

Lab Session # 01

Objectives:

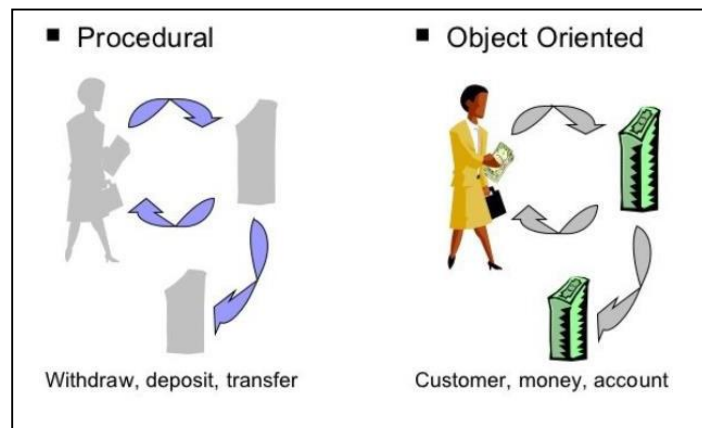
1. Transformation from Procedural To Object Oriented Programming
2. General Concept of OOP with java
3. Introduction to Software Design and Architecture
4. How to interact with papyrus
5. Installation Process
6. Java Library
7. Java Profile
8. Concept of UML
9. Code Generation from UML to java
10. Exercise

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)
- 

Transformation from Procedural to Object Oriented Programming



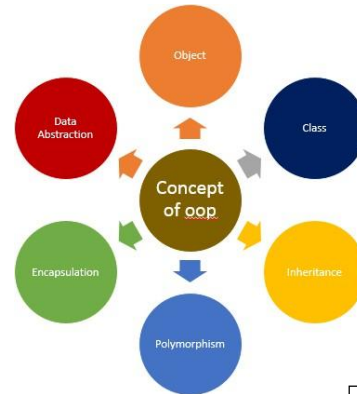
Object-Oriented Programming System

**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

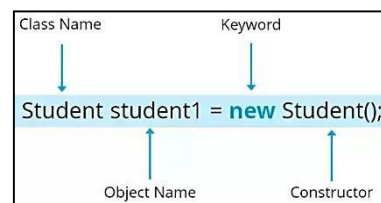
objects. The basic concept of OOP is to create objects, re-use them throughout the program, and manipulate these objects to get results.

***The Following Are the Basic Concepts of Oop:***

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



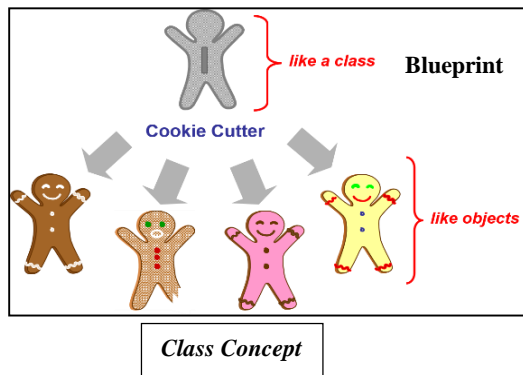
**Object:** An object can be defined as an instance of a class, and there can be multiple instances of a class in a program. An Object is one of the Java OOPs concepts which contain both the data and the function, which operates on the data.



**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.



**Program as Example**

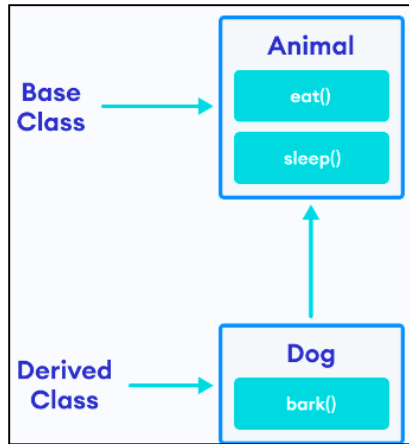
```

✓ 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);
✓ }
✓ }

