



## UML Getting Started - UML Modeling in Eclipse

Written Date : March 03, 2016

[Eclipse](#) is truly one of the best [integrated development environment](#) (IDE). To many software developers, Eclipse was the first IDE they ever used for serious software development. Eclipse is easy to use, and it comes with a lot of useful features to make implementation faster and more accurate, ultimately improving efficiency and software stability.

---

[Visual Paradigm](#) is an agile development platform that provides software developers with the wide variety of toolset they need for designing great software products. It supports a collection of widely-used modeling notations, such as [UML](#), [ERD](#) and [BPMN](#) and more. Software developers can either run Visual Paradigm in standalone manner, or to run it in embedded extension within the IDE. With this Eclipse UML tool, developers can perform visual modeling and other [agile development activities](#) within a unified Eclipse IDE platform.

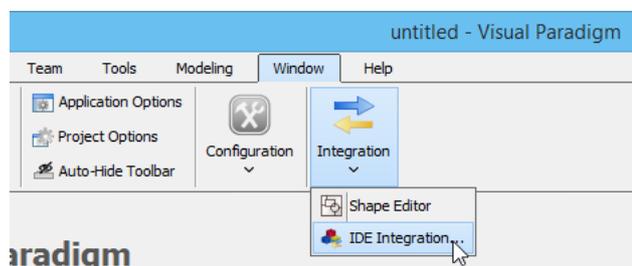
In this tutorial, you will walk through the steps required to integrate Visual Paradigm with Eclipse. The second part of the tutorial will demonstrate the automatic generation of Java code from UML class diagram.

### Preparation

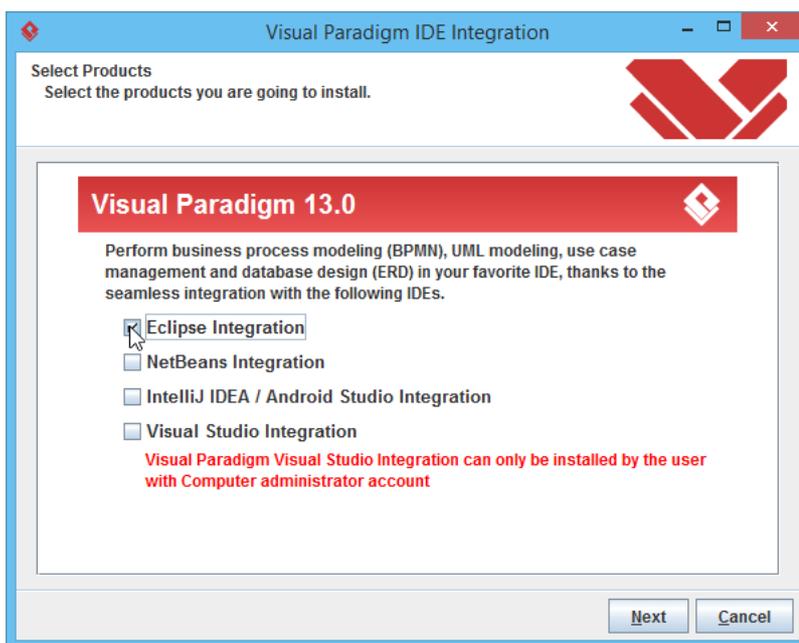
In order to follow this tutorial, you must have Visual Paradigm installed, which can be downloaded from [Visual Paradigm download page](#). Besides, you also need the Eclipse platform, which can be downloaded at the [Eclipse official site](#).

