NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES

Object Oriented Analysis & Design (CS-309)

Hamza Ahmed || Muhammad Nadeem

Hamza.ahmed@nu.edu.pk || Muhammad.Nadeem@nu.edu.pk

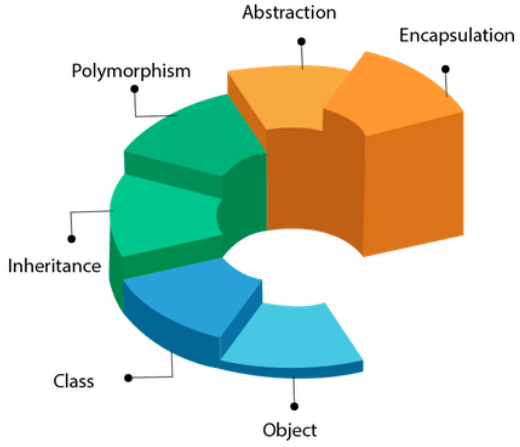
Lab Session # 01

Objectives:

1. **Revision of OOP concepts**
2. **How to interact with papyrus**
3. **Java profile and library**
    a. **Java profile**
    b. **Java library**
    c. **Common code generation profile**
4. **Code generation from UML elements**
5. **Up model introduction**

Object-Oriented Programming is a methodology or paradigm to design a program using classes and objects. It simplifies the software development and maintenance by providing some concepts:

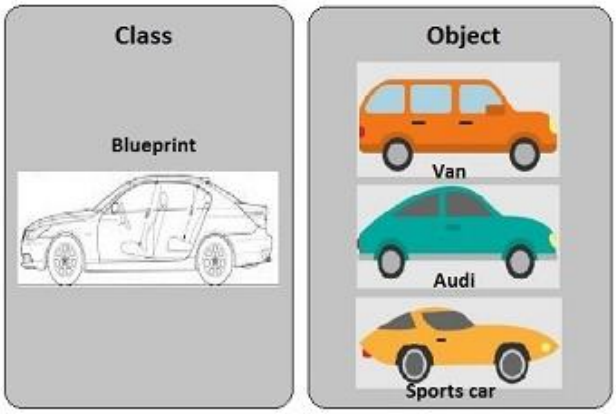
- Object
- Class
- Inheritance
- Polymorphism
- Abstraction
- Encapsulation



Object

Any entity that has state and behavior is known as an object. For example a Car, bike,Van etc. It can be physical or logical. An Object can be defined as an instance of a class. An object contains an address and takes up some space in memory.

Class



Collection of objects is called class. It is a logical entity.

A class can also be defined as a blueprint from which you can create an individual object. Class doesn't consume any space.

Example:

```
1.  class Employee{
2.     int id;
3.     String name;
4.     float salary;
5.     void insert(int i, String n, float s) {
6.        id=i;
7.        name=n;
8.        salary=s;
9.     }
10.    void display(){System.out.println(id+" "+name+" "+salary);}
11. }
12. public class TestEmployee {
13. public static void main(String[] args) {
14.    Employee e1=new Employee();
15.    Employee e2=new Employee();
16.    Employee e3=new Employee();
17.    e1.insert(101,"ajeet",45000);
18.    e2.insert(102,"irfan",25000);
19.    e3.insert(103,"nakul",55000);
20.    e1.display();
21.    e2.display();
22.    e3.display();
23. }
24. }
```

Inheritance

When one object acquires all the properties and behaviors of a parent object, it is known as inheritance. It provides code reusability. It is used to achieve runtime polymorphism.

Example:

```
1. class Employee{
2.  float salary=40000;
3. }
4. class Programmer extends Employee{
5.  int bonus=10000;
6.  public static void main(String args[]){
7.    Programmer p=new Programmer();
8.    System.out.println("Programmer salary is:"+p.salary);
9.    System.out.println("Bonus of Programmer is:"+p.bonus);
10. }
11. }
```

Polymorphism

If *one task is performed by different ways*, it is known as polymorphism. For example: to convince the customer differently, to draw something, for example, shape, triangle, rectangle, etc.

In Java, we use method overloading and method overriding to achieve polymorphism.

Another example can be to speak something; for example, a cat speaks meow, dog barks woof, etc.

Example: (overloading)

```
1. class Adder{
2. static int add(int a, int b){return a+b;}
3. static double add(double a, double b){return a+b;}
4. }
5. class TestOverloading2{
6. public static void main(String[] args){
7. System.out.println(Adder.add(11,11));
8. System.out.println(Adder.add(12.3,12.6));
9. }}
```

Example: (overriding)

```
1. class Vehicle{
2.   //defining a method
3.   void run(){System.out.println("Vehicle is running");}
4. }
5. //Creating a child class
6. class Bike extends Vehicle{
7.   //defining the same method as in the parent class
8.   void run(){System.out.println("Bike is running safely");}
9.
10.  public static void main(String args[]){
11.  Bike obj = new Bike();//creating object
12.  obj.run();//calling method
13.  }
14. }
```

Abstraction

Hiding internal details and showing functionality is known as abstraction. For example phone call, we don't know the internal processing.

In Java, we use abstract class and interface to achieve abstraction.

Example:

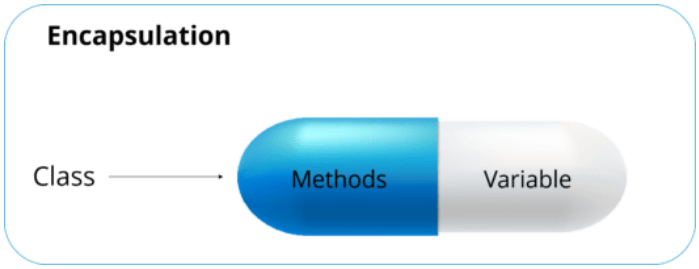
```
1. abstract class Bike{
2.   abstract void run();
3. }
4. class Honda4 extends Bike{
5. void run(){System.out.println("running safely");}
6. public static void main(String args[]){
7.  Bike obj = new Honda4();
8.  obj.run();
9. }
10.}
```

Abstraction hides complexity by giving you a more abstract picture, a sort of 10,000 feet view, while Encapsulation hides internal working so that you can change it later. In other words, Abstraction hides details at the **design** level, while Encapsulation hides details at the **implementation** level.

Read more: https://javarevisited.blogspot.com/2017/04/difference-between-abstraction-and-encapsulation-in-java-oop.html#ixzz5x70Jb71Q

Encapsulation

Binding (or wrapping) code and data together into a single unit are known as encapsulation. For example capsule, it is wrapped with different medicines.



Example:

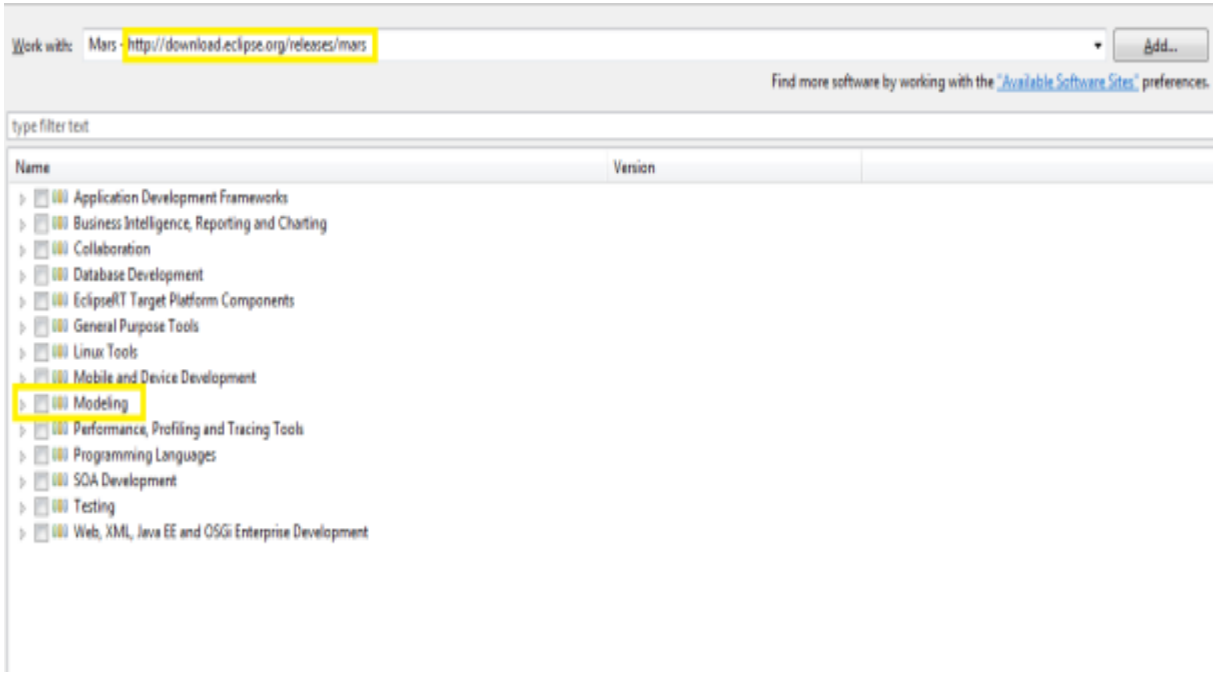
```
1. //A Java class which is a fully encapsulated class.
2. //It has a private data member and getter and setter methods.
3. public class Student{
4. //private data member
5. private String name;
6. //getter method for name
7. public String getName(){
8. return name;
9. }
10.//setter method for name
11.public void setName(String name){
```

```
12. this.name=name
13. }
1.  }
2.   //A Java class to test the encapsulated class.
3.  class Test{
4.  public static void main(String[] args){
5.  //creating instance of the encapsulated class
6.  Student s=new Student();
7.  //setting value in the name member
8.  s.setName("vijay");
9.  //getting value of the name member
10. System.out.println(s.getName());
11. }
12. }
```

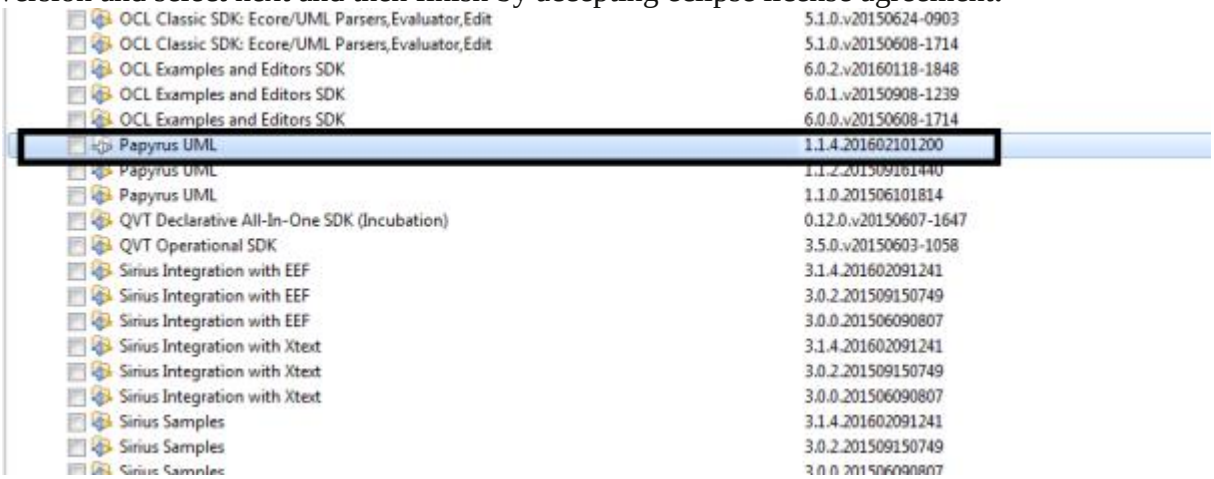
How to install Papyrus and generate java code in Eclipse

Here, following steps you can download papyrus in existing eclipse and generate Java code using Class diagram.

1. Install Papyrus plugin
   Goto **Help** –> **Install New Software**
   And in Work With enter URL : http://download.eclipse.org/releases/mars
   http://www.eclipse.org/oxygen/



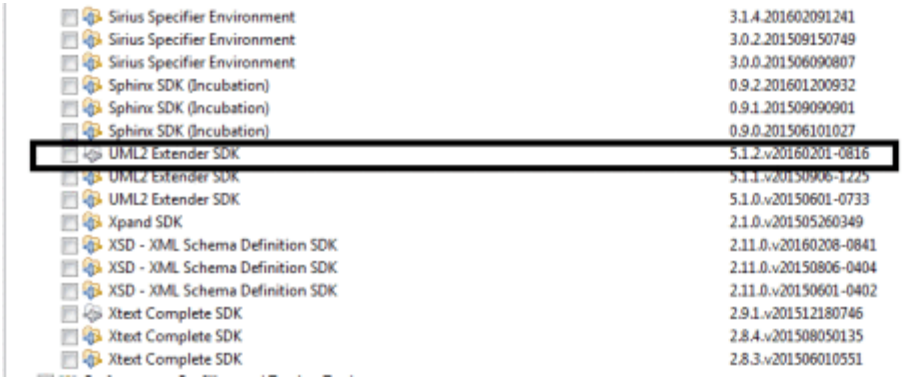
2. As shown in above image expand modeling node and look for **Papyrus UML.** Choose the latest version and select next and then finish by accepting eclipse license agreement**.**



Papyrus UML Plugin

After installation it will ask for restart eclipse. Do restart.

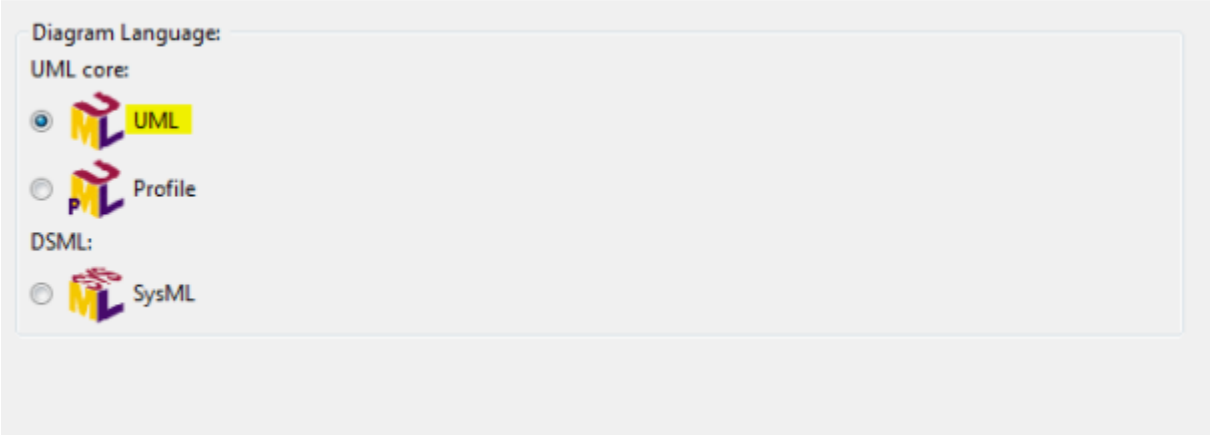
3. Install UML 2 for java code generation using papyrus.
   UML 2 is not part of Papyrus plugin. So you need to install it separately as following.
   Goto **Help –> Install New Software. Use** http://download.eclipse.org/releases/marsand in modeling node search for UML 2.



Uml 2 Plugin

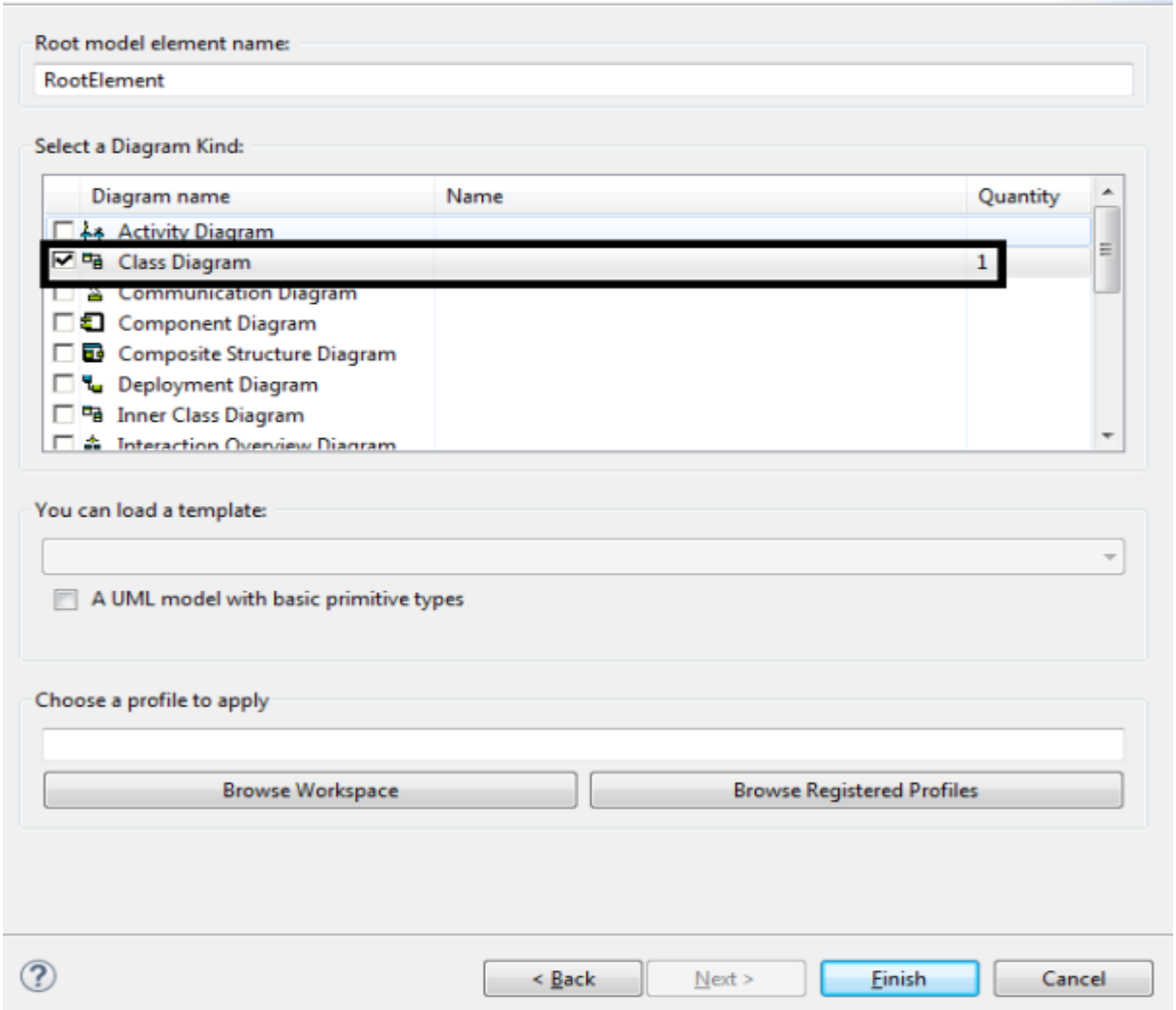
Choose the latest version and select next and then finish by accepting eclipse license agreement**.** That's it. We are done with installation.

4. Exercise : Java Code generation from papyrus.
   Create one empty java project.
   Right click on project and select **New –> Other.** Then select **Papyrus Model.** Select **Next.** In diagram Language Select **UML.**

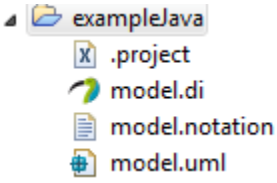


Select UML Diagram

Then Select Next and select the Project name which is created as said in step 4.
Then Select Next and select Class Diagram as below

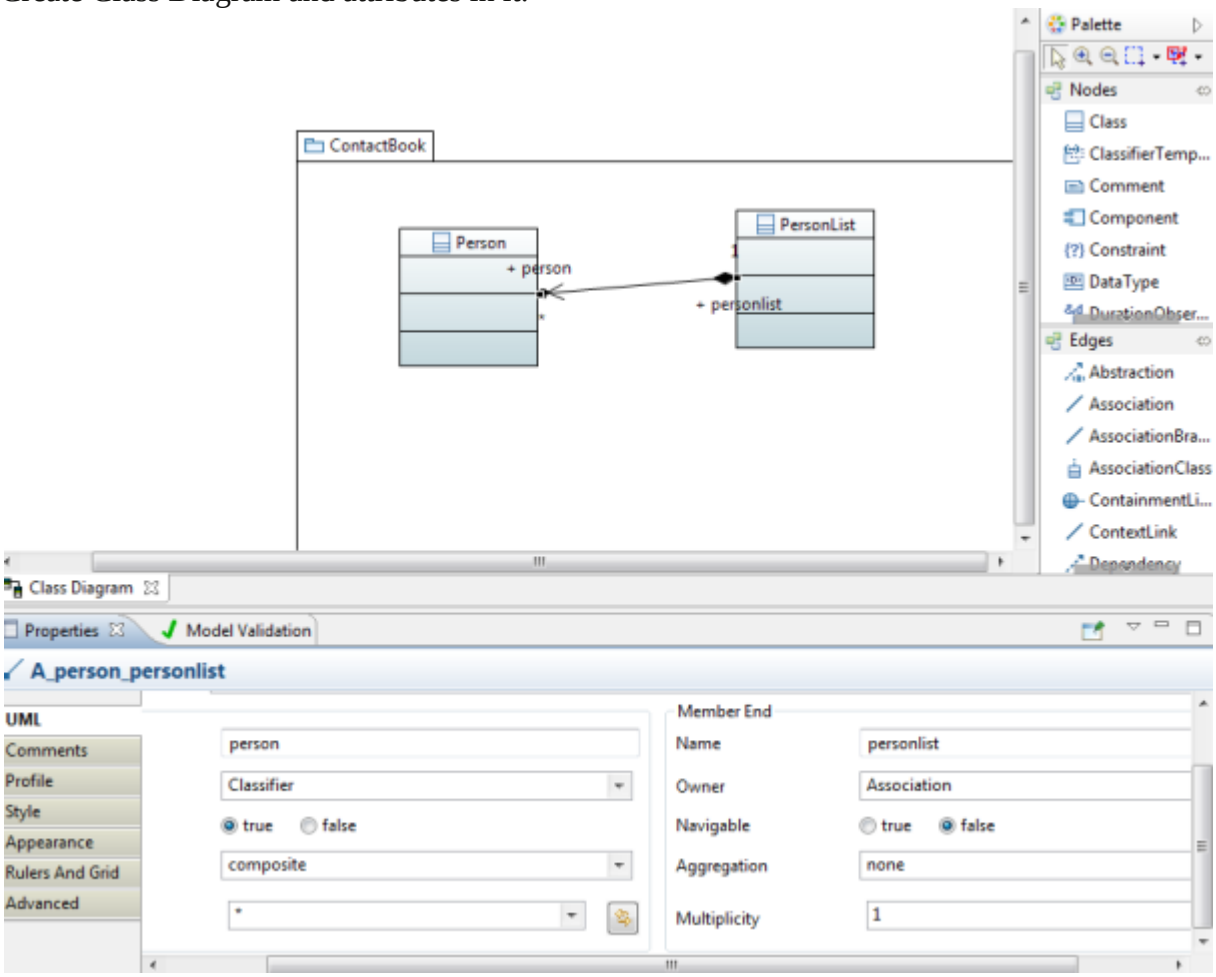


Then Finish. it will generate all necessary files under your project.