```

Program as Example

```

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

```

```

<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 ('T');
✓ s1.Speak("Hey from SDA-2021"); }}

```

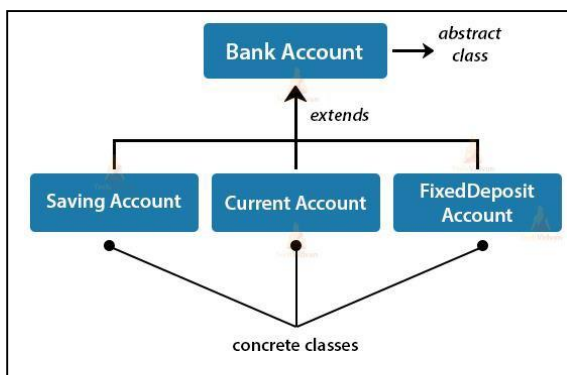
## Polymorphism-Overriding

```
✓ 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.

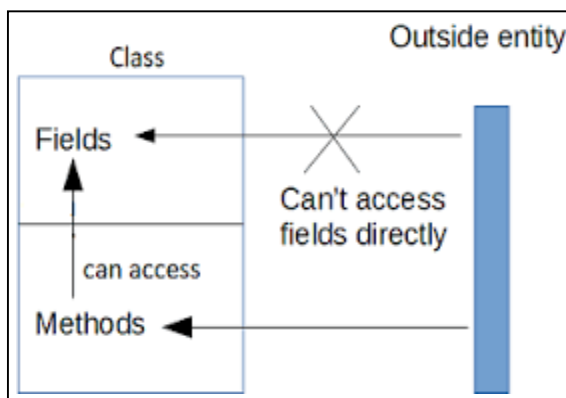


Program as Example

```
✓ 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

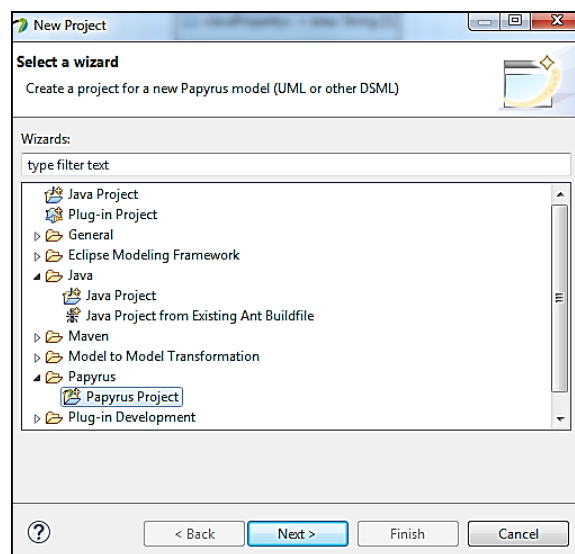
### Class Diagram

**Class name:** *Shape*

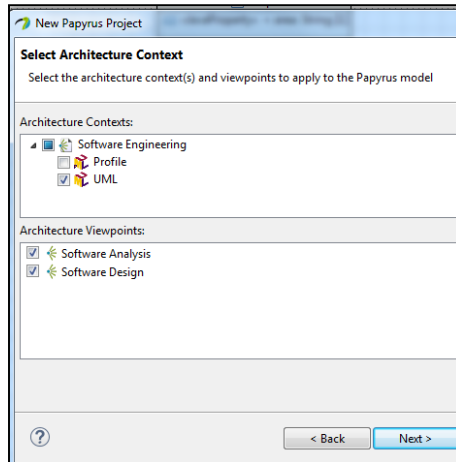
**Properties:** *Area*

### Steps to create UML to Java Code Generation:

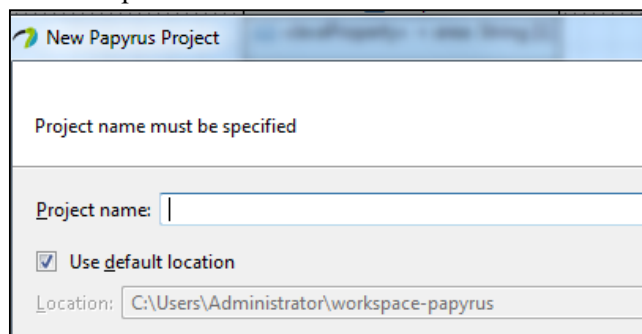
1. Go to file-> new->project. Select “Papyrus Project from New Project Window.



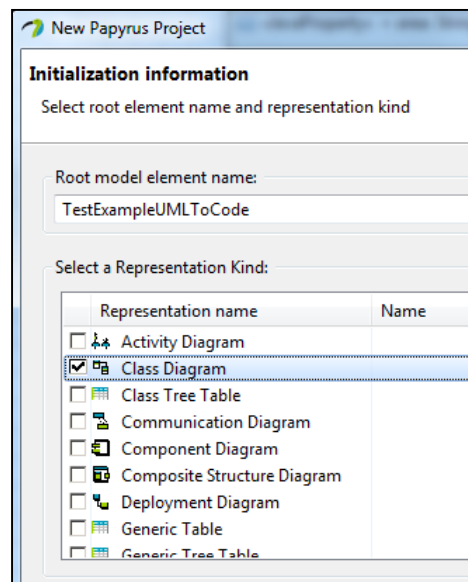
2. Select UML and click next



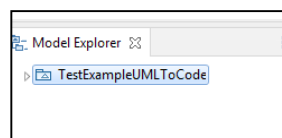
3. Write Project name “TestExampleUMLToCode”



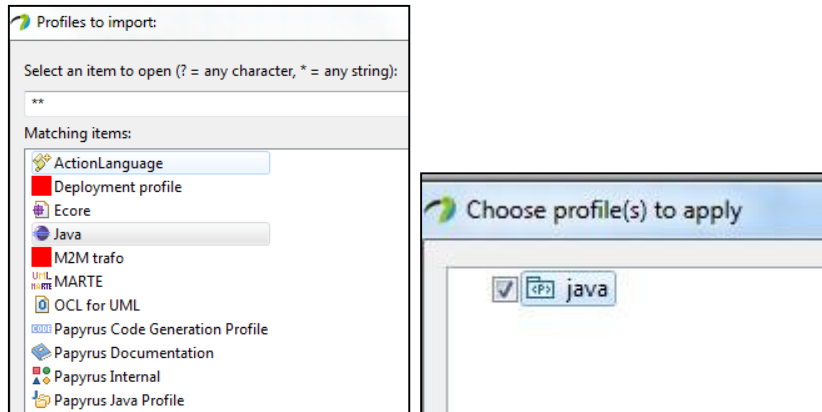
4. Select Class Diagram and click on finish



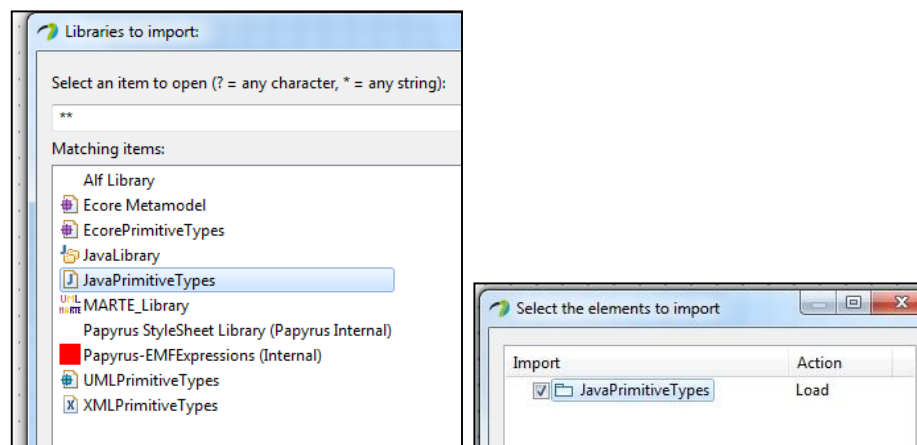
5. Right click on “Model Explorer Window” select import option then select “import registered Files”