### Installing Visual Paradigm in Eclipse

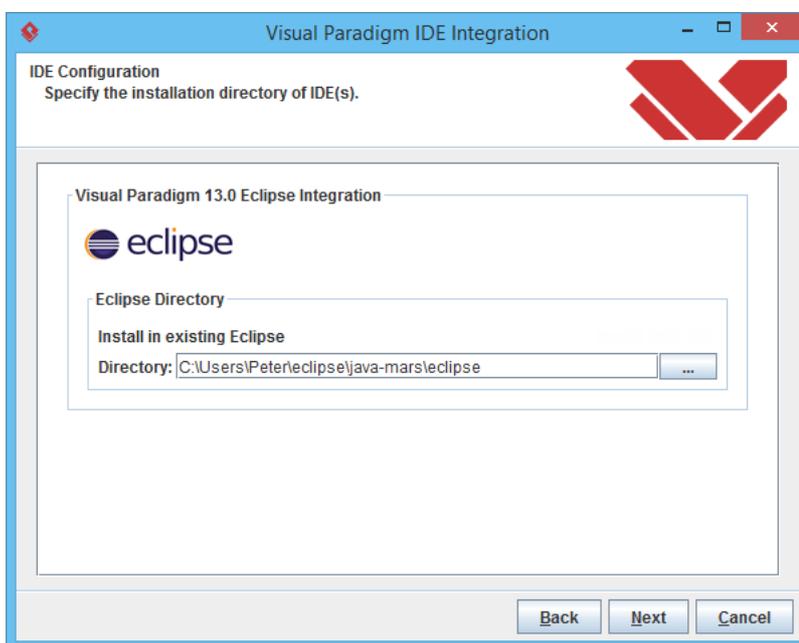
1. In Visual Paradigm, select **Window > Integration > IDE Integration...** from the application toolbar.



2. In the **Visual Paradigm IDE Integration** window, check **Eclipse Integration**.



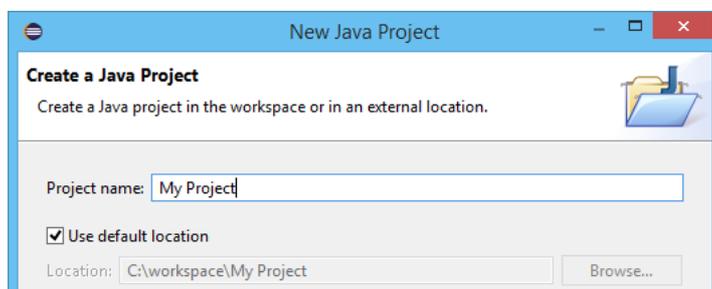
3. Click **Next**.
4. Enter the path of Eclipse and click **Next**.



This begins files copying. If you see the error messages "java.io.IOException: Cannot make dirs for file...", please restart Visual Paradigm with the **Run as Administrator** option. When finished files copying, close Visual Paradigm and move on to the next section to see how to create a Java project in Eclipse along with UML model.

## Creating a Java Project

1. Start Eclipse.
2. Select **File > New > Java Project** from the main menu to open the **New Java Project** window.
3. In the **New Java Project** window, enter *My Project* in the **Project name** field.

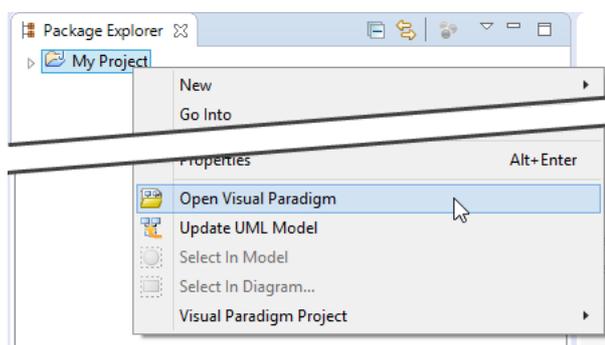


4. Click **Finish**.

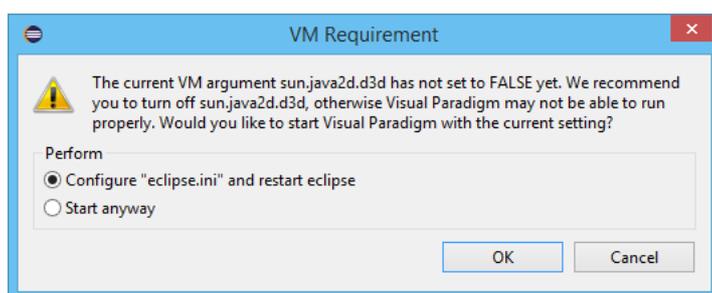
## Creating a UML Model for the Java Project

Now, you have an empty Java project. Let's create a UML model for it. To create a UML model:

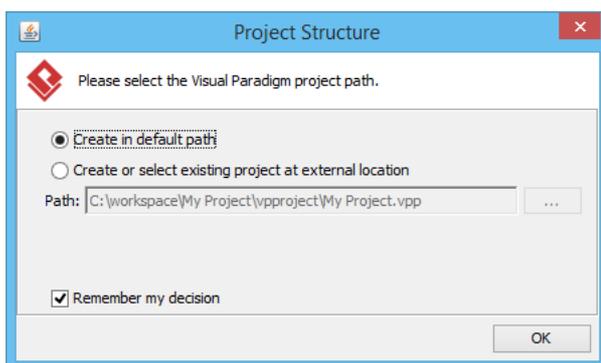
1. Right-click on the project node in **Package Explorer** and select **Open Visual Paradigm** from the popup menu.



2. If you see the **VM Requirement** dialog box, please keep the option **Configure "eclipse.ini"** and restart eclipse selected and click **OK** to restart Eclipse, and then re-perform the previous step to open Visual Paradigm.



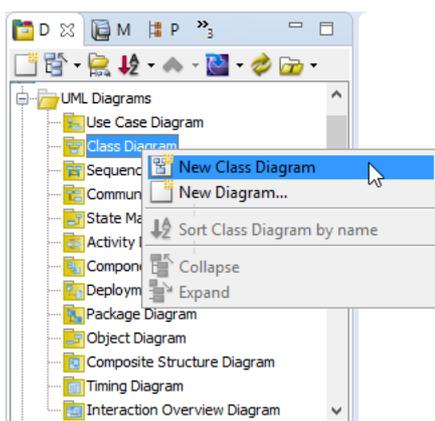
3. Click **OK** when you are prompted to select a path to store the .vpp file.



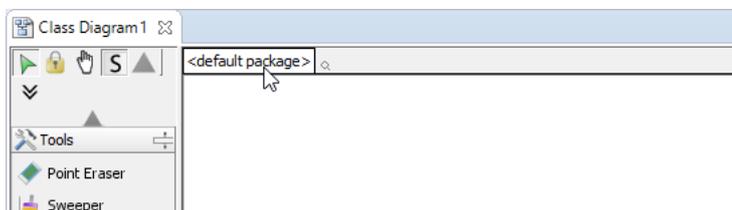
## UML Modeling in Eclipse

Let's draw a simple class diagram. We will generate Java code from it in the next section.

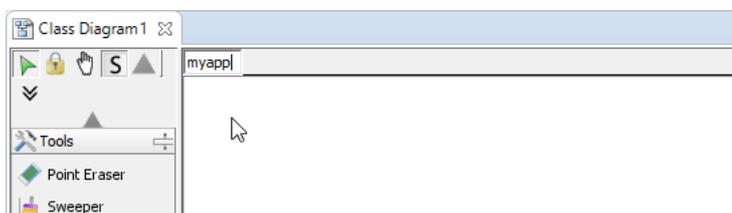
1. In **Diagram Navigator**, right-click on **Class Diagram** node and select **New Class Diagram** from the popup menu.



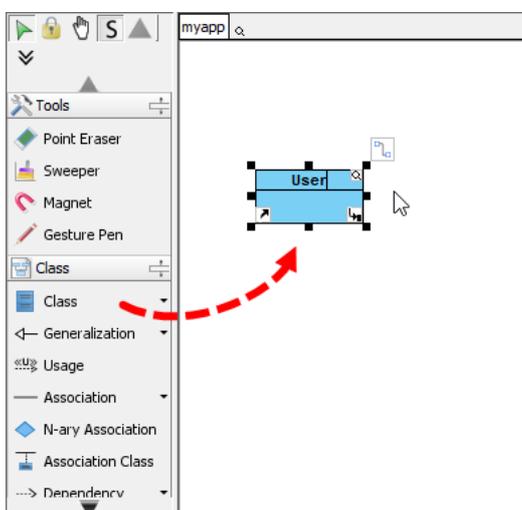
2. A new diagram is created. Double click on the package header field at the top left corner of the diagram, with a labeled **<default package>** in it.



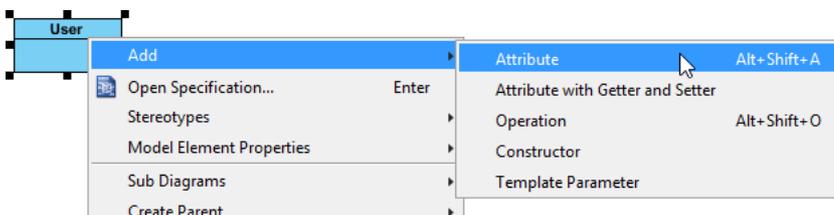
3. Enter *myapp* and press **Enter**. From now on classes to be drawn in this diagram will be placed in a (new) package named *myapp*. In practice you can also enter a nested package structure like *com.vp.myapp*.



4. Create a class. Select **Class** from the diagram toolbar. Drag it out and put it onto the diagram. Enter *User* as name and press **Enter** to confirm.

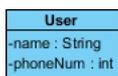


5. A user has two attributes: name and phone number. Let's add them. Right-click on the *User* class and select **Add > Attribute** from the popup menu.

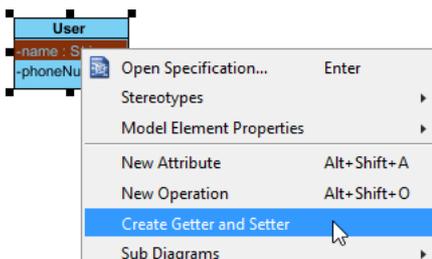


6. Enter *name : String* to create the *name* attribute in String type. Then press **Enter** to confirm.

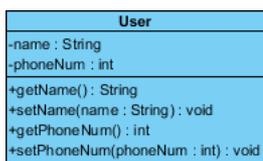
- Similarly, enter *phoneNum* : *int* to create the *phoneNum* attribute in *int* type. Then press **Enter** to confirm it and press **Esc** to cancel the next attribute.



- We want Visual Paradigm to generate getter and setter for the attributes during code generation. Right-click on the *name* attribute and select **Create Getter and Setter** from the popup menu.

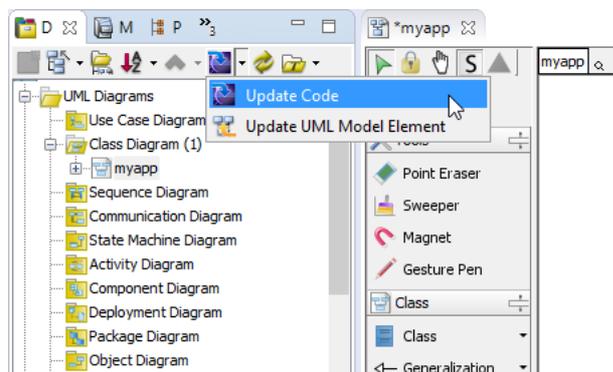


- Repeat the previous step on the attribute *phoneNum*. Up to now, your class diagram should look like this:

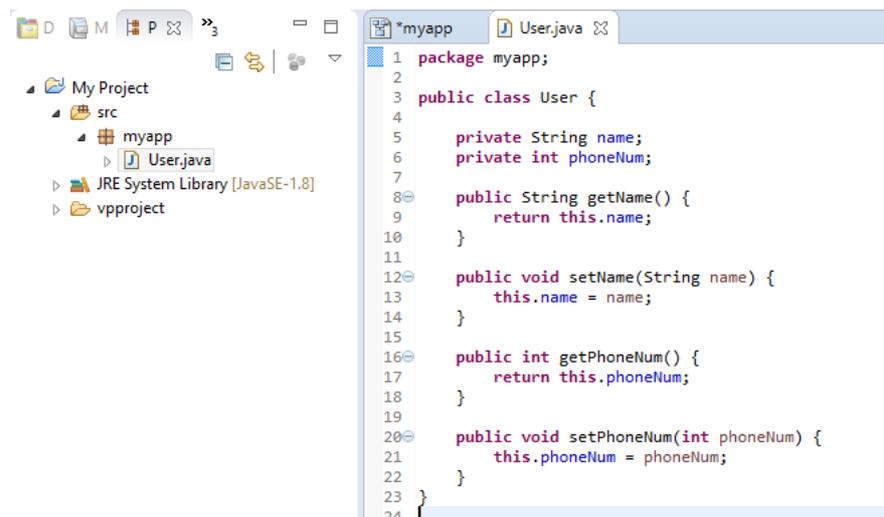


## Generate Java Code from UML Class

Let's produce Java source code from the UML class. There are several ways to achieve this. Here let's try the one that generate code for the entire UML model. Click on the **Update Code** button at the top of **Diagram Navigator**.



In the **Package Explorer**, expand the project node and the *src* folder node. The package *myapp* and *User* class are there. Open *User.java*. You can see the *User* class filled with attributes and its getter and setter.

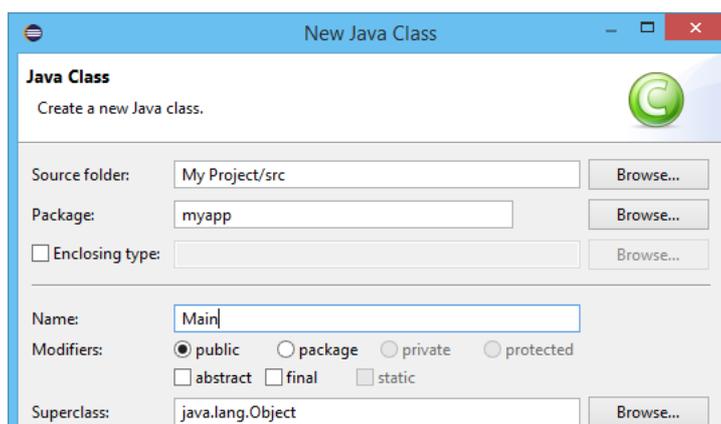


```
1 package myapp;
2
3 public class User {
4     private String name;
5     private int phoneNum;
6
7     public String getName() {
8         return this.name;
9     }
10
11     public void setName(String name) {
12         this.name = name;
13     }
14
15     public int getPhoneNum() {
16         return this.phoneNum;
17     }
18
19     public void setPhoneNum(int phoneNum) {
20         this.phoneNum = phoneNum;
21     }
22 }
23
24
```

## Perform Coding

In this section, you are going to build an executable application with the *User* class.

1. Create a *Main* class. In the **Package Explorer**, right-click on the package *myapp* and select **New > Class** from the popup menu.
2. In the **New Java Class** window, enter *Main* as class name and click **Finish**.



3. Create a static main method in the *Main* class that create and instantiate two *User* objects. Set the name and phone number through the setter.

```
*myapp  User.java  Main.java
1 package myapp;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         User u1 = new User();
7         u1.setName("Peter");
8         u1.setPhoneNum(100000);
9
10        User u2 = new User();
11        u2.setName("Mary");
12        u2.setPhoneNum(200000);
13    }
14
15 }
16
```

4. It would be nice to add into the *User* class a method to print out its name and phone number. Add a public method *printInfo()* in the *User* class.

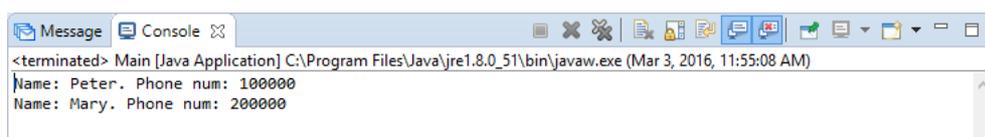
```
*myapp  User.java  Main.java
1 package myapp;
2
3 public class User {
4
5     private String name;
6     private int phoneNum;
7
8     public String getName() {
9         return this.name;
10    }
11
12    public void setName(String name) {
13        this.name = name;
14    }
15
16    public int getPhoneNum() {
17        return this.phoneNum;
18    }
19
20    public void setPhoneNum(int phoneNum) {
21        this.phoneNum = phoneNum;
22    }
23
24    public void printInfo(){
25        System.out.println("Name: " + name + ". Phone num: " + phoneNum);
26    }
27 }
28
```

5. Call `printInfo()` from the `Main` class to display the information of created users.

```
1 package myapp;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         User u1 = new User();
7         u1.setName("Peter");
8         u1.setPhoneNum(100000);
9
10        User u2 = new User();
11        u2.setName("Mary");
12        u2.setPhoneNum(200000);
13
14        u1.printInfo();
15        u2.printInfo();
16    }
17
18 }
19 }
```

## Running the Application

Let's run the application. Keep the `Main` class opened. Select **Run > Run As > Java Application** from the main menu. You should see the **Console** view appear with users' information printed in it.



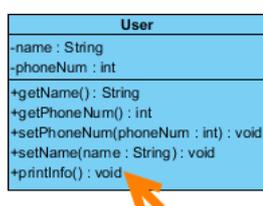
## Updating the UML Model from Java Code

You have made some changes in source code, such as the creation of `Main` class and `printInfo()` in `User` class. In order to keep the design consistent with your source code, you need to update the UML model from code. Let's do it now.

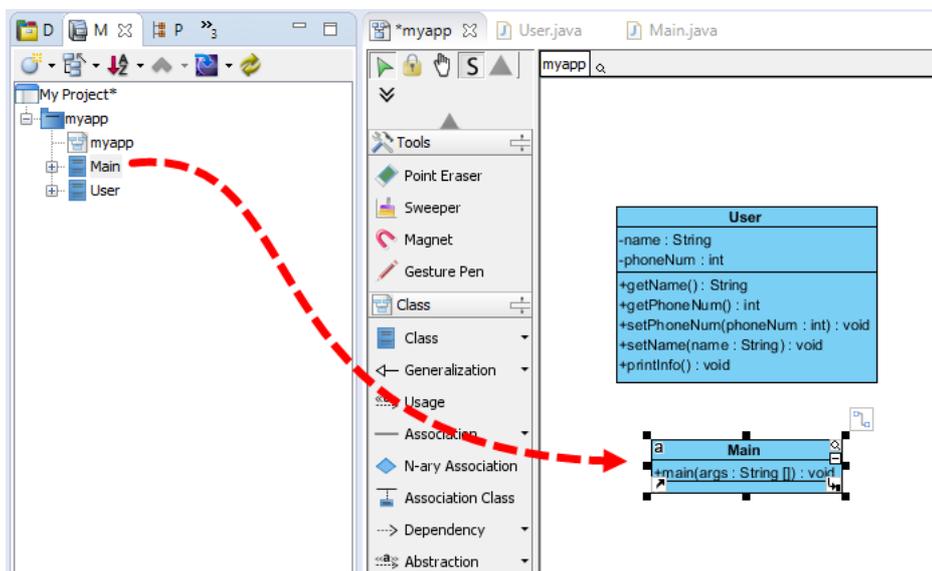
1. In the Eclipse toolbar, click on the **Update UML Model** button.



2. Open the class diagram. The `printInfo()` method is presented in the `User` class.



3. For the *Main* class, you can find it under the **Model Explorer**. Drag it out and put it below the *User* class.



#### Related Links

- [Tutorial - Hibernate in Eclipse](#)
- [Tutorial - C# Round-trip engineering](#)
- [Tutorial - Control Shape Appearance with Stereotype](#)
- [Full set of UML tools and UML diagrams](#)



Visual Paradigm home page  
(<https://www.visual-paradigm.com/>)

Visual Paradigm tutorials  
(<https://www.visual-paradigm.com/tutorials/>)