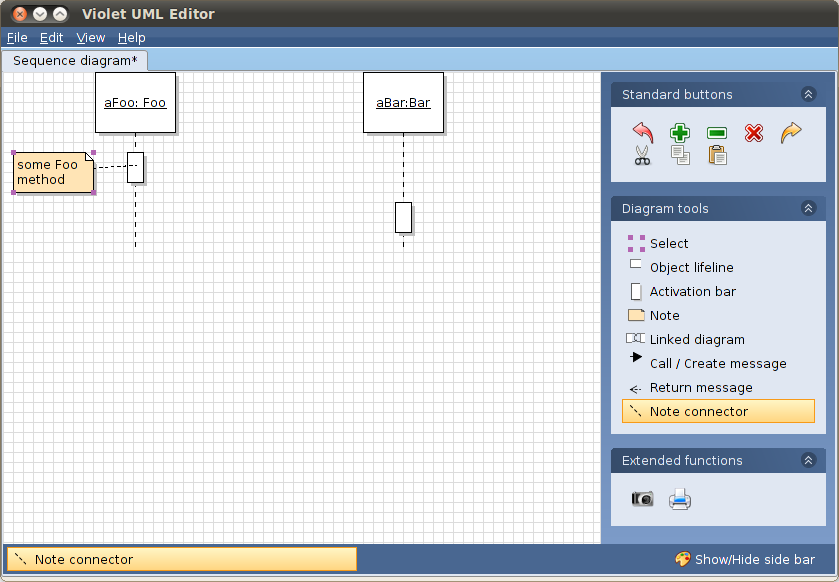
**Creating a Note:**

Select the Note tool and click where you want the note to appear in the diagram. Right click the note to get a dialog accepting the text of the note. (Note that you can also change the note's background color.) In this case we are indicating the Foo method that is represented by aFoo's activation bar:



**Connecting a Note:**

To connect the note to a specific point in the diagram, select the Note connector tool and drag from the note to the specific point:

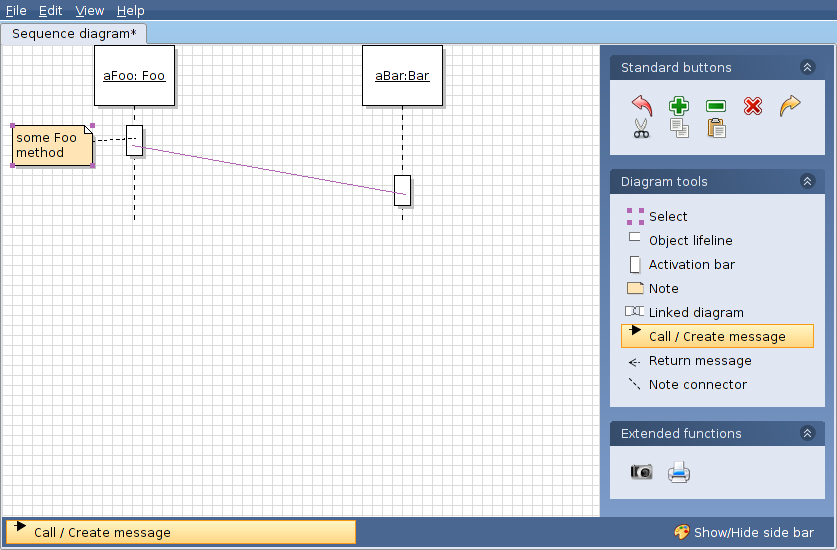


#### Method Calls

To show how objects collaborate to produce a result, sequence diagrams model method calls, which can be thought of as sending a message from one object to another. This example will show aFoo calling aBar's method someBarMethod with value 1 being returned.

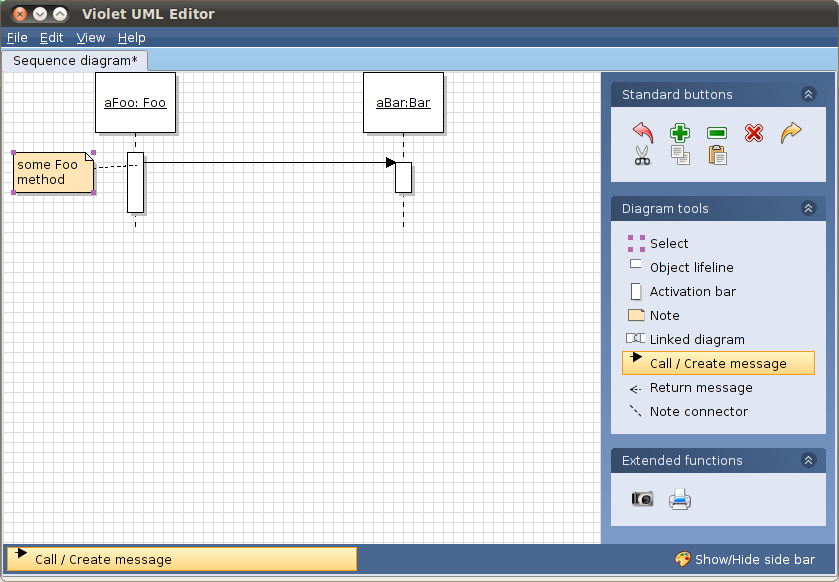
**Connect Two Activation Bars**:

Select the Call / Create message tool and drag from the calling object's activation bar to the receiving object's activation bar:



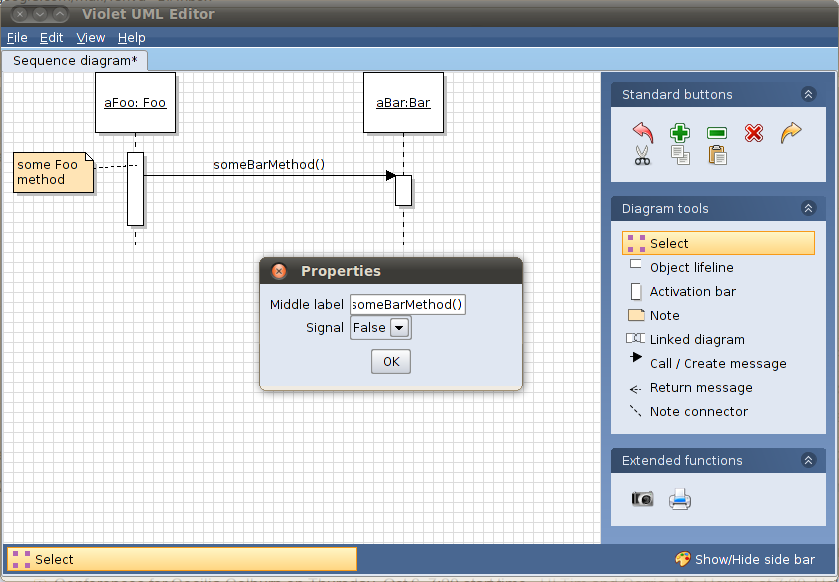
**Realign the Activation Bars**:

Upon release of the mouse, Violet will realign the objects' activation bars:



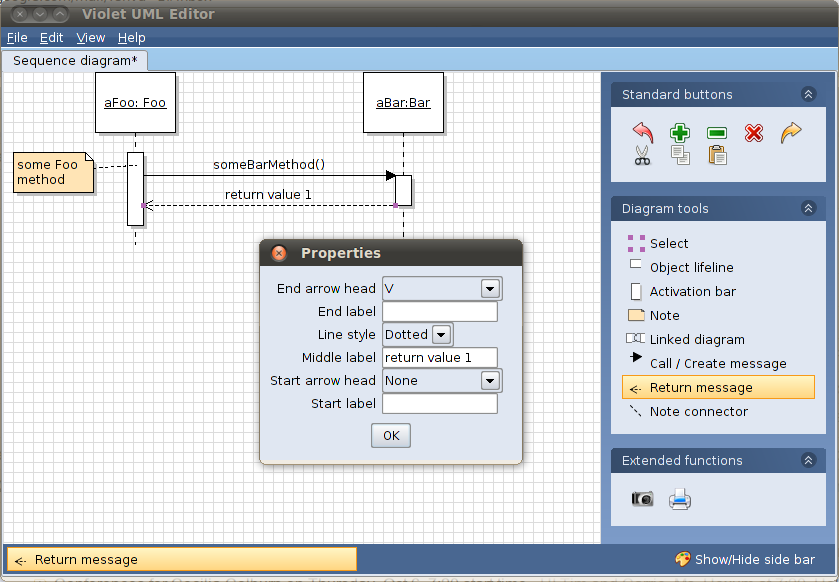
**Label the Method Call**:

Right click the connecting line to get a dialog that accepts the name of the method being called, with or without arguments:



**Indicate the Return Value**:

Select the Return message tool and drag from the receiving object's activation bar to the sending object's activation bar. Right click the connecting line to get a dialog that accepts the value being returned:

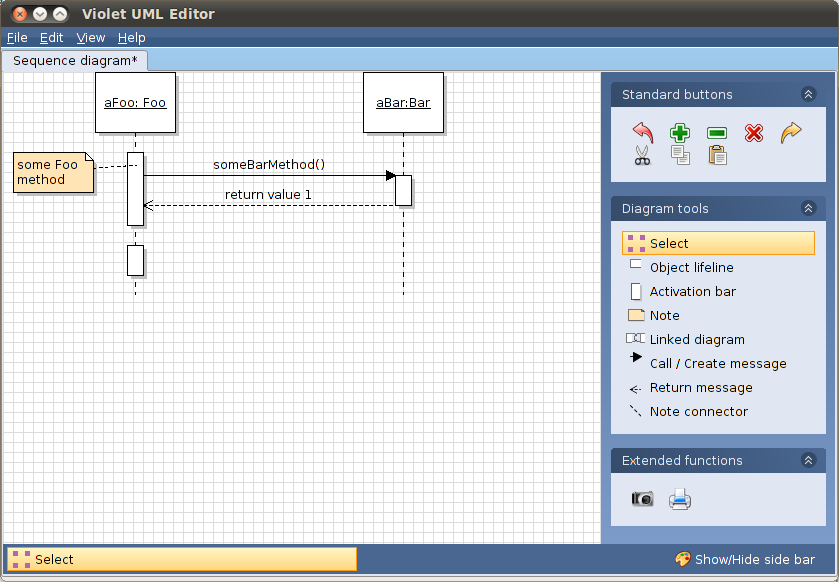


#### Self Calls

Sometimes it is necessary to model an object calling its own method. In this example we show aFoo's method someFooMethod calling its own method anotherFooMethod. There does not appear to be any good way to indicate the return value of a self call in Violet.

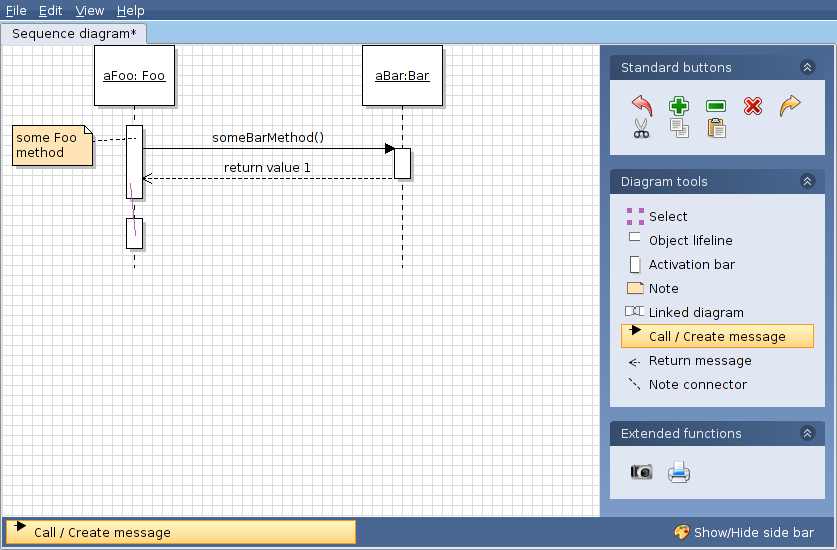
Create a New Activation Bar:

Start by creating a new activation bar for the object that is doing the self calling by using the Activation bar tool. Hang the new activation bar on the activation bar of the method that is doing the self calling:



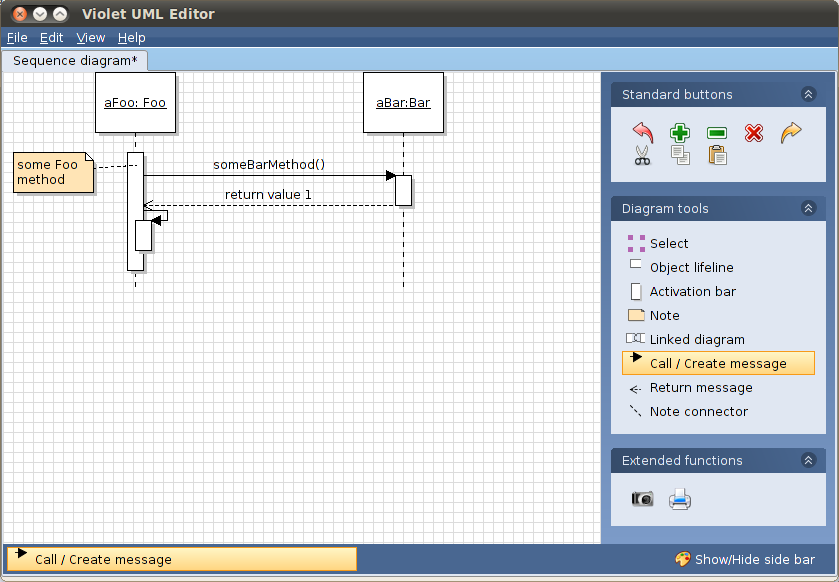
Connect the Activation Bars:

Select the Call / Create message tool and drag from the object's upper activation bar down to the lower activation bar that was just created:



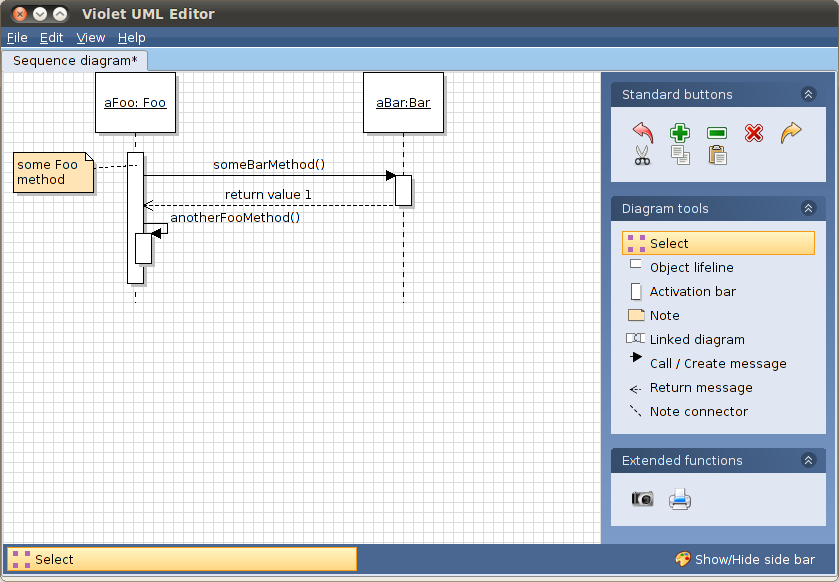
Redraw As A Self Call:

Upon release of the mouse, Violet will redraw the connection as a self call:



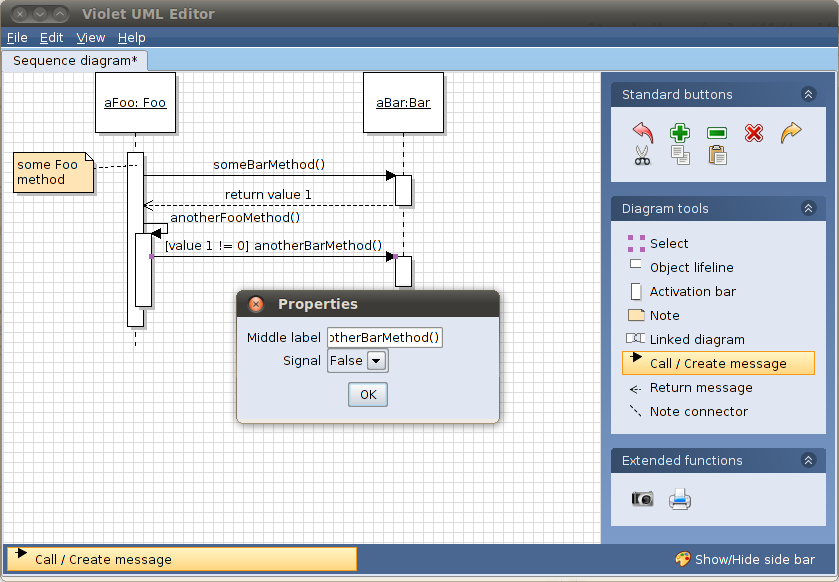
Label the Self Call:

Right click the self call connection to get a dialog accepting the name of the method being called:



#### Conditional Method Calls

To indicate conditional method calls, simply label the call line with a prefix "[condition]" where condition is a boolean valued expression. Below, aFoo calls aBar's method anotherBarMethod only if value 1 is not equal to zero. Note that a second activation bar has been added to aBar's lifeline before making the connection:



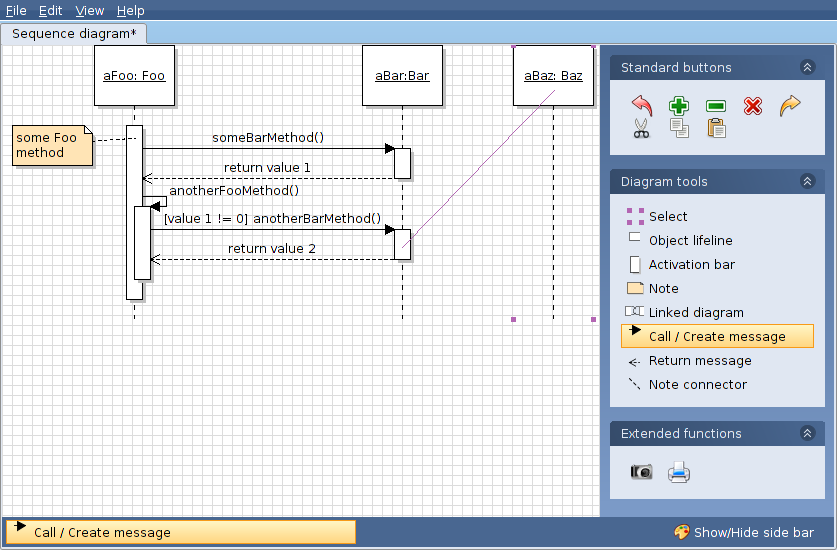
#### Modeling the Creation of Objects

Sometimes a sequence diagram needs to model the creation of an object.

In this example the object aBaz of type Baz is created during execution of aBar's method anotherBarMethod.

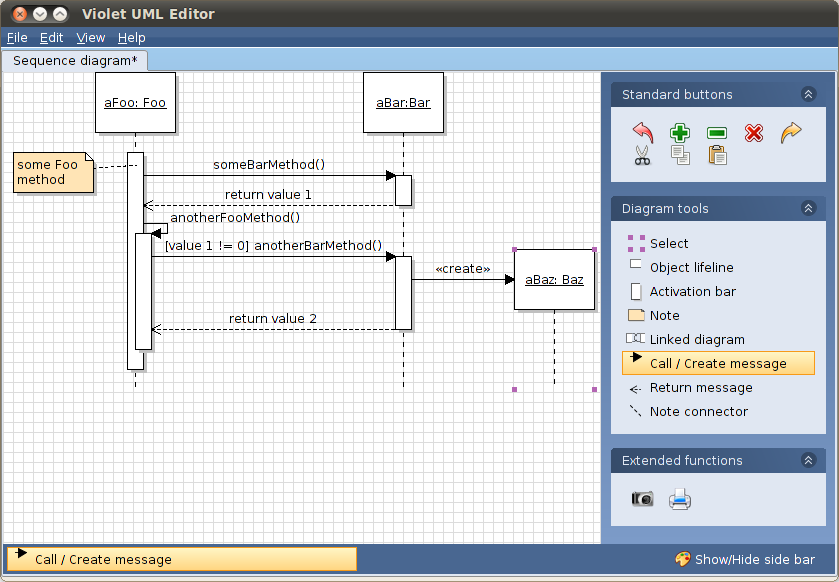
**Connecting an Activation Bar to an Object:**

First create the new object's lifeline using the Object lifeline tool. Do not create an activation bar for the new object. Using the Call / Create message tool connect the activation bar of the method that creates the new object directly to that object:



**Realigning to Show Object Creation:**

Upon release of the mouse, Violet will realign the object to indicate its creation. It will also automatically label the connection with the «create» stereotype:



# **Object diagrams**

The creation of an object diagram is done in a way similar to a class diagram by choosing File > New > Object Diagram. Please refer to today’s lecture and last week’s seminar exercise on how to create a class diagram.

# **Download and Install Violet UML Editor**

What is Violet? Violet is a UML editor with these benefits:

1. It is very easy to learn and use
2. It draws nice-looking diagrams of the most commonly used types (class, sequence, state, etc.)
3. It is completely free (includes source, distributed under the GNU General Public License)
4. It is cross-platform

Violet is intended for students, teachers, and authors who need to produce simple UML diagrams quickly. It is not intended as an industrial strength tool. Programs such as Rational Rose, StarUML, ArgoUML, and so on, are fine choices for serious users of UML. However students and casual users may not like them because they are too hard to learn or too expensive. If you just want to draw simple UML diagrams without too much fuss, chances are you'll like Violet. If you have more serious needs, check out one of the other programs.

Downloading and Running Violet

1. Violet comes as an executable Java archive file which you can download here: <https://www.d.umn.edu/~gshute/cs2511/javalabs/uml_diagrams/violet.jar>
2. You must have Java installed on your machine.
3. On some operating systems you can launch Violet by ***double-clicking its icon (i.e downloaded .jar file, it works on my computer)***. If that doesn't work for you, open a command window, change to the folder where your violet.jar file is located, and run the command:

***java -jar violet.jar***

You should see a screen that looks like:

