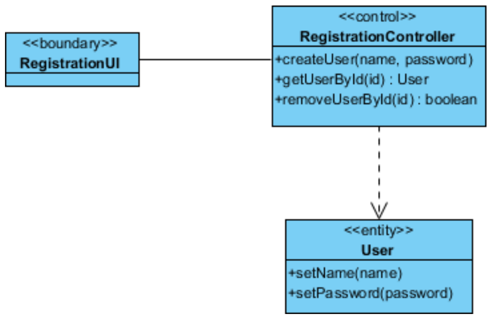
How to Draw UML Sequence Diagram?

Written Date : March 16, 2016

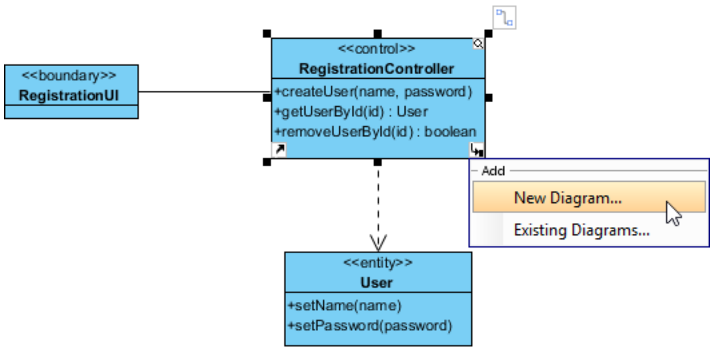
When you want to model the structure of a system or an application, you can make use of class diagram. When you want to model the interaction between objects in runtime, with the sequence of method invocation, you can make use of sequence diagram. Class diagram and sequence diagram can be related with each other. While a class in class diagram represents a blueprint of data, a lifeline in sequence diagram represents an instance of such blueprint. In this tutorial, we will start from a simple class diagram, and make use of a sequence diagram to model the dynamic method invocation related to a controller class modeled in class diagram.

Creating a Sequence Diagram from a Class

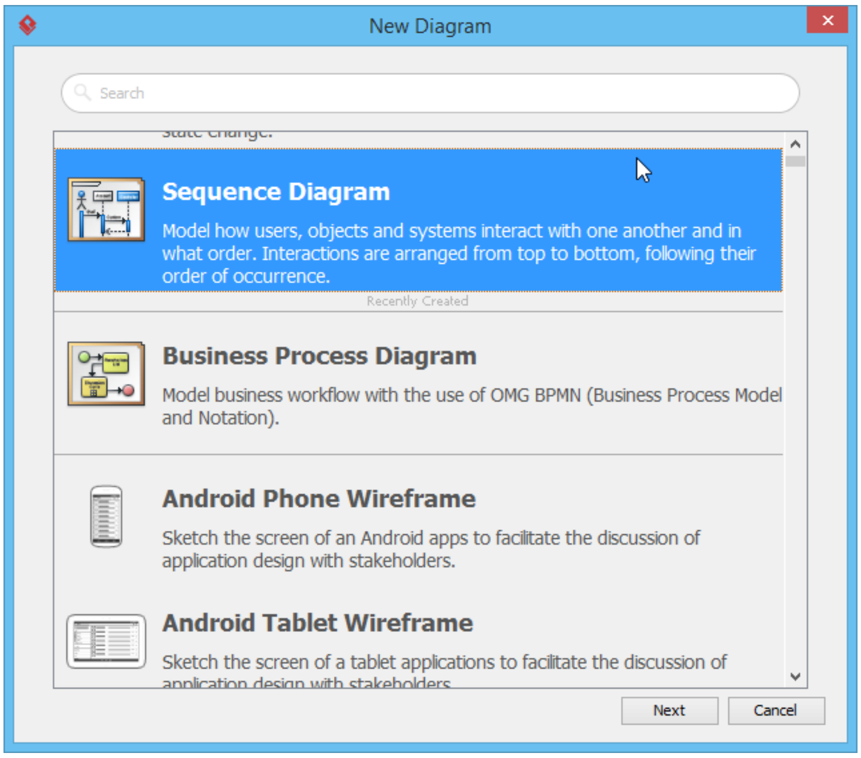
1. Download Simple-Registration.vpp. You can also find this file at the bottom of this tutorial.
2. Open the downloaded .vpp file in Visual Paradigm. To open a project, select **Project > Open** from the application toolbar.
3. Open the class diagram *Registration*. Study the diagram content. We have three classes - *RegistrationUI*, *RegistrationController* and *User*.