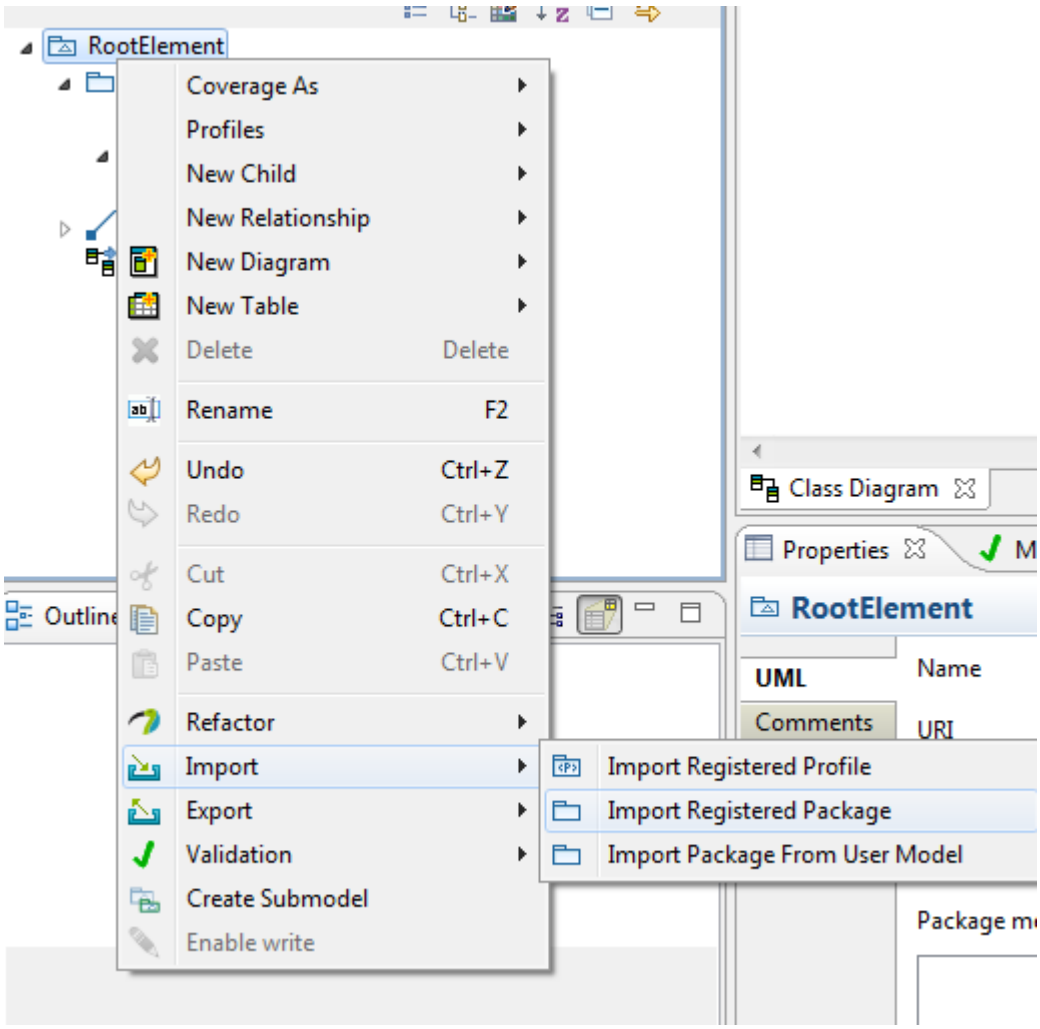


Example project

5. Open model.di file and change your perspective to **Papyrus** Perspective.
6. Create Class Diagram and attributes in it.

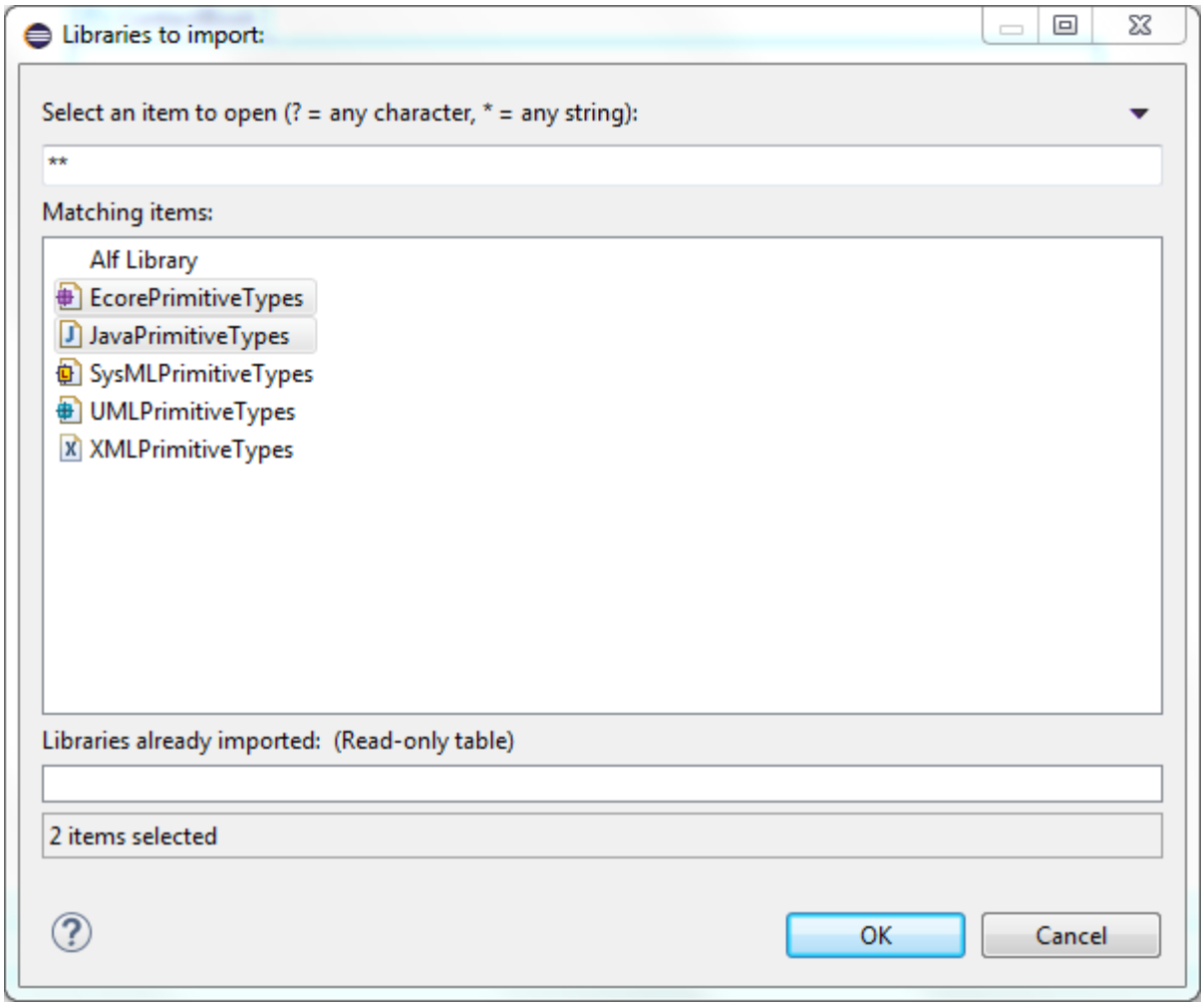


Now, Import primitive types of J**ava** and **Emf** to use it for properties in class diagram as below.