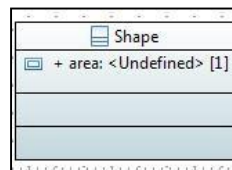
6. After Import Register file “profiles to import” window will appear select java profile and click ok



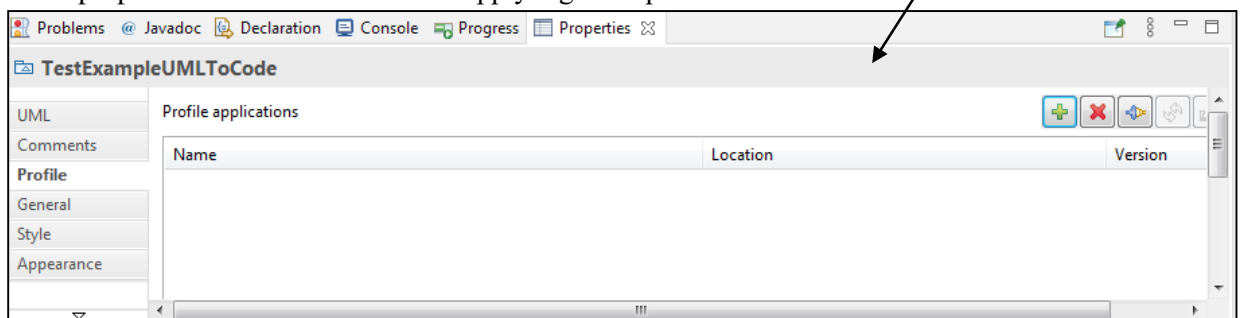
7. Right click on “Model Explorer Window” select import option then select “import registered Package ”
8. After Import Register package “Library to import” window will appear select “JavaPrimitiveTypes” and click ok.



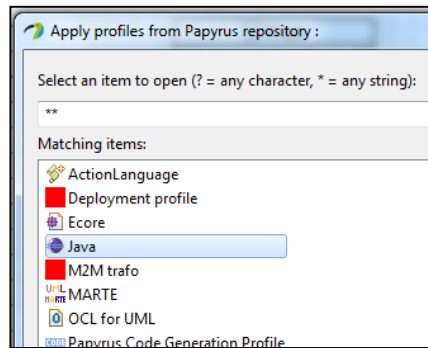
9. Now drag the class and properties component from “palette” window.



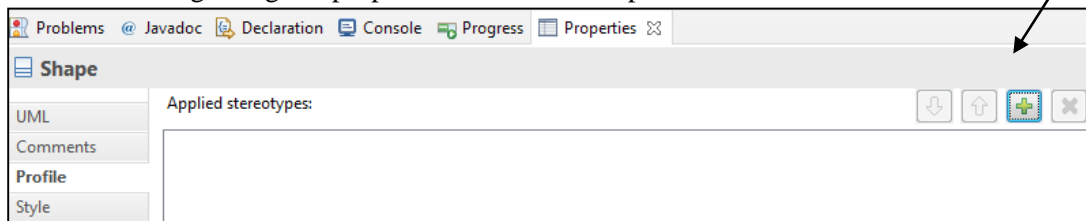
10. Go to properties console and click on “apply registered profile”.