4. Now, we want to model the interaction between object instances of these classes in runtime. Since the controller class is responsible to control the registration process, add a sub-sequence diagram from it. Move the mouse pointer to *RegistrationController*. Click on the resource icon at the bottom right corner and select **New Diagram...** from the popup menu.



5. In the **New Diagram** window, select **Sequence Diagram** and click **Next**. Keep the diagram name as provided and click **OK** to confirm. This creates an empty UML sequence diagram.


New Diagram

Search

Sequence Diagram
Model how users, objects and systems interact with one another and in what order. Interactions are arranged from top to bottom, following their order of occurrence.

Recently Created

Business Process Diagram
Model business workflow with the use of OMG BPMN (Business Process Model and Notation).

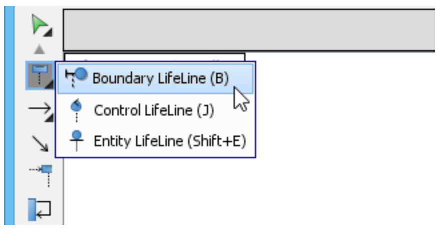
Android Phone Wireframe
Sketch the screen of an Android apps to facilitate the discussion of application design with stakeholders.

Android Tablet Wireframe
Sketch the screen of a tablet applications to facilitate the discussion of application design with stakeholders

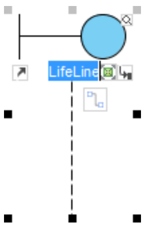
Next Cancel

Drawing the Sequence Diagram

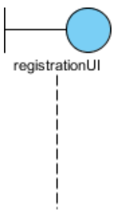
1. Select **Boundary Lifeline (B)** from the diagram toolbar.



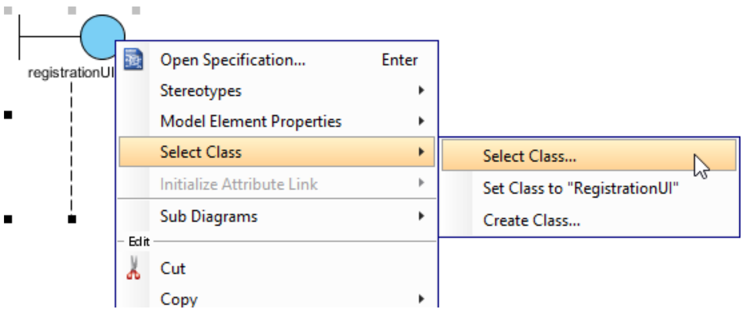
2. Click on the diagram to create a boundary lifeline.



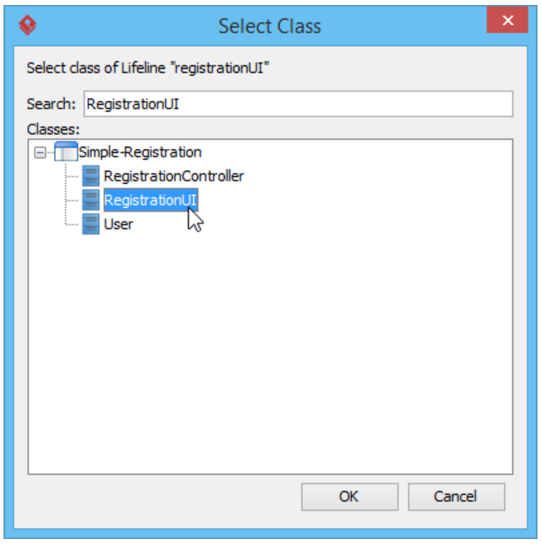
3. Enter *registrationUI* as the name of lifelife and then press **Enter** to confirm.



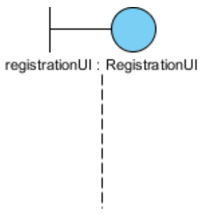
4. Right-click on the lifeline and select **Select Class > Select Class...** from the popup menu.



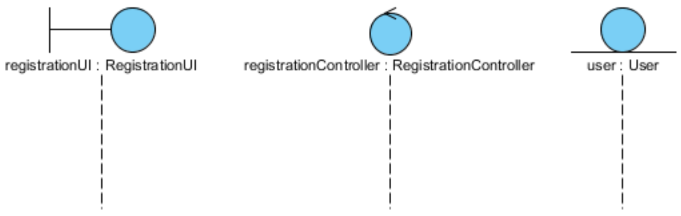
5. In the **Select Class** window, select the *RegistrationUI* class and then click **OK** to confirm.



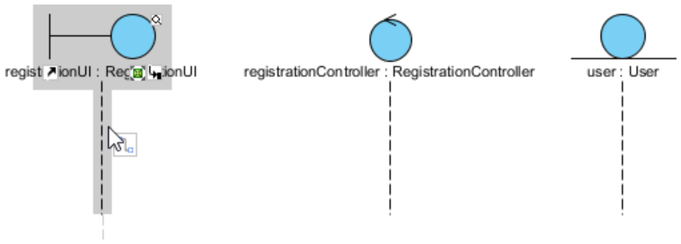
After that, the lifeline will look like the following.



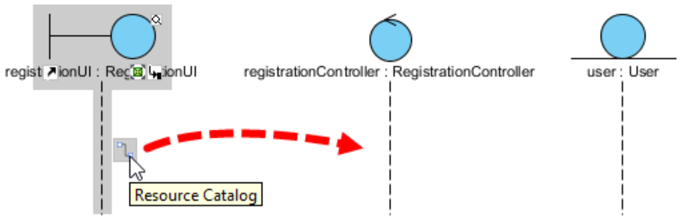
6. Create a **Control Lifeline** *registrationController : RegistrationController* and an **Entity Lifeline** *user : User*. Don't forget to select the appropriate classes for them. The diagram will look like the following.


registrationUI : RegistrationUI
registrationController : RegistrationController
user : User

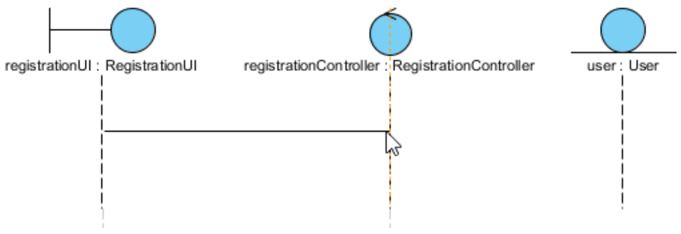
7. Let's model the method invocations between lifelines. Move the mouse pointer over lifeline *registrationUI*.


registrationUI : RegistrationUI
registrationController : RegistrationController
user : User

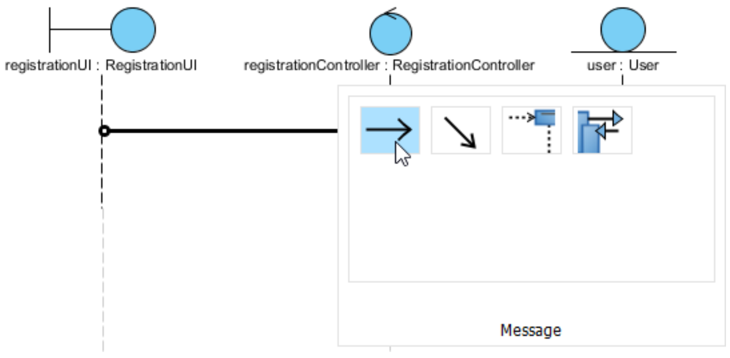
8. Press on the **Resource Catalog** resource and drag it out.