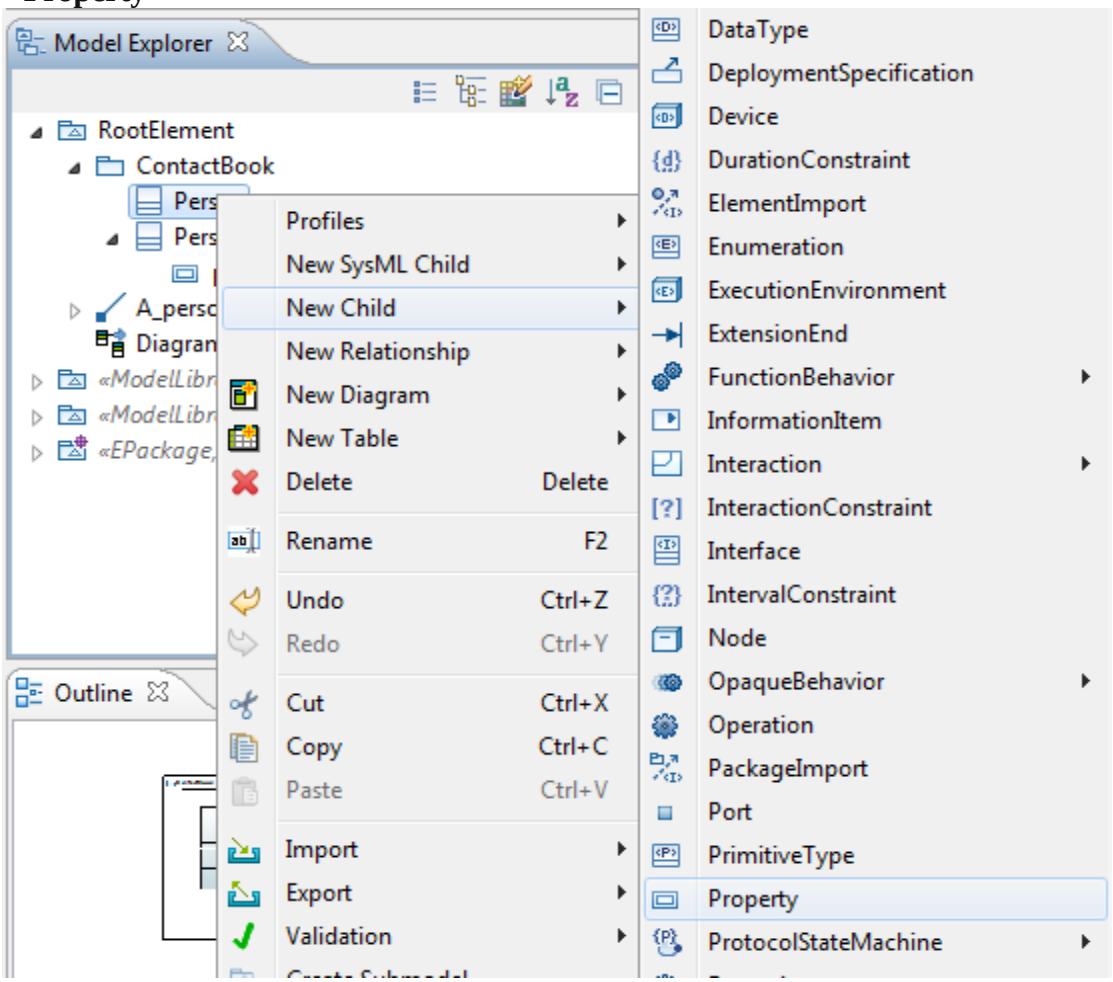
Import Registered Package

Select Ecore and Java Premitive Types as shown as below and select ok.



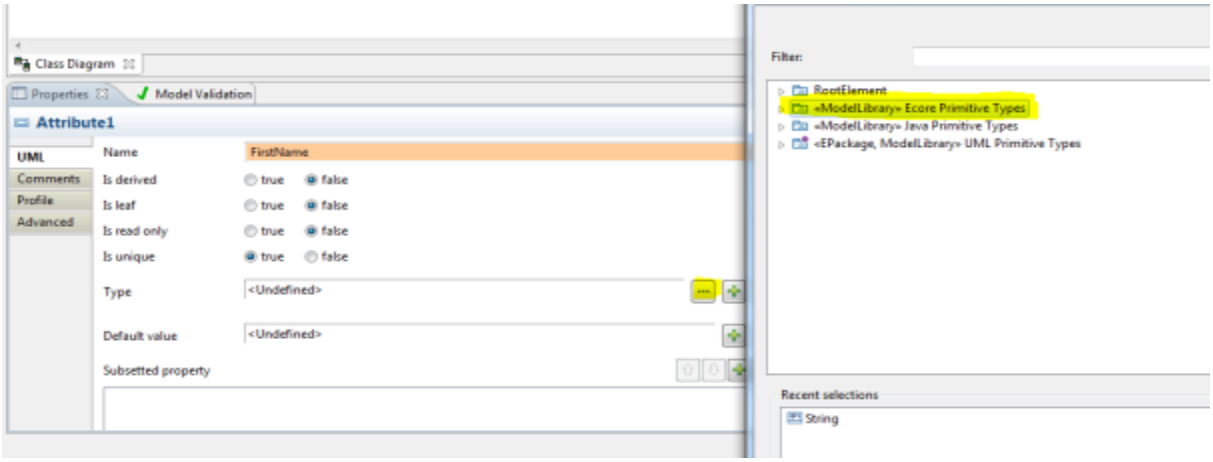
Ecore And **Java** Premitive types

7. Go to Model Explorer and add right click on Person Class and add select **New child** –> **Property**



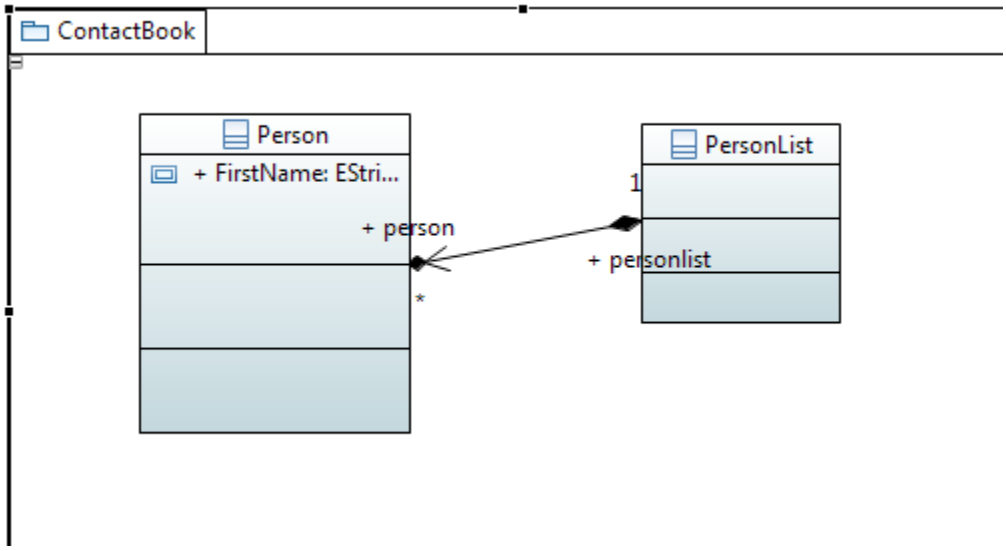
8. Add property in class

   Give Name and select Type String as shown as below in Ecore Premitive type.