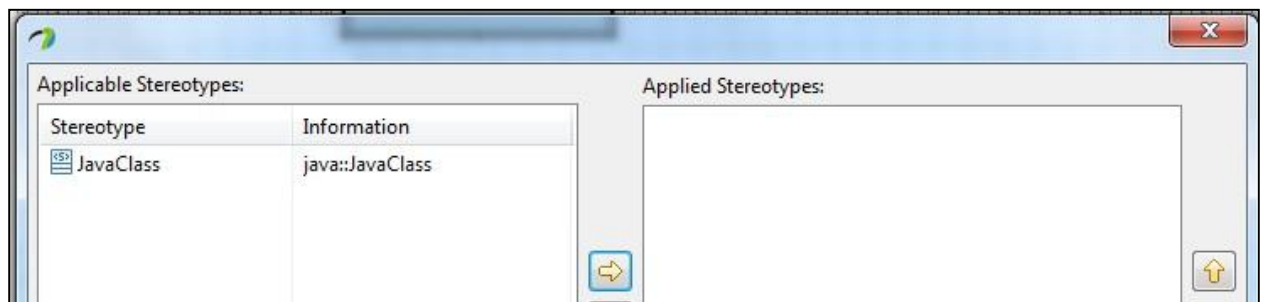
11. Select java and click ok.



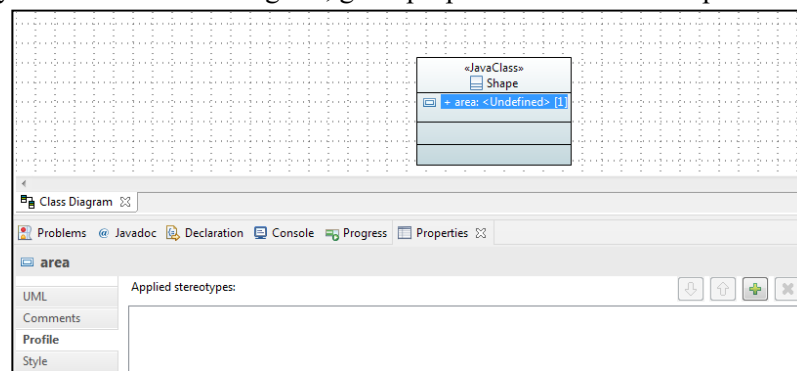
12. Select class Diagram, go to properties console select profile and hit + icon



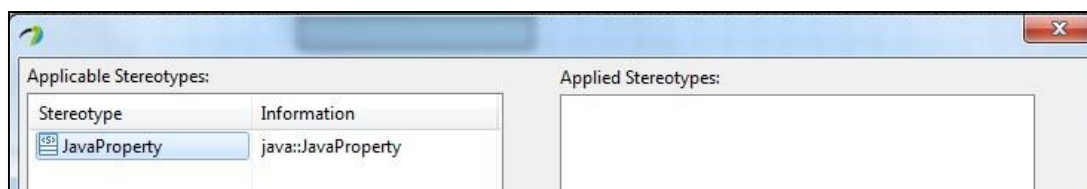
13. Shuffle JavaClass to applied stereotypes.