registrationUI : RegistrationUI
registrationController : RegistrationController
user : User
Resource Catalog

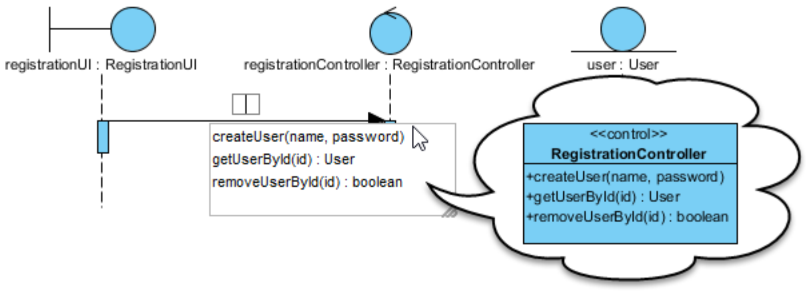
9. Move to lifeline *registrationController* and release the mouse button.


registrationUI : RegistrationUI
registrationController : RegistrationController
user : User

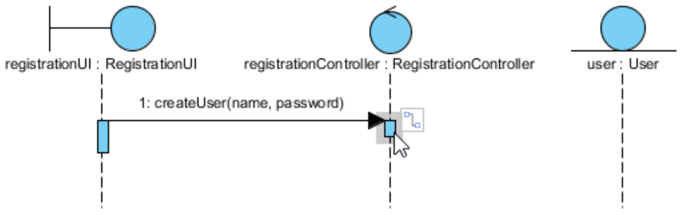
10. Select **Message** from Resource Catalog.


registrationUI : RegistrationUI
registrationController : RegistrationController
user : User
Message

11. This pops up a list of name that you can choose for the new sequence message. You can see that those are operation of *classRegistrationController*. Select *createUser(name, password)*.


registrationUI : RegistrationUI
registrationController : RegistrationController
user : User
createUser(name, password)
getUserById(id) : User
removeUserById(id) : boolean
<<control>>
RegistrationController
+createUser(name, password)
+getUserById(id) : User
+removeUserById(id) : boolean

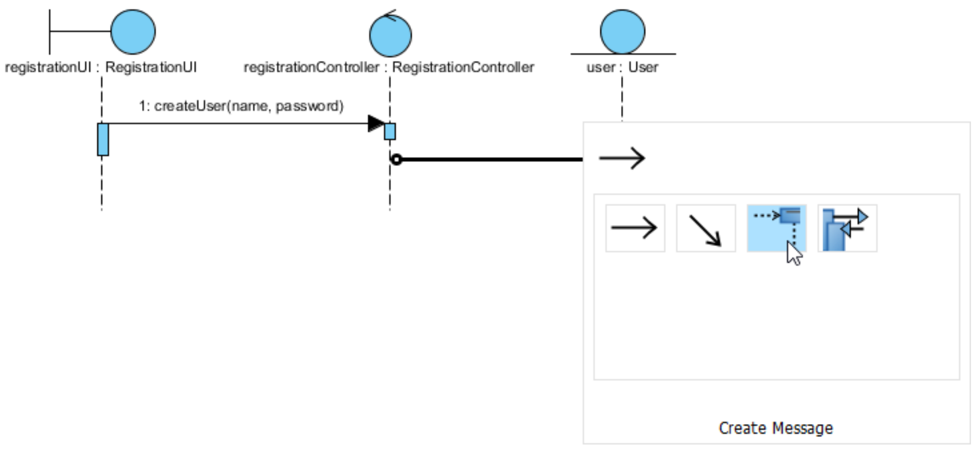
12. Relate lifeline *registrationController* and user. We say that *registrationController* creates the user lifeline. Therefore, we need to relate them with a create message. Move the mouse pointer over the activation in lifeline *registrationController*.


registrationUI : RegistrationUI
registrationController : RegistrationController
user : User
1: createUser(name, password)

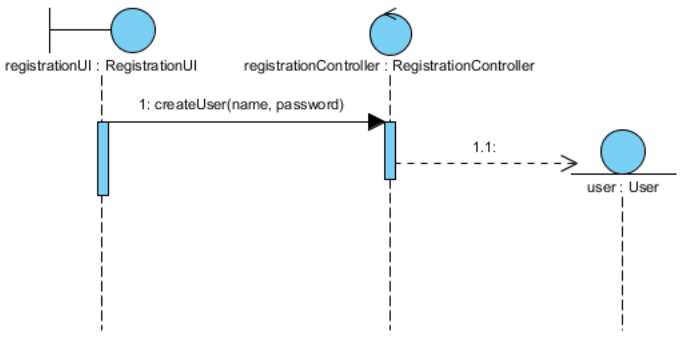
13. Press on the resource **Resource Catalog** and drag it out.

14. Release the mouse button on the lifeline *user*.

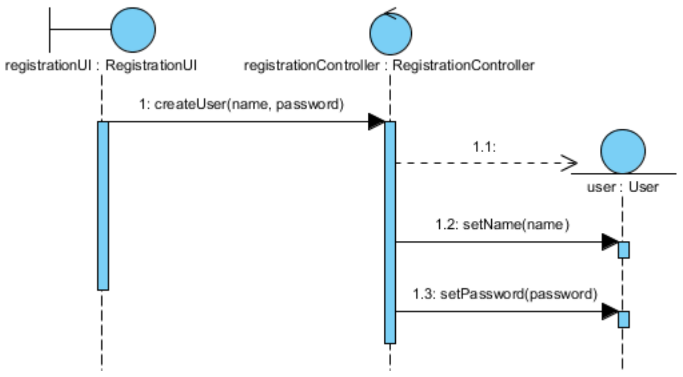
15. Select **Create Message** from Resource Catalog.


registrationUI : RegistrationUI
registrationController : RegistrationController
user : User
1: createUser(name, password)
Create Message

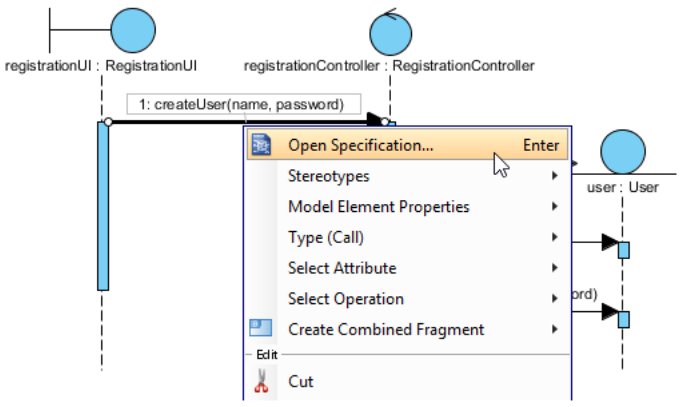
Create message is created. Your diagram should look like this:


registrationUI : RegistrationUI
registrationController : RegistrationController
1: createUser(name, password)
1.1:
user : User

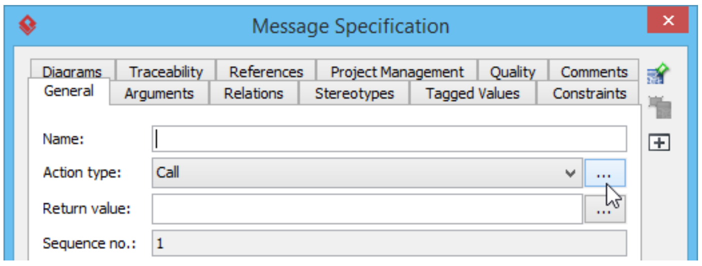
16. Create messages *setName* and *setPassword* from lifeline *registrationController* to user. Up to now, the diagram becomes:


registrationUI : RegistrationUI
registrationController : RegistrationController
1: createUser(name, password)
1.1:
user : User
1.2: setName(name)
1.3: setPassword(password)

17. You can also specify the arguments of operations. Take the message *createUser(name, password)* as example. Right-click on it and select **Open Specification...** from the popup menu.

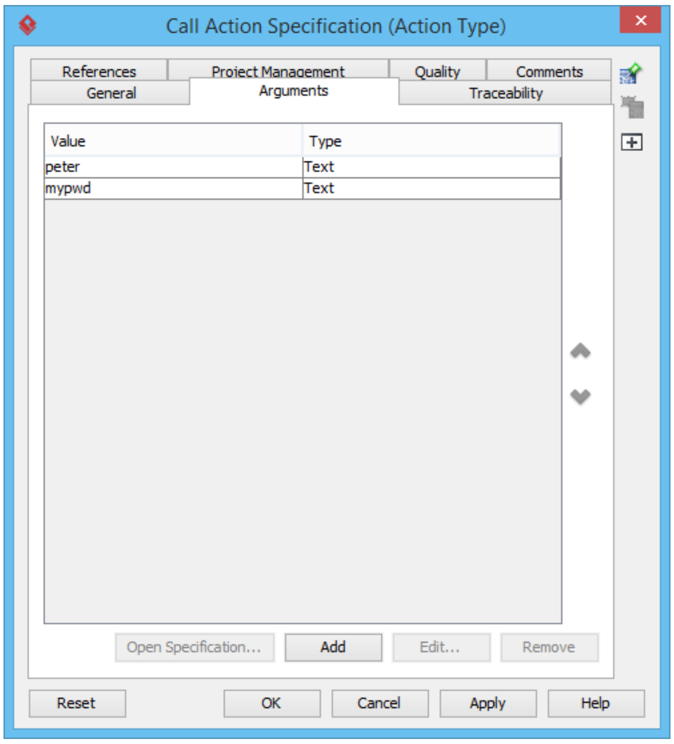

registrationUI : RegistrationUI
registrationController : RegistrationController
1: createUser(name, password)
user : User
Open Specification... Enter
Stereotypes
Model Element Properties
Type (Call)
Select Attribute
Select Operation
Create Combined Fragment
Edit
Cut

18. Edit the action type property by clicking on the button with dotted caption, next to **Action type**.



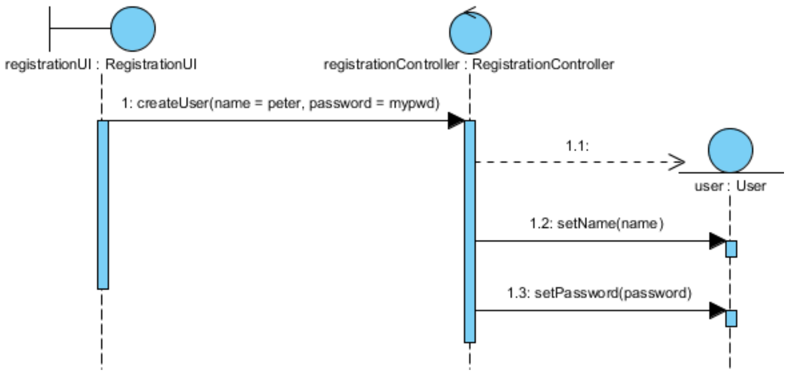
Name:	
Action type:	Call
Return value:	
Sequence no.:	1

19. In the **Call Action Specification** window, click **Add > Text...** to add an argument. In this example, click **Add > Text...** to add argument *peter*. Click **Add > Text...** again to add argument *mypwd*. Note that the two arguments are actually referring to the two parameters given by the operation. If you add the third argument here, it will be ignored (as there are only two operations defined).



Value	Type
peter	Text
mypwd	Text

20. Click **OK** to close the windows and go back to diagram. The arguments are added and presented on diagram. Finally the diagram become:


registrationUI : RegistrationUI
registrationController : RegistrationController
1: createUser(name = peter, password = mypwd)
1.1:
user : User
1.2: setName(name)
1.3: setPassword(password)

Resources
1. [What is Sequence Diagram?]
2. [Simple-Registration.vpp]

Related Links

- More UML sequence diagram toolset Full set of UML tools and UML diagrams