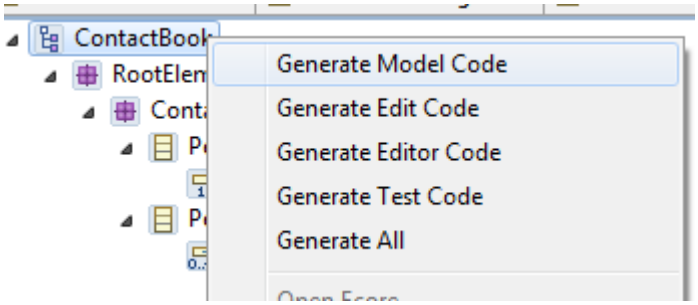


Assign Type to attribute

Drag and drop that property from Model Explorer to class in editor.

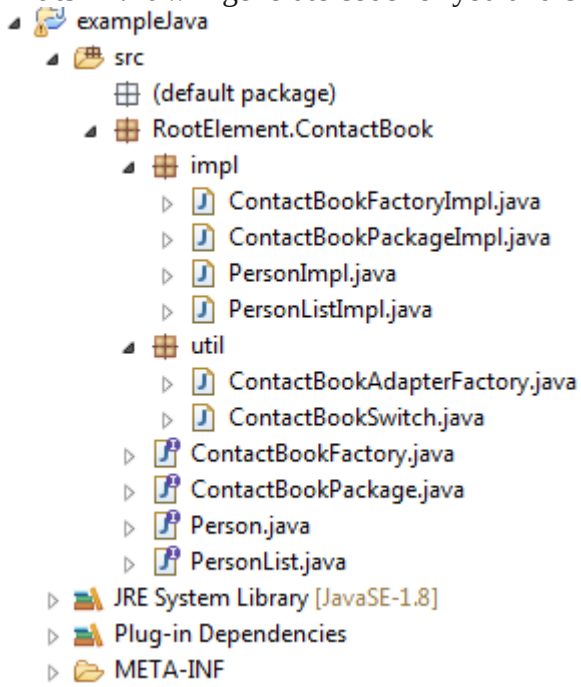


9. Switch to Plugin development perspective and in project create new **gen model** as following. Select project and open create new Wizard(**Ctrl + N**). Select **EMF Generator Model.** Give File name **ContactBook.genmodel.** Then select **UML model.** (Note if you don't find UML Model then UML 2 plugin is not installed properly).
   Select Model URI by browsing workspace and select model.uml file in you project. Select Next and then **Finish**.
10. Generate Code for Contact Book as below.
    Open **ContactBook.genmodel** and right click on Contact book and Select **Generate Model Code.**



Generate Model Code

11. **Thats IT!** It will generate code for you and convert your project in to plugin too.



Java Code Generation

The Java code generation is available in the extra plugins of Papyrus. It can be used by installing the latest version of Papyrus. It allows to create Java source code from a class diagram.

Installation instructions

The Java features of Papyrus are available as part of the **Papyrus Software Designer** extra feature.

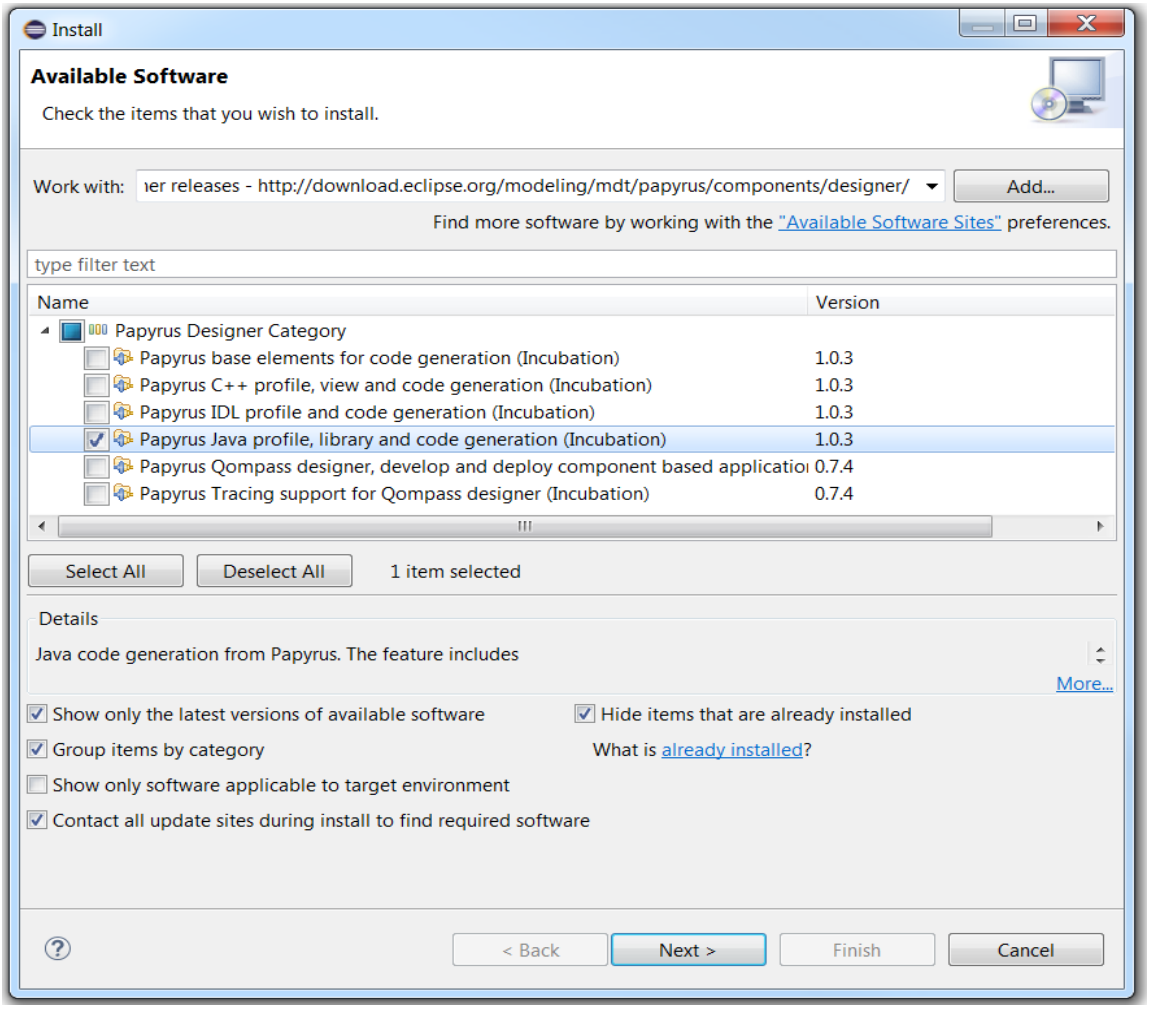
You can install it by using the Papyrus Software Designer update-site:

- **http://download.eclipse.org/modeling/mdt/papyrus/components/designer/**

Or you can use the nightly build update site (paste following url in eclipse update site):

- **https://hudson.eclipse.org/papyrus/view/Designer/job/papyrus-designer-neon-papyrusnightly/lastSuccessfulBuild/artifact/releng/org.eclipse.papyrus.designer.p2/target/repository/**

Then select "Papyrus Java profile, library and code generation (Incubation)" feature. You need JDT installed (which comes with most Eclipse packages).

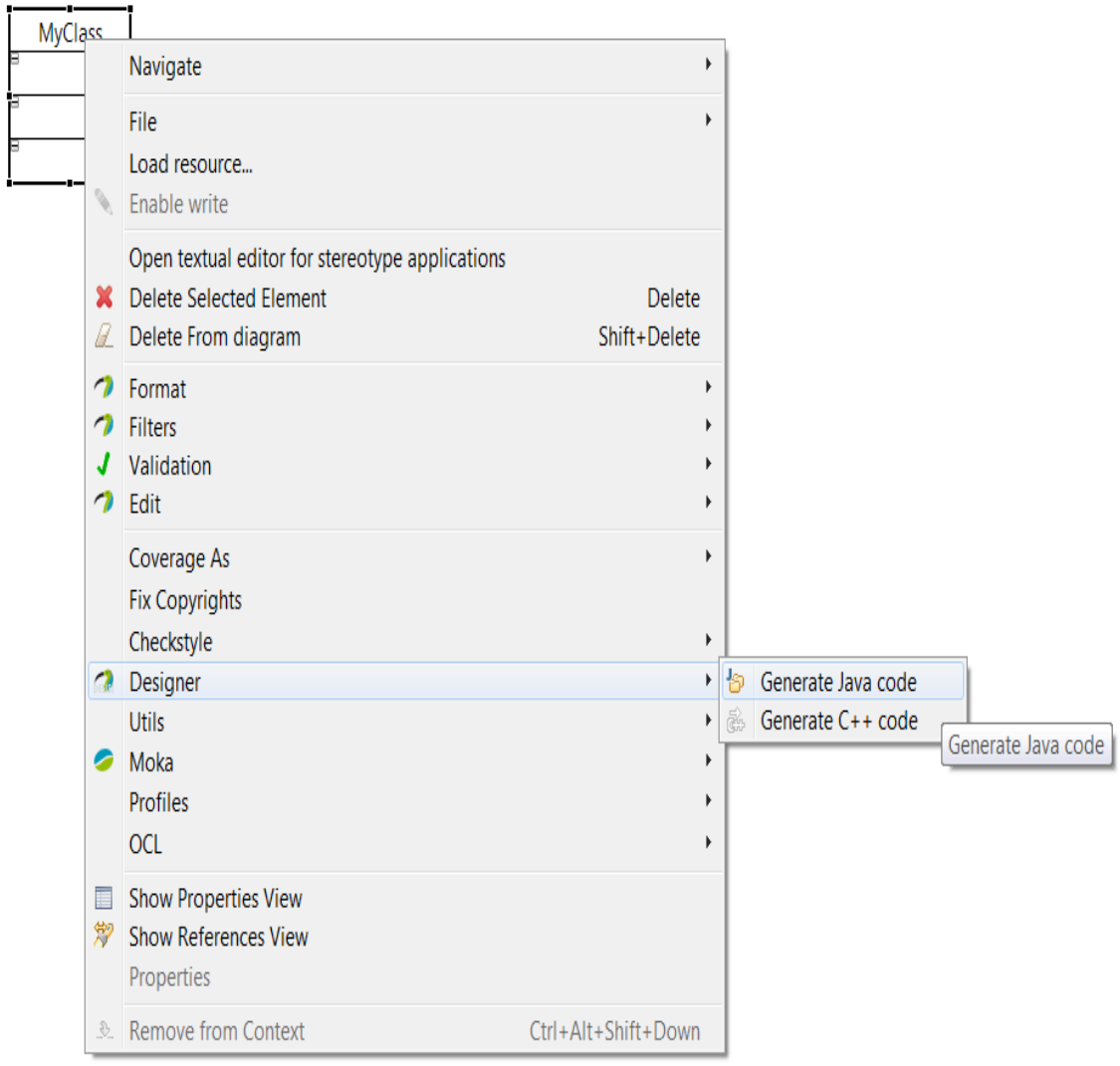


Code generation from UML elements

You can generate code for a specific classifier or package in your UML model. To generate code, do the following:

- Right click on classifier or package, either in a diagram or in the model explorer
- Designer > Generate Java code
- Follow the JDT dialogs that let you create a new JDT project in Eclipse, where code will be generated, if there is no JDT project associated with your model (e.g. the first time you generate code from your model)

When you generate code from a classifier, its required classifiers are also generated. Required classifiers are classifiers related for the generated classifier, e.g. typing one of its attributes, inheritance relationship, dependency relationship. When you generate code from a package (e.g. the root of your model), all of its classifiers, and their required classifiers, will be generated.



Generation preferences

You can change the Java code generation preferences using the Eclipse preferences menu, and choose Papyrus > Java code generation. Available options are:

- Extension for Java files
- Prefix for name of created JDT project (if unspecified in the model, see below)
- Header for generated files