NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES

CS3005 – Software Design & Architecture Lab

LAB Instructors: Sobia Iftikhar "Sobia.iftikhar@nu.edu.pk"

Lab 01

Pre-Requisites

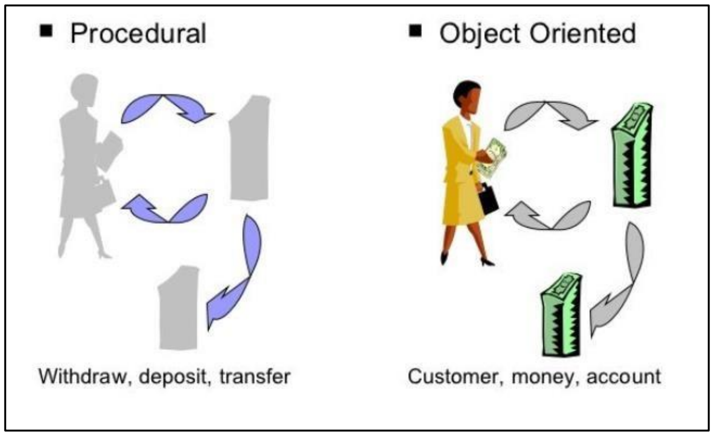
These links will be help for the proper installation of eclipse Papyrus.

1. https://www.eclipse.org/downloads/download.php?file=/modeling/mdt/papyrus/rcp/2020-06/4.8.0/papyrus-2020-06-4.8.0-win64.zip
2. https://www.java.com/en/download/
3. http://download.eclipse.org/mmt/qvto/updates/releases/3.6.0.
4. papyrus software designer (through eclipse marketplace)

Outline

- Transformation From Procedural To Object Oriented Programming
- General Concept of OOP with java
- Introduction to Software Design and Architecture
- How to interact with papyrus
    - Installation Process
    - Java Library
    - Java Profile
- Concept of UML
- Code Generation from UML to java
- Exercise

Transformation from Procedural to Object Oriented Programming

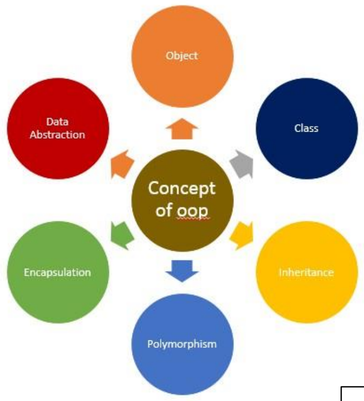


Object-Oriented Programming System

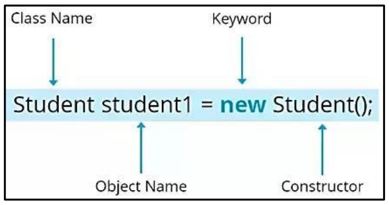
OOP is a programming concept that works on the principles of abstraction, encapsulation, inheritance, and polymorphism. It allows users to create objects they want and create methods to handle those objects. The basic concept of OOPs is to create objects, re-use them throughout the program, and manipulate these objects to get results.

The Following Are the Basic Concepts of Oop:

Object
Class
Encapsulation
Abstraction
Polymorphism
Inheritance

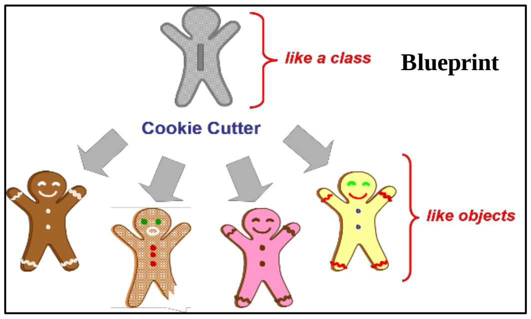


Object: An object can be defined as an instance of a class, and there can be multiple instances of a class in a program. An Object is one of the Java OOPs concepts which contains both the data and the function, which operates on the data.



Object Creation in java

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.



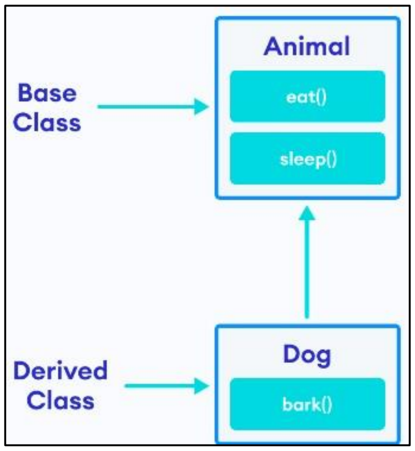
Class Concept

Program as Example

```
class shape{
int a , b;
int area;
shape(int x, int y){
a = x;
b = y;
}
void  calArea() {
area =  a * b ;
   System.out.println("area is : "+area);
}}
public class ClassExample {
public static void main(String[]
   args) {
// SDA-LAB ""
shape s1  = new shape(2, 2);
s1.calArea();
}}
```

Inheritance: The technique of deriving a new class from an old one is called inheritance

- Capability of a class to derive properties and characteristics from another class.
- The extended (or child) class contains all the features of its base (or parent) class, and may additionally have some unique features of its own.

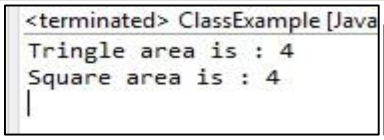


```
class Square{
int a , b;
int area;
Square(int x, int y){
a = x;
b = y;
}
void  SquareArea() {
area =  a * b ;
  System.out.println("Square area is :
  "+area);
}
}
```

```
class Tringle extends Square{
 int a , b;
 int area;
Tringle(int x, int y){
super(x, y);
 a = x;
 b = y;
 }
void  TringleArea() {
area =  a * b ;
  System.out.println("Tringle area is :
  "+area);
 }
 }
public class ClassExample {

public static void main(String[] args)
  {
// SDA-LAB
Tringle s1  = new Tringle(2, 2);
s1.TringleArea();
s1.SquareArea();
}
```

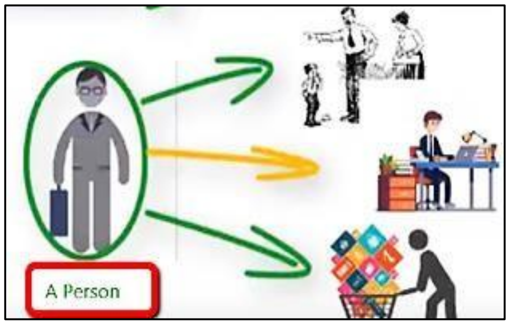
Program as Example



```
<terminated> ClassExample [Java
Tringle area is : 4
Square area is : 4
```

Output

Polymorphism: Polymorphism is a feature of OOPs that allows the object to behave differently in different conditions. We can define polymorphism as the ability of a message to be displayed in more than one form



- A real-life example of polymorphism, a person at the same time can have different characteristics. Like a man at the same time is a father, a husband, an employee.
- In Java, Polymorphism achieved by overloading and method overriding to achieve polymorphism.

Polymorphism-Overloading

```
class BaseClass{
void  Speak (char a) {
System.out.println(a);
}
void  Speak(String a) {
System.out.println(a);
}}
```

Program as Example

```
public class ClassExample {
public static void main(String[] args) {
// SDA-LAB
BaseClass s1  = new BaseClass();
s1. Speak ('I');
s1.Speak("Hey from SDA-2021"); }}
```

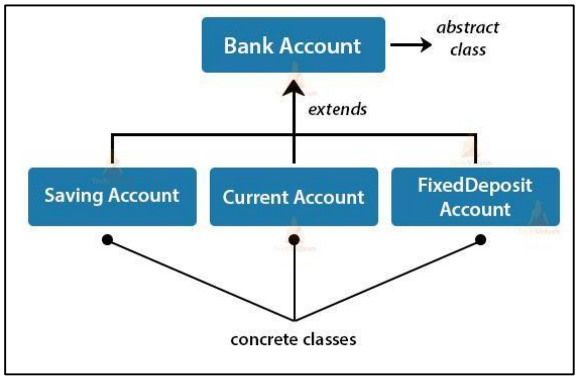
Polymorphism-Overriding

```
class BaseClass{
void  Speak() {
System.out.println("hey from Base class ");
}}
class Drived extends BaseClass{
void  Speak() {
System.out.println("hey from drive class ");
}
```

Program as Example

```
}
public class ClassExample {
public static void main(String[] args) {
// SDA-LAB
BaseClass s1  = new Drived();
s1.Speak();
}
```

Abstraction: Abstraction means displaying only essential information and hiding the details. 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.

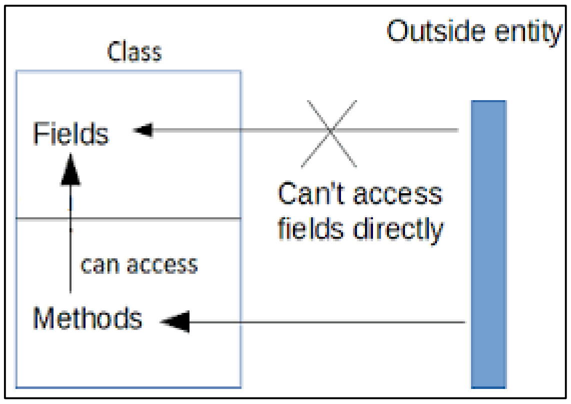


Program as Example

```
abstract class BaseClass{
abstract void  Speak();}
class Drived extends BaseClass{
void  Speak() {
System.out.println("hey  from  drive  class
  ");}}
public class ClassExample {
public static void main(String[] args) {
// SDA-LAB
BaseClass s1  = new Drived();
s1.Speak();
 }}
```

Encapsulation: Encapsulation is a process of combining data members and functions in a single unit called class.

- This is to prevent the access to the data directly, the access to them is provided through the functions of the class.
- To achieve this, you must declare class variables/attributes as private (cannot be accessed from outside the class).



```
class Person {
private int age;
public int getAge() {
return age;}
public void setAge(int age) {
this.age = age;
}}
public class ClassExample {
public static void main(String[] args) {
Person p1 = new Person();
p1.setAge(24);
System.out.println("My age is " +
  p1.getAge());}}
```

Program as Example

Introduction to Software Design and Architecture

Software architecture involves the high level structure of software system abstraction, by using decomposition and composition, with architectural style and quality attributes. A software architecture design must conform to the major functionality and performance requirements of the system, as well as satisfy the non-functional requirements such as reliability, scalability, portability, and availability.

A software architecture must describe its group of components, their connections, interactions among them and deployment configuration of all components.

A software architecture can be defined in many ways −

UML (Unified Modeling Language) − UML is one of object-oriented solutions used in software modeling and design.

Architecture View Model (4+1 view model) − Architecture view model represents the functional and non-functional requirements of software application.

ADL (Architecture Description Language) − ADL defines the software architecture formally and semantically.

Structural Diagrams: Structural diagrams represent the static aspects of a system

Class diagram
Object diagram
Component diagram
Deployment diagram
Package diagram
Composite structure

Behavioral Diagrams: Dynamic aspects are basically the changing/moving parts of a system

Use case diagram
Sequence diagram
Communication diagram
State chart diagram
Activity diagram
Time sequence diagram

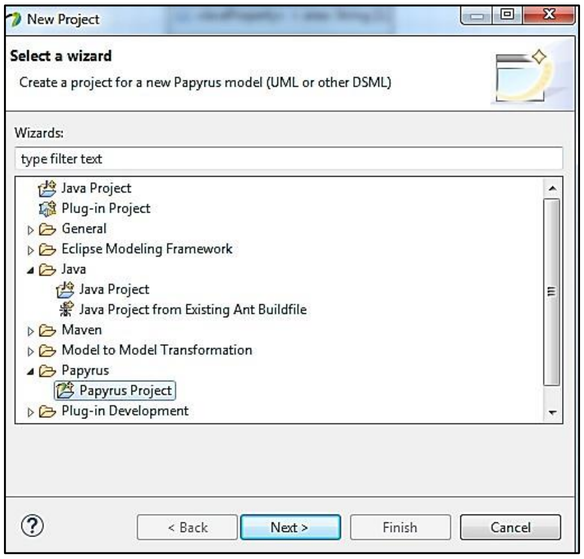
UML TO Code Generation

Class Diagram

Class name*:* *Shape*
Properties: *Area*

Steps to create UML to Java Code Generation:

1. Go to file-> new->project. Select "Papyrus Project from New Project Window.


New Project

Select a wizard

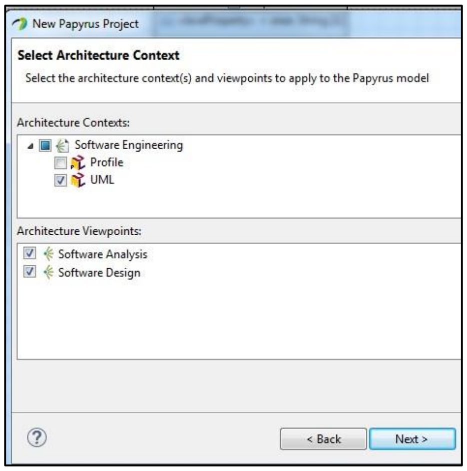
Create a project for a new Papyrus model (UML or other DSML)

Wizards:

type filter text

- Java Project
- Plug-in Project
- General
- Eclipse Modeling Framework
- Java
  - Java Project
  - Java Project from Existing Ant Buildfile
- Maven
- Model to Model Transformation
- Papyrus
  - Papyrus Project
- Plug-in Development

2. Select UML and click next


New Papyrus Project

Select Architecture Context

Select the architecture context(s) and viewpoints to apply to the Papyrus model

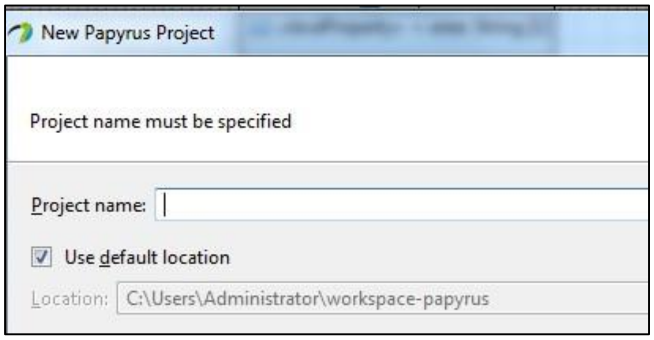
Architecture Contexts:

- [■] Software Engineering
  - [ ] Profile
  - [✓] UML

Architecture Viewpoints:

- [✓] Software Analysis
- [✓] Software Design

3. Write Project name “TestExampleUMLToCode”


New Papyrus Project

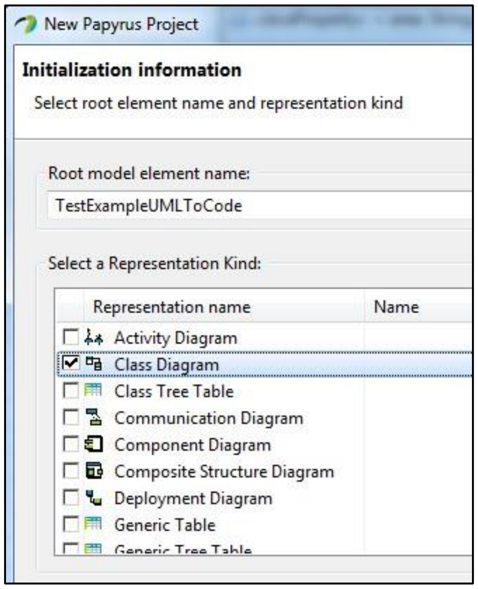
Project name must be specified

Project name:

[✓] Use default location

Location: C:\Users\Administrator\workspace-papyrus

4. Select Class Diagram and click on finish


New Papyrus Project

Initialization information

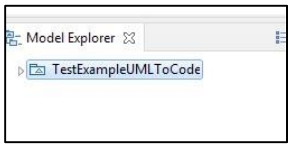
Select root element name and representation kind

Root model element name: TestExampleUMLToCode

Select a Representation Kind:

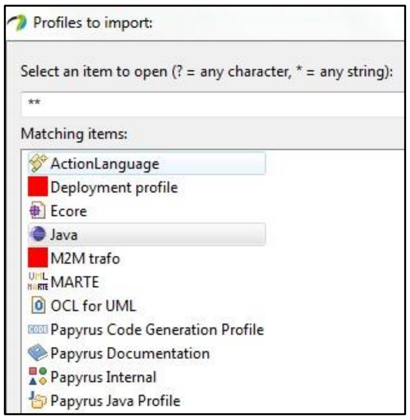
Representation name	Name
[ ] Activity Diagram	
[✓] Class Diagram	
[ ] Class Tree Table	
[ ] Communication Diagram	
[ ] Component Diagram	
[ ] Composite Structure Diagram	
[ ] Deployment Diagram	
[ ] Generic Table	
[ ] Generic Tree Table	

5. Right click on “Model Explorer Window” select import option then select “import registered Files ”


Model Explorer

TestExampleUMLToCode

6. After Import Register file “profiles to import” window will appear select java profile and click ok

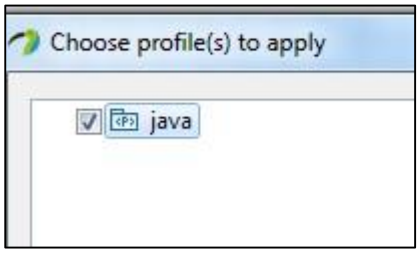

Profiles to import:

Select an item to open (? = any character, * = any string):

**

Matching items:

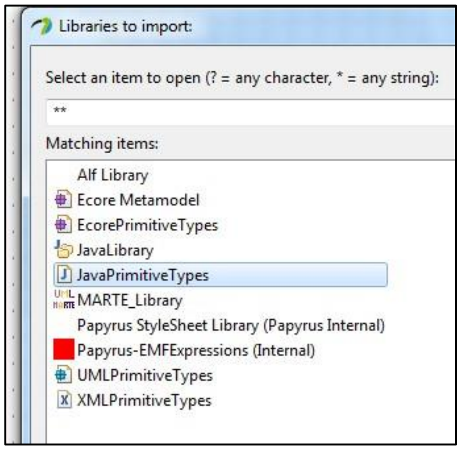
- ActionLanguage
- Deployment profile
- Ecore
- Java
- M2M trafo
- MARTE
- OCL for UML
- Papyrus Code Generation Profile
- Papyrus Documentation
- Papyrus Internal
- Papyrus Java Profile


Choose profile(s) to apply

java

7. Right click on "Model Explorer Window" select import option then select "import registered Package "

8. After Import Register package "Library to import" window will appear select "JavaPrimitvesTypes" and click ok.

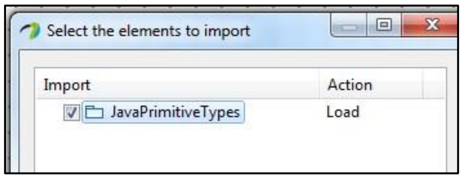

Libraries to import:

Select an item to open (? = any character, * = any string):

**

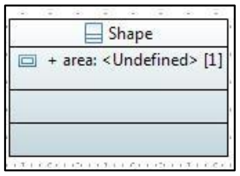
Matching items:

- Alf Library
- Ecore Metamodel
- EcorePrimitiveTypes
- JavaLibrary
- JavaPrimitiveTypes
- MARTE_Library
- Papyrus StyleSheet Library (Papyrus Internal)
- Papyrus-EMFExpressions (Internal)
- UMLPrimitiveTypes
- XMLPrimitiveTypes

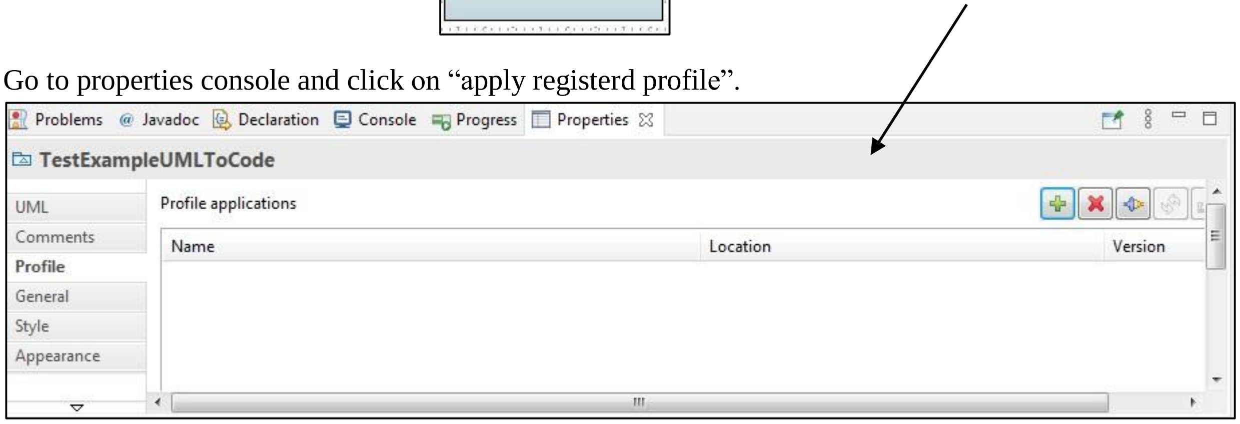

Select the elements to import

Import	Action
JavaPrimitiveTypes	Load

9. Now drag the class and properties component from "palette" window.



10. Go to properties console and click on "apply registerd profile".

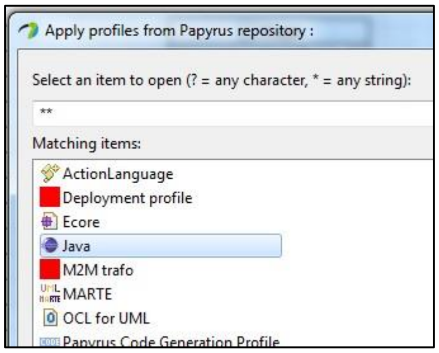


TestExampleUMLToCode

Profile applications

Name	Location	Version

11. Select java and click ok.



Apply profiles from Papyrus repository :

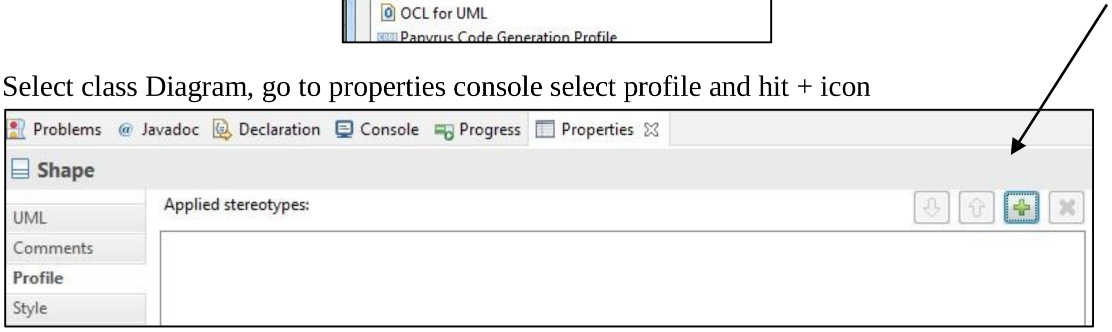
Select an item to open (? = any character, * = any string):

**

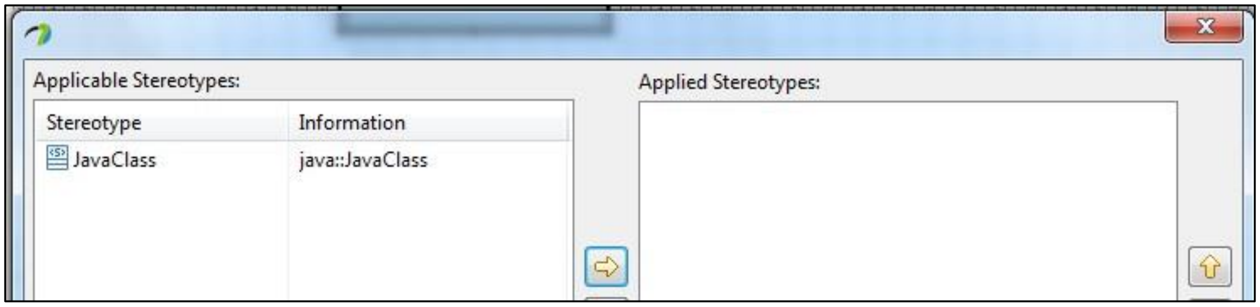
Matching items:

- ActionLanguage
- Deployment profile
- Ecore
- Java
- M2M trafo
- MARTE
- OCL for UML
- Papyrus Code Generation Profile

12. Select class Diagram, go to properties console select profile and hit + icon



13. Shuffle JavaClass to applied stereotypes.

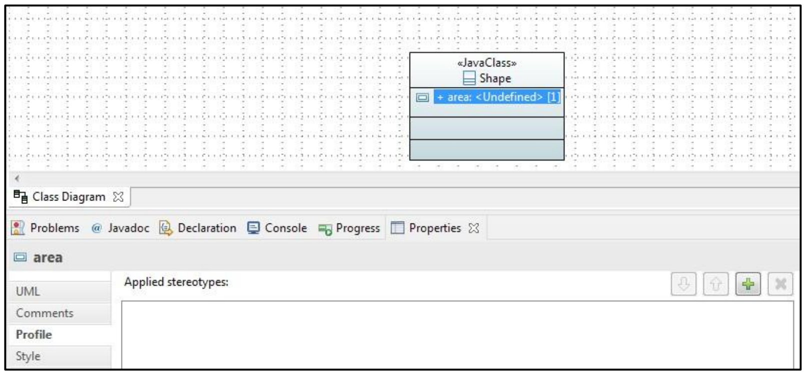


Applicable Stereotypes:

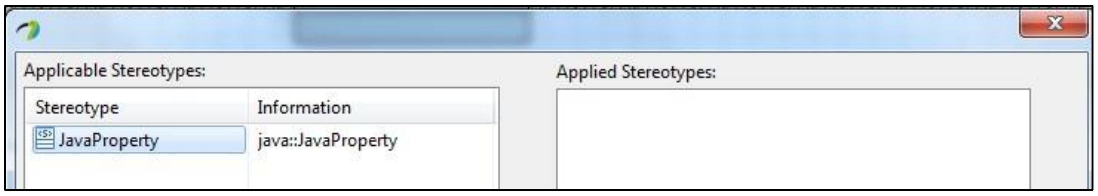
Stereotype	Information
JavaClass	java::JavaClass

Applied Stereotypes:

14. Select property “area” from class diagram, go to properties console select profile and hit + icon



15. Shuffle JavaProperty to applied stereotypes.

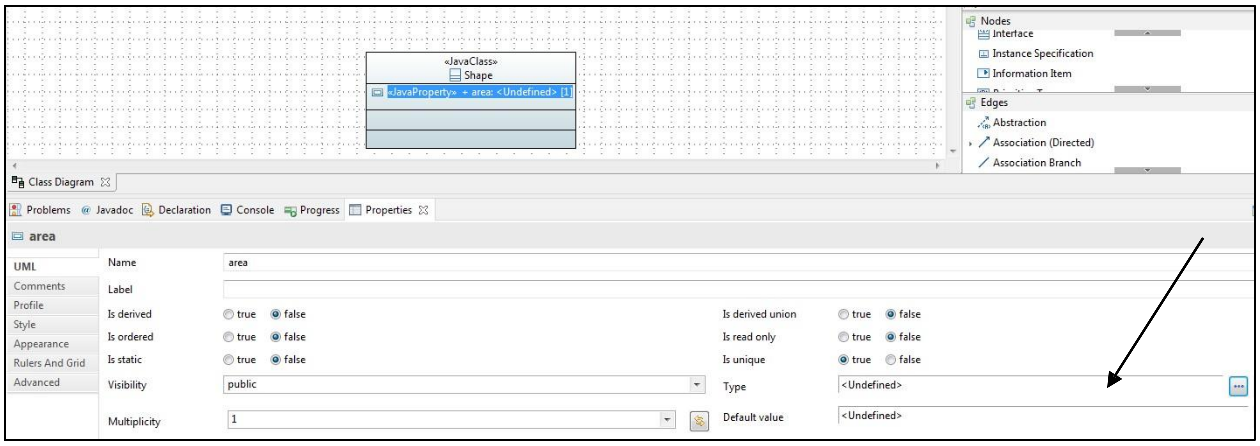


Applicable Stereotypes:

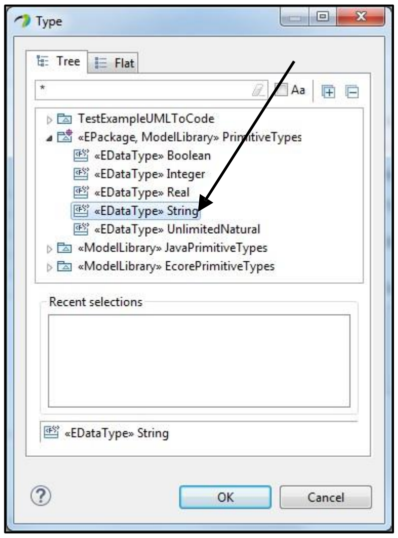
Stereotype	Information
JavaProperty	java::JavaProperty

Applied Stereotypes:

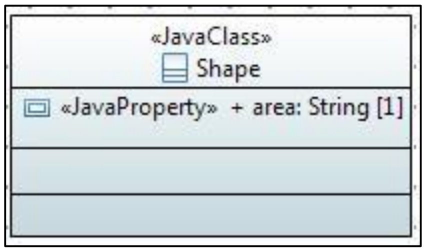
16. Select the property “area” from class diagram and change the type from undefined to string.

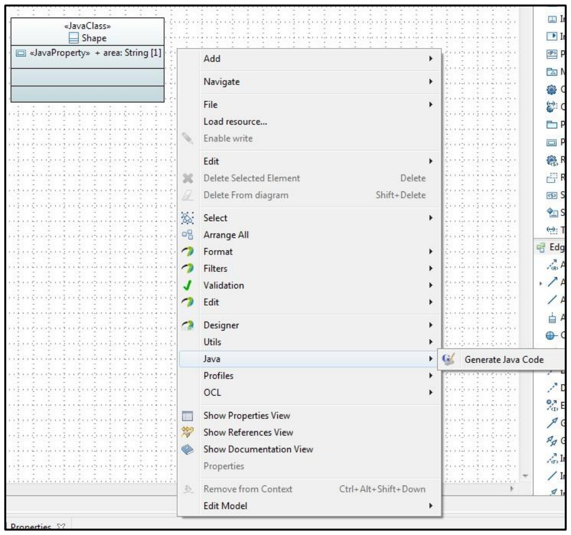


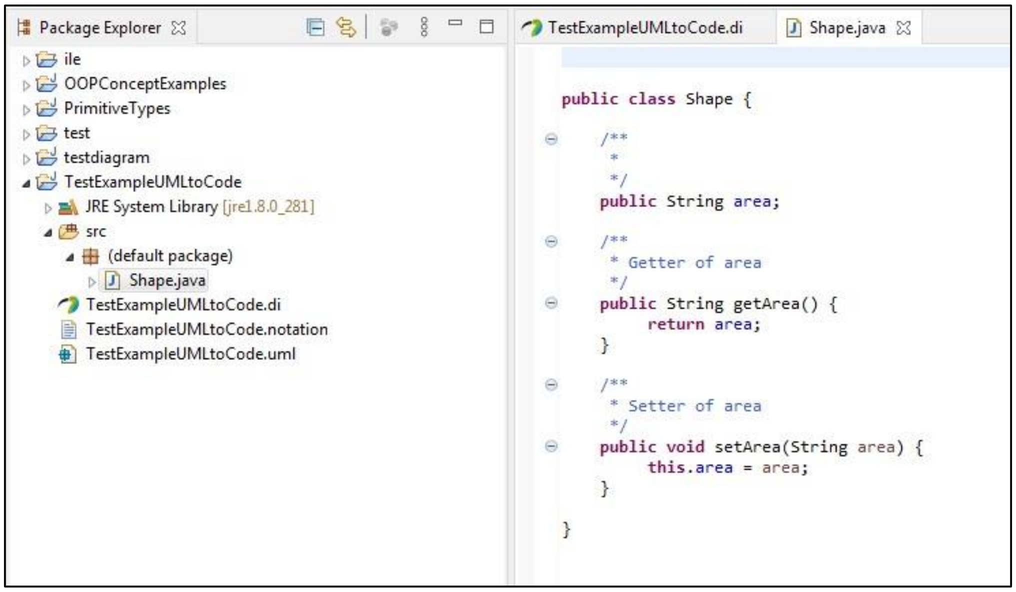
Name	area		
Label			
Is derived	true [✓ false]	Is derived union	true [✓ false]
Is ordered	true [✓ false]	Is read only	true [✓ false]
Is static	true [✓ false]	Is unique	[✓ true] false
Visibility	public	Type	<Undefined>
Multiplicity	1	Default value	<Undefined>



Final View Class Diagram. Right click on window and select java and Genrate code.







```
public class Shape {

    /**
     *
     */
    public String area;

    /**
     * Getter of area
     */
    public String getArea() {
        return area;
    }

    /**
     * Setter of area
     */
    public void setArea(String area) {
        this.area = area;
    }

}
```

Activity

1. Write a program in java to generate an class Employee, also create concrete classes name as "**Lecturer"**, "**Assistant Professor**",. inherits the class "**Employee**". Generate the object for each concrete classes that will calculate their respective salary and display the name of organization and salary on console.

   The working hours of **Assistant Professor** is 48 per month and his salary is 1000 per hour.

   The working hours of **Lecturer** is 48 per semester and his salary is 800 per hour.

   Make sure salary per hour must be given through constructor of their respective class and organization name.

   - Make sure Employee class will be the main class.
   - Employee class is responsible to set the name method that is coming from the constructor of concrete classes.
   - Name var send by the concrete class to "set Name" method by calling.
     Note: name method will maintain inheritance

2. Write an overloaded function "min" that takes either two or three parameters of type int and returns the minimum of them. That is you create two sub-functions: 1.int min (int,int) and int min(int, int, int).

   both return minimum
   write also the main program that call the two functions

3. Crate Base class Cake and the following table, define TWO (2) subclasses named.

OrderCake	ReadymadeCake
+CalculateQuantity()	+CalculateQuantity()
+CalculatePrice()	+CalculatePrice()

Formula: For OrderCake

weight * SingalCakePrice* Quantity// 1kg = 800price

Formula: For ReadymadeCake

weight * SingalCakePrice* Quantity// 1kg = 500price

i. Make sure quantity and weight must be given by user at run time

ii. Display the how many of cakes are order and what is the total bill.