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

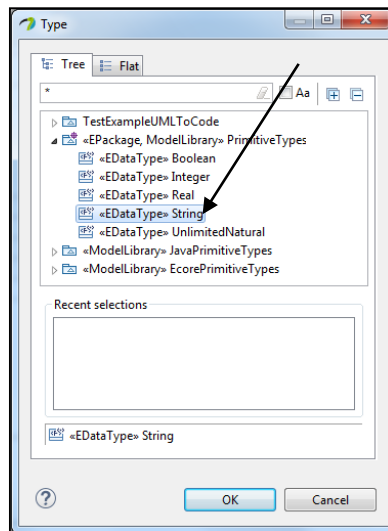
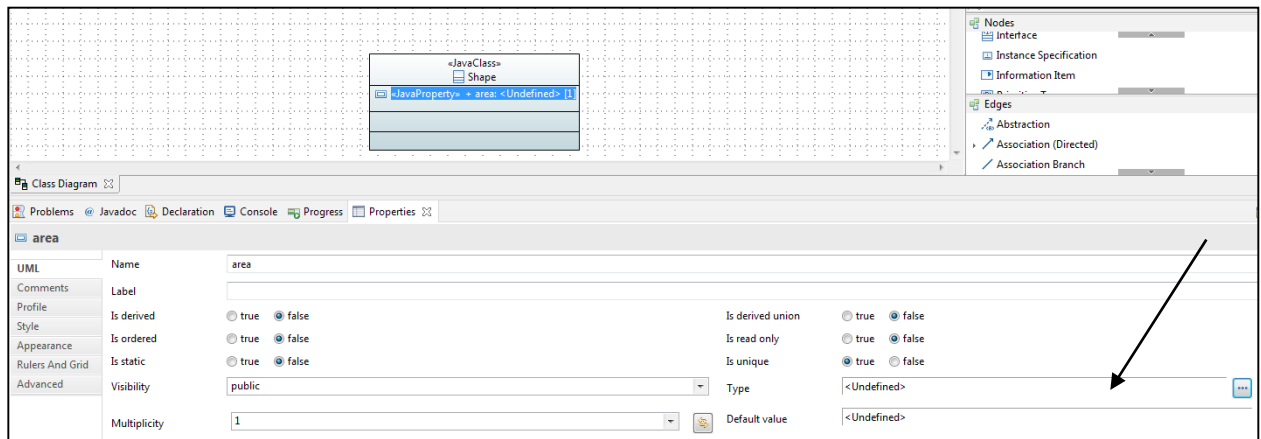


15. Shuffle JavaProperty to applied stereotypes.

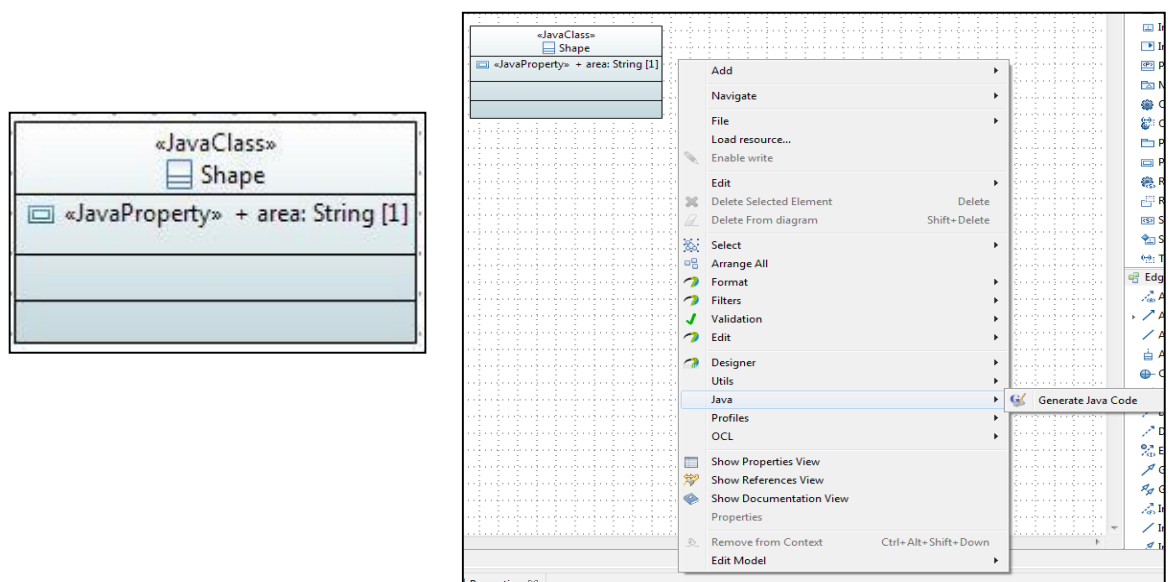




16. Select the property “area” from class diagram and change the type from undefined to string.



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



### Generated Code:

