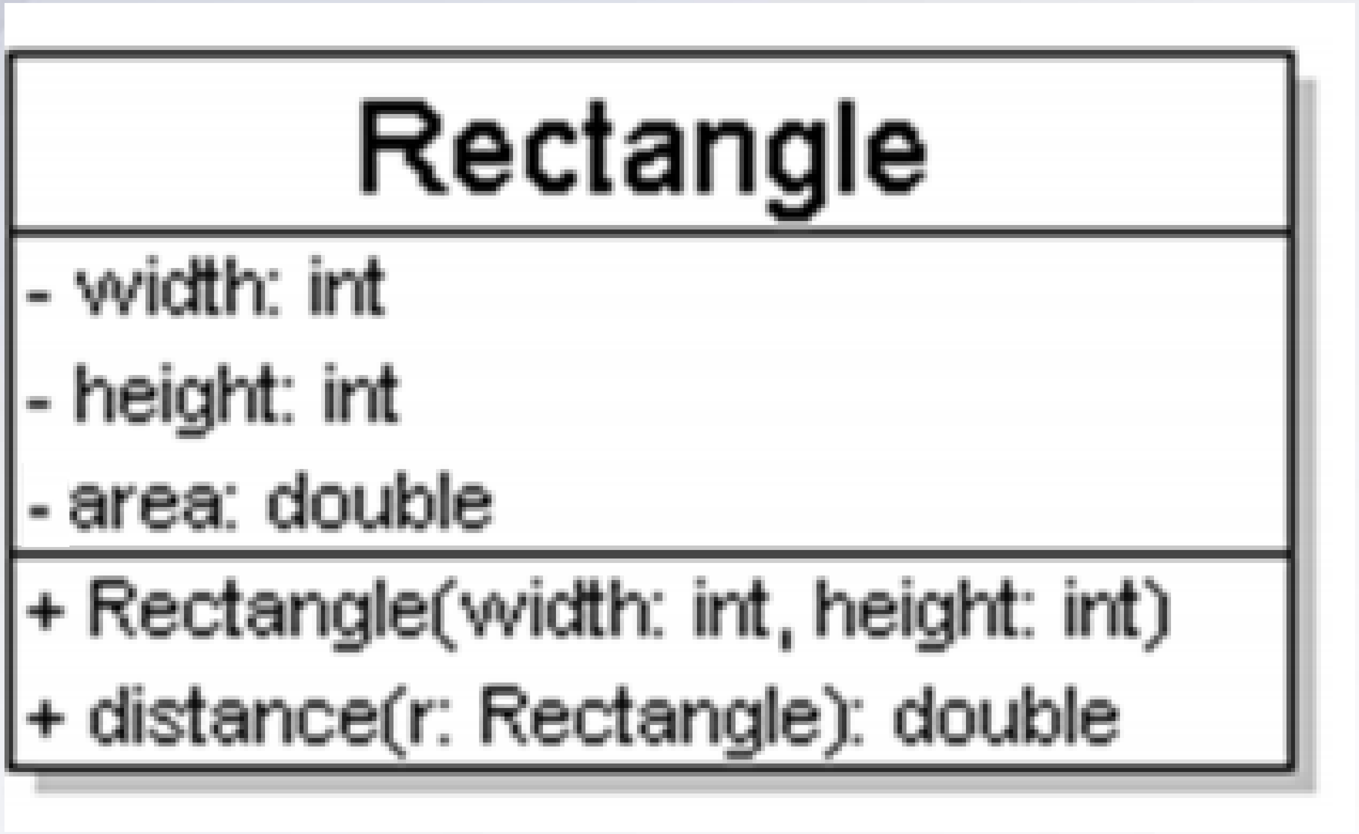
# Class Diagrams in Enterprise Architect

Department of Software

## a class Diagrams

Classes are represented by rectangles which either carry only the name of that class, or also the attribute and operations. The three compartments - Class name, Attributes, Operations - are each divided by a horizontal line. Class names usually start with a capital letter and are mostly substantive in singular (collection classes, among others, in plural where applicable).

The attributes of a class are noted with at least their names, and can contain additional data pertaining to their type, an initial value, attribute values and constraints. Methods are also noted with at least their name, as well as with possible parameters, their type and initial values, as well as possible attribute values and constraints.


Rectangle
- width: int
- height: int
- area: double
+ Rectangle(width: int, height: int)
+ distance(r: Rectangle): double

Note :

- class name in top of box
- write < interfaces > on top of interfaces' names

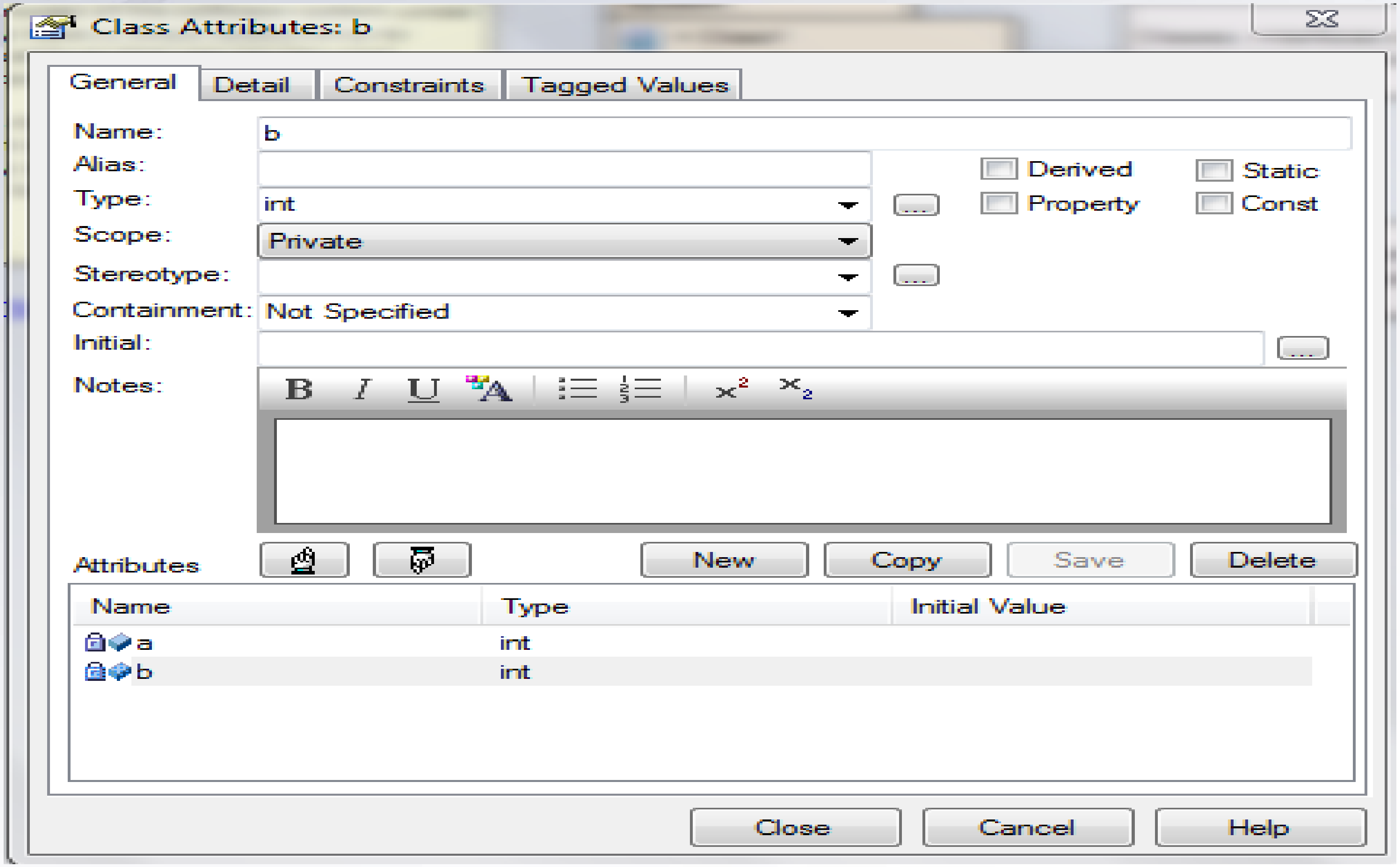
- attributes (optional)
- should include all fields of the object

- operations / methods (optional)
- may (get/set) methods

ENTERPRISE ARCHITECT

## To create attribute :

## Right click on class →attribute


Class Attributes: b

Name: b

Alias:

Type: int

Scope: Private

Stereotype:

Containment: Not Specified

Initial:

Notes:

Derived [ ] Static [ ] Property [ ] Const [ ]

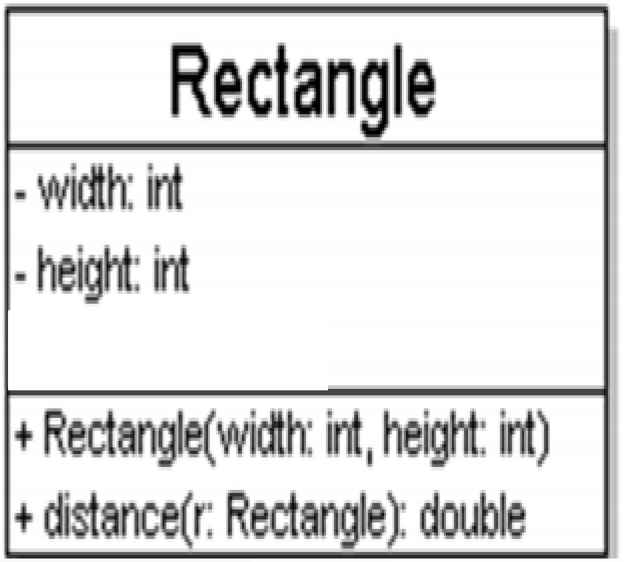
| Name | Type | Initial Value |
|---|---|---|
| a | int | |
| b | int | |

ENTERPRISE ARCHITECT

## create attribute :

- attributes (fields, instance variables)
  - *visibility name : type* [count] = *default_value*
  - visibility:

| | |
|---|---|
| + | public |
| # | protected |
| - | private |
| ~ | package (default) |


Rectangle
- width: int
- height: int
+ Rectangle(width: int, height: int)
+ distance(r: Rectangle): double

UML has the following types of **visibility**:

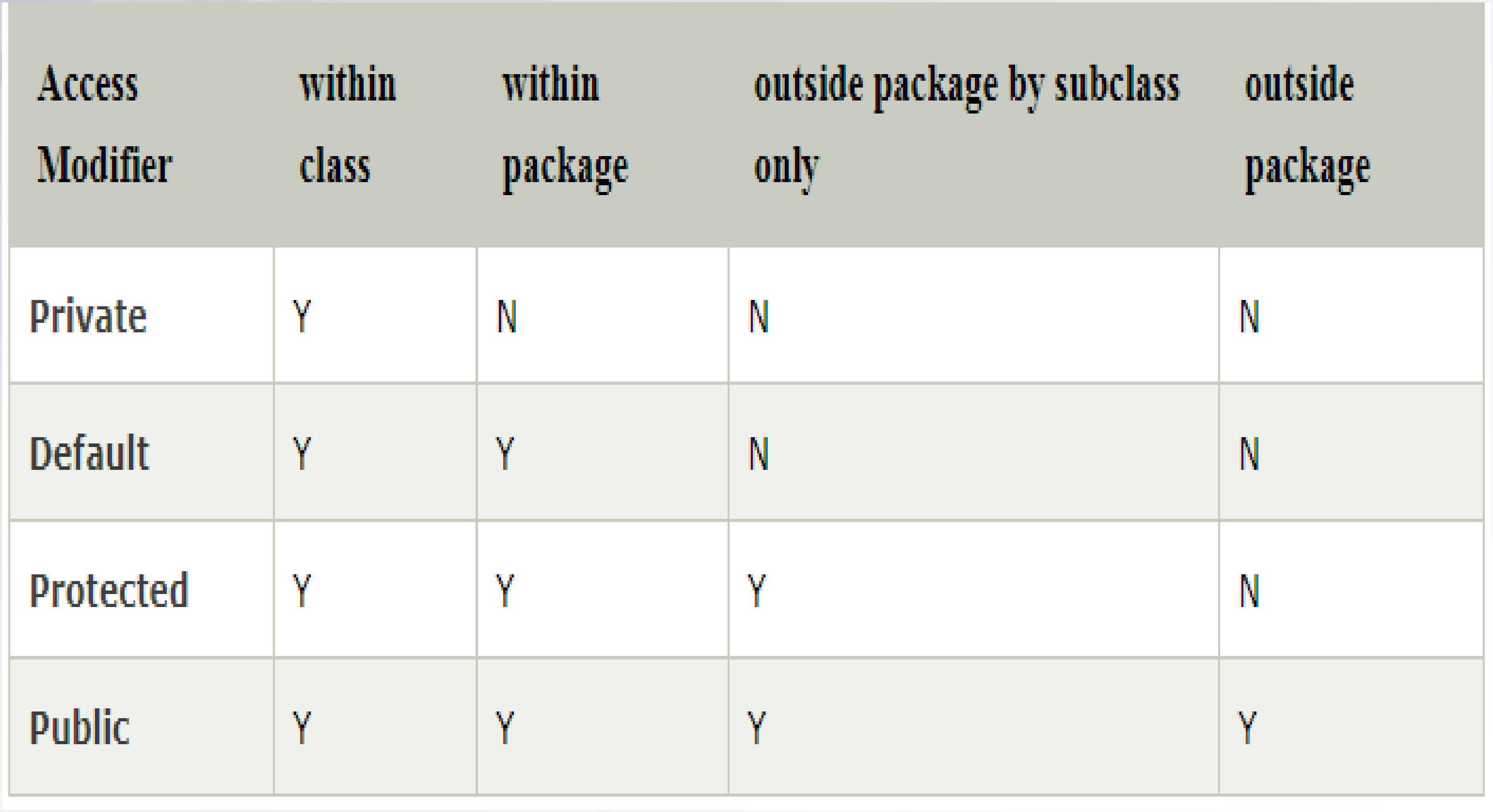
**Private**: The access level of a private modifier is only within the class. It cannot be accessed from outside the class.

**package** (**Default)**: The access level of a default modifier is only within the package. It cannot be accessed from outside the package. If you do not specify any access level, it will be the default.

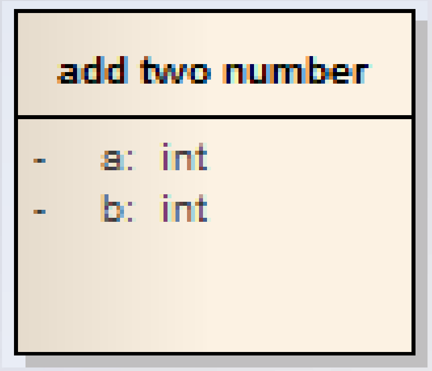
**Protected**: The access level of a protected modifier is within the package and outside the package through child class. If you do not make the child class, it cannot be accessed from outside the package.

**Public**: The access level of a public modifier is everywhere. It can be accessed from within the class, outside the class, within the package and outside the package.

ENTERPRISE ARCHITECT

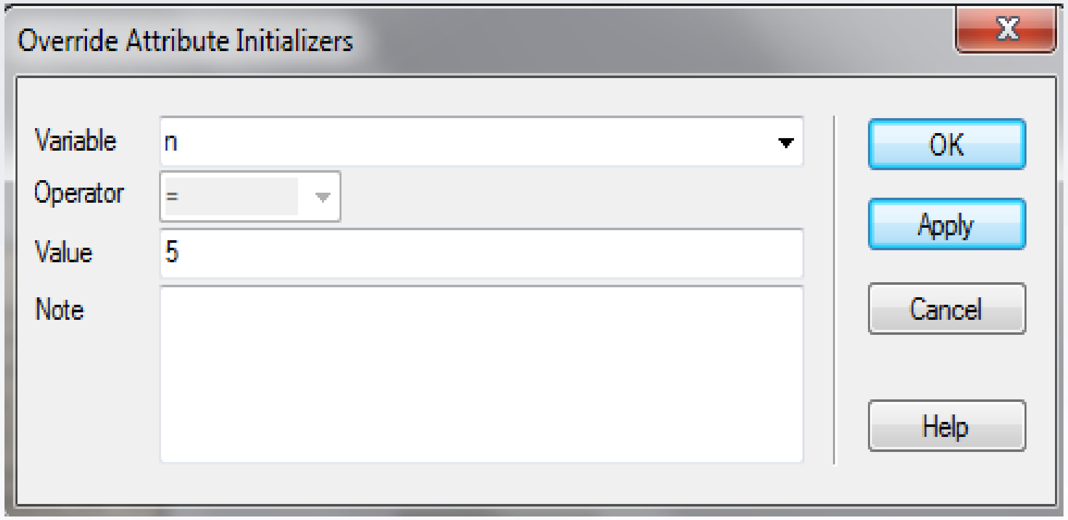


| Access Modifier | within class | within package | outside package by subclass only | outside package |
|---|---|---|---|---|
| Private | Y | N | N | N |
| Default | Y | Y | N | N |
| Protected | Y | Y | Y | N |
| Public | Y | Y | Y | Y |

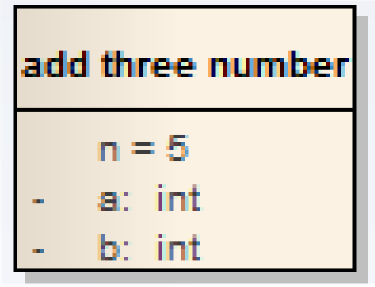


## To add initial value:

**Right click on class →advance → override attribute initializers**

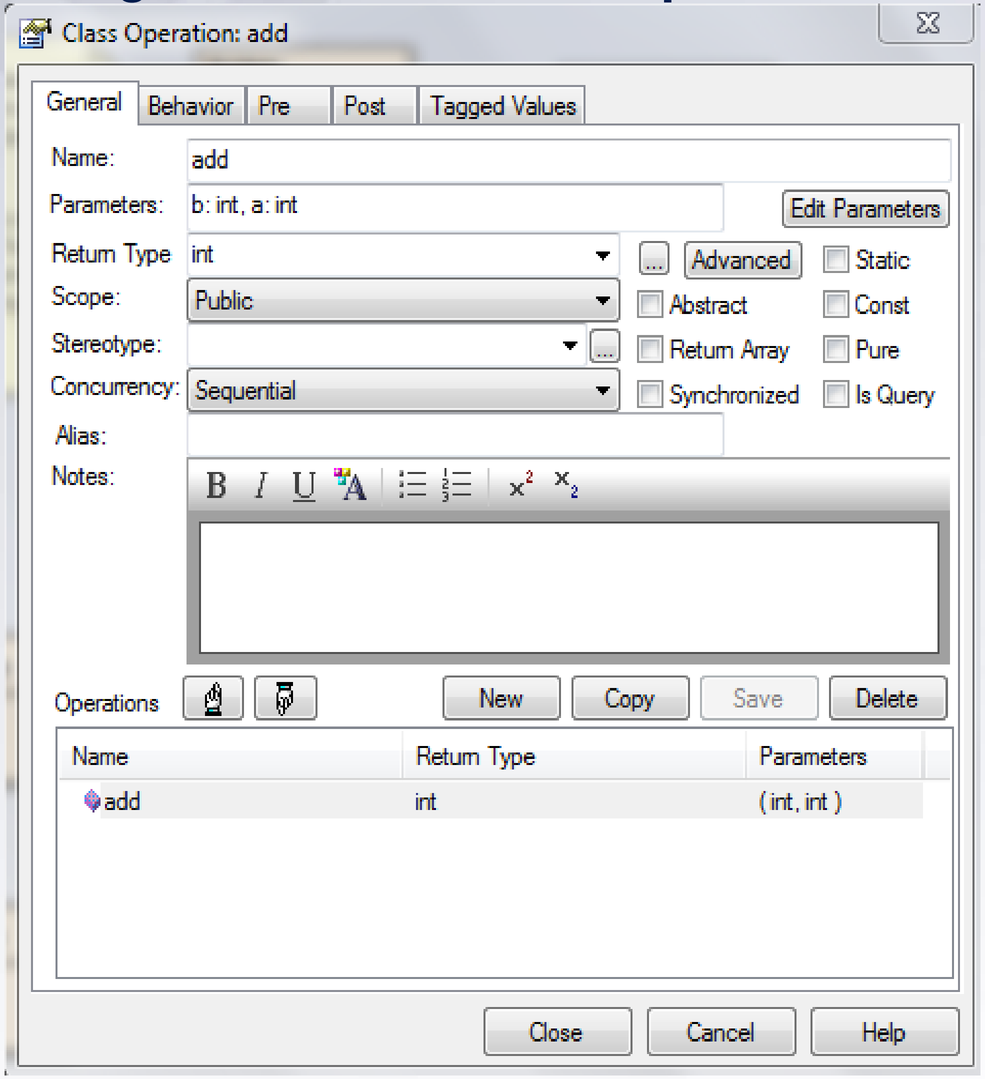

Override Attribute Initializers

Variable: n
Operator: =
Value: 5
Note:



## To create operation :

## Right click on class →operation



Class Operation: add

General | Behavior | Pre | Post | Tagged Values

Name: add

Parameters: b: int, a: int

Return Type: int

Scope: Public

Stereotype:

Concurrency: Sequential

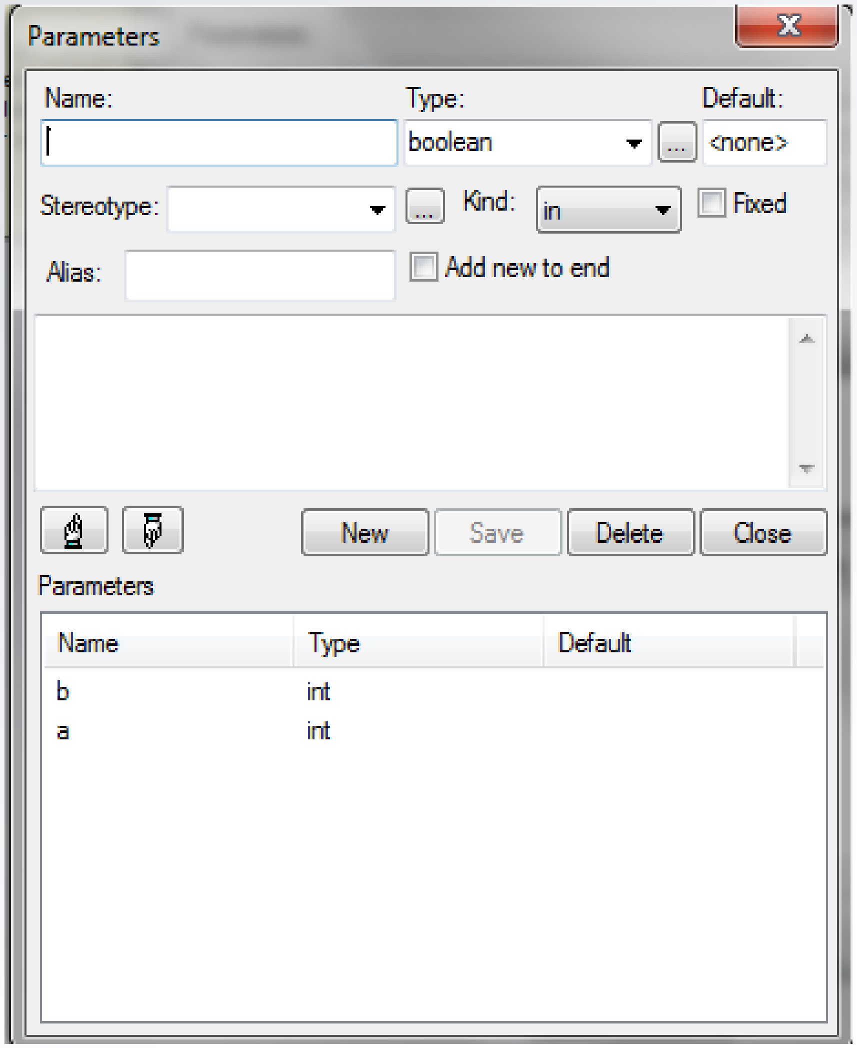
Alias:

Notes:

Static [ ] Const [ ] Pure [ ] Is Query [ ] Abstract [ ] Return Array [ ] Synchronized [ ]

Operations

| Name | Return Type | Parameters |
|---|---|---|
| add | int | ( int, int ) |



Parameters

Name:

Type: boolean

Default: <none>

Stereotype:

Kind: in

Fixed [ ]

Alias:

Add new to end [ ]

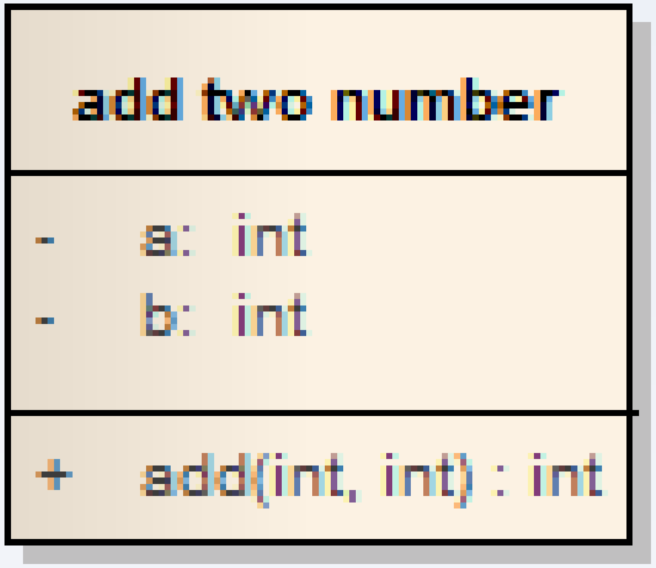
Parameters

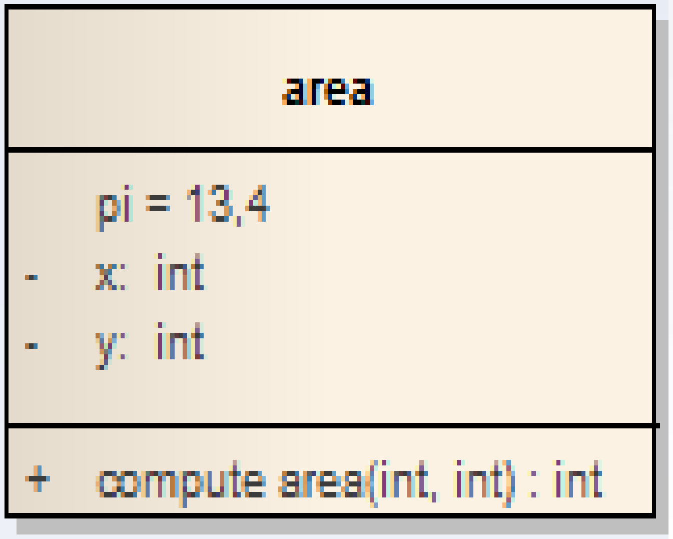
| Name | Type | Default |
|---|---|---|
| b | int | |
| a | int | |

Where a is private and integer number

b is private and integer number

add if function received two integer number and return integer number(a+b)



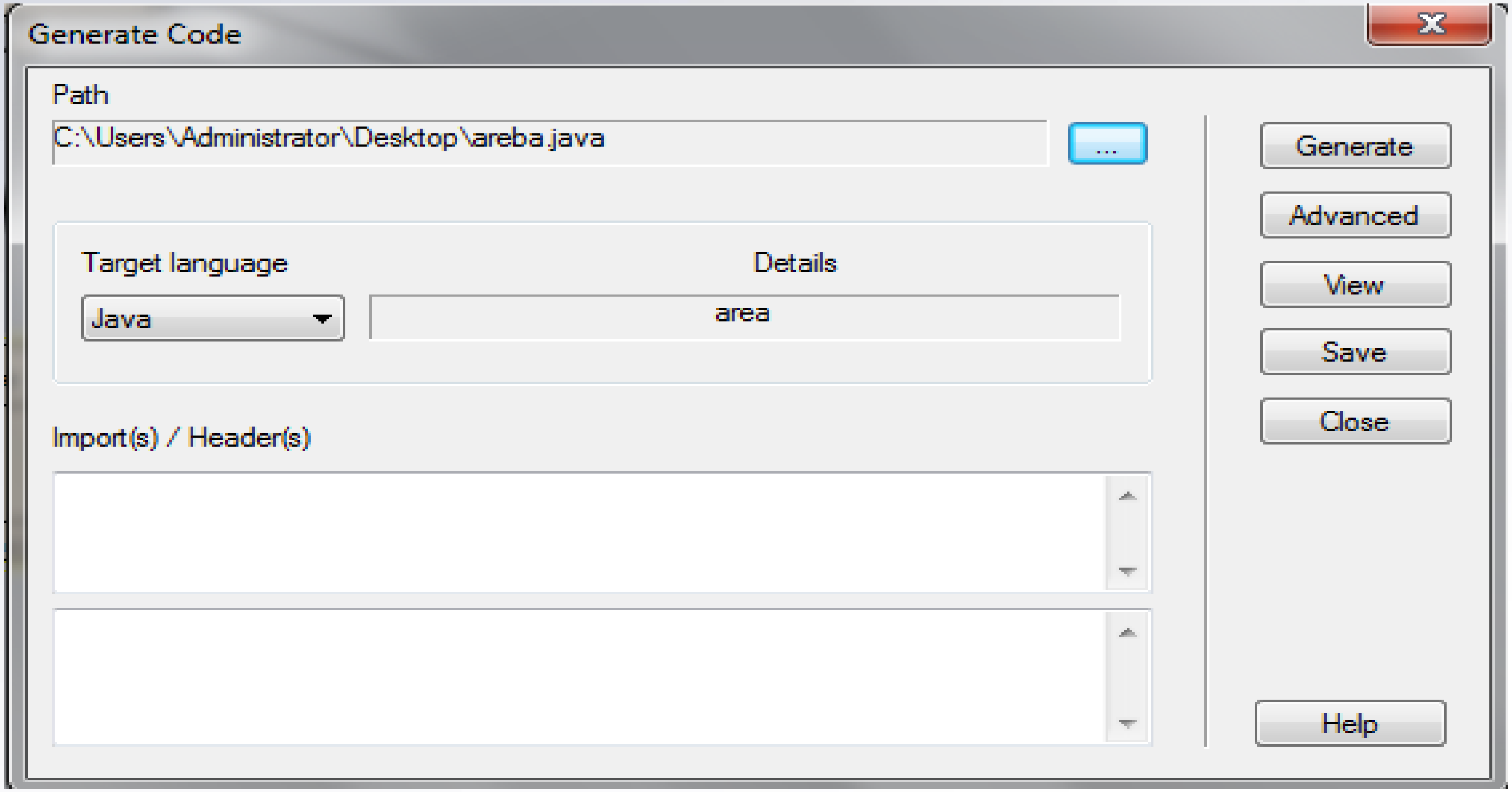


ENTERPRISE ARCHITECT

## Code Generation

**To generate code :**

**Right click on class → generate code**


Generate Code

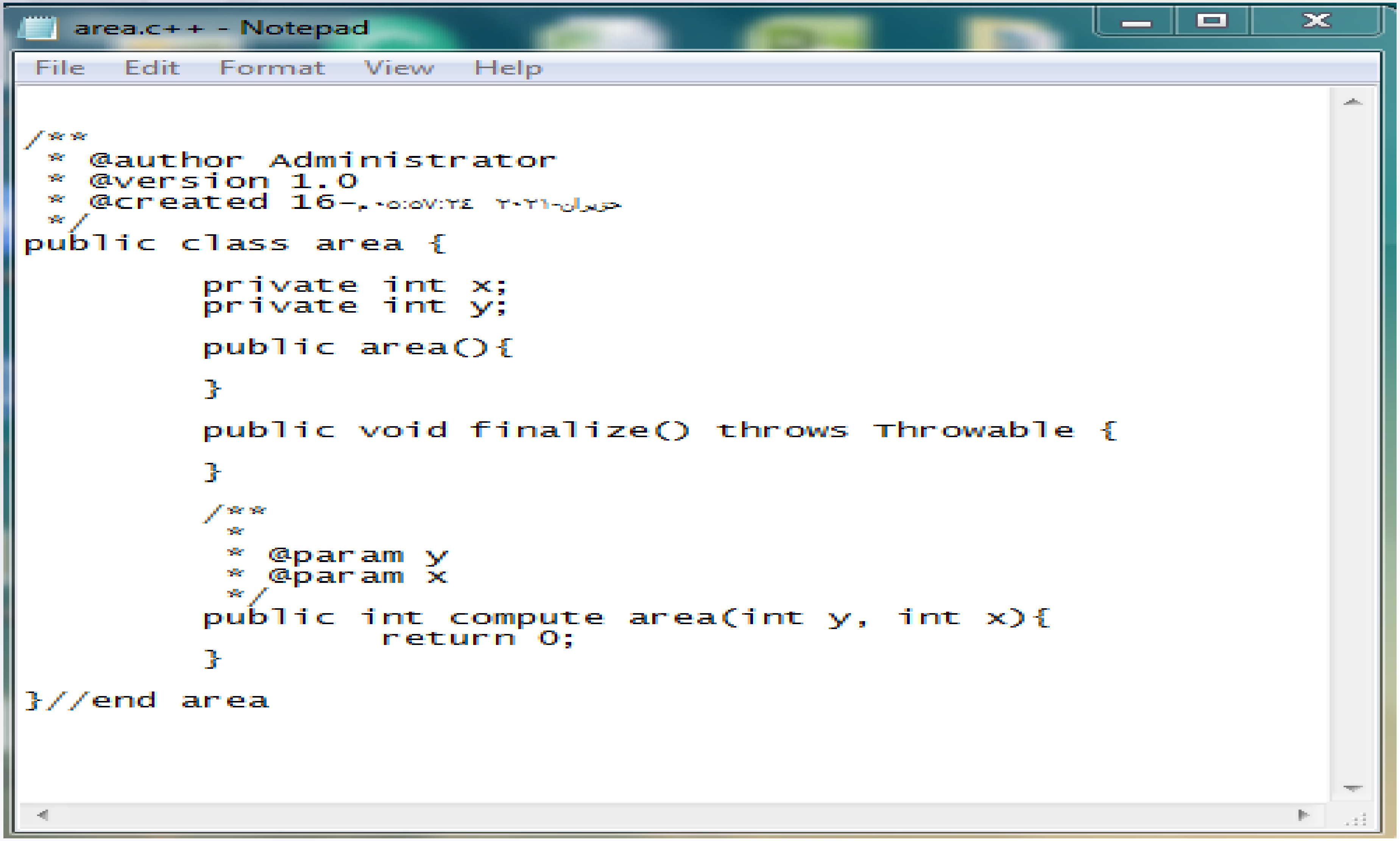
Path: C:\Users\Administrator\Desktop\areba.java

Target language: Java

Details: area

Import(s) / Header(s)

## Code Generation

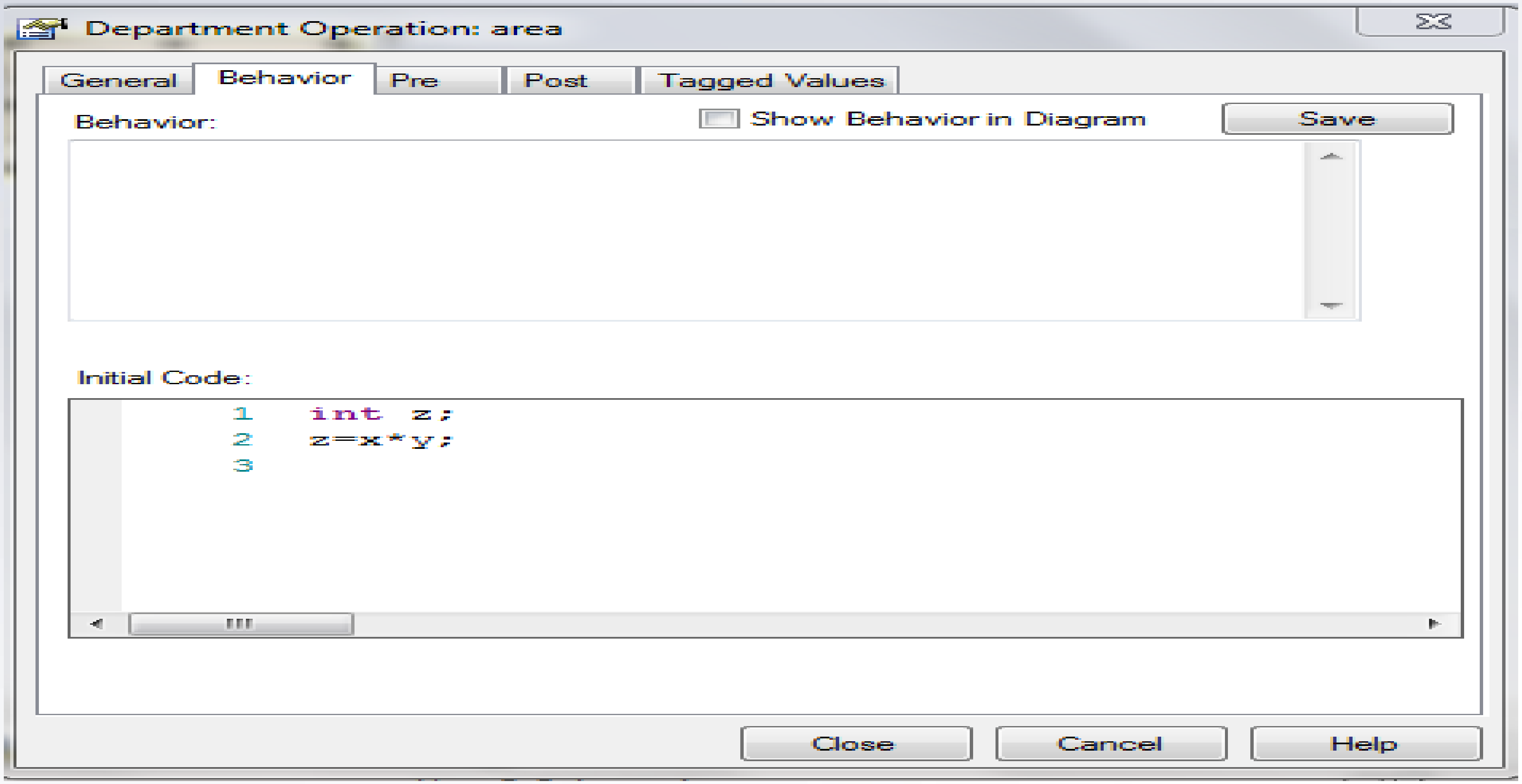


```
/**
 * @author Administrator
 * @version 1.0
 * @created 16-حزيران-٢٠٢١ ٠٥:٥٧:٢٤ م
 */
public class area {

        private int x;
        private int y;

        public area(){

        }

        public void finalize() throws Throwable {

        }

        /**
         *
         * @param y
         * @param x
         */
        public int compute area(int y, int x){
                return 0;
        }

}//end area
```

## Code Generation

To add code in method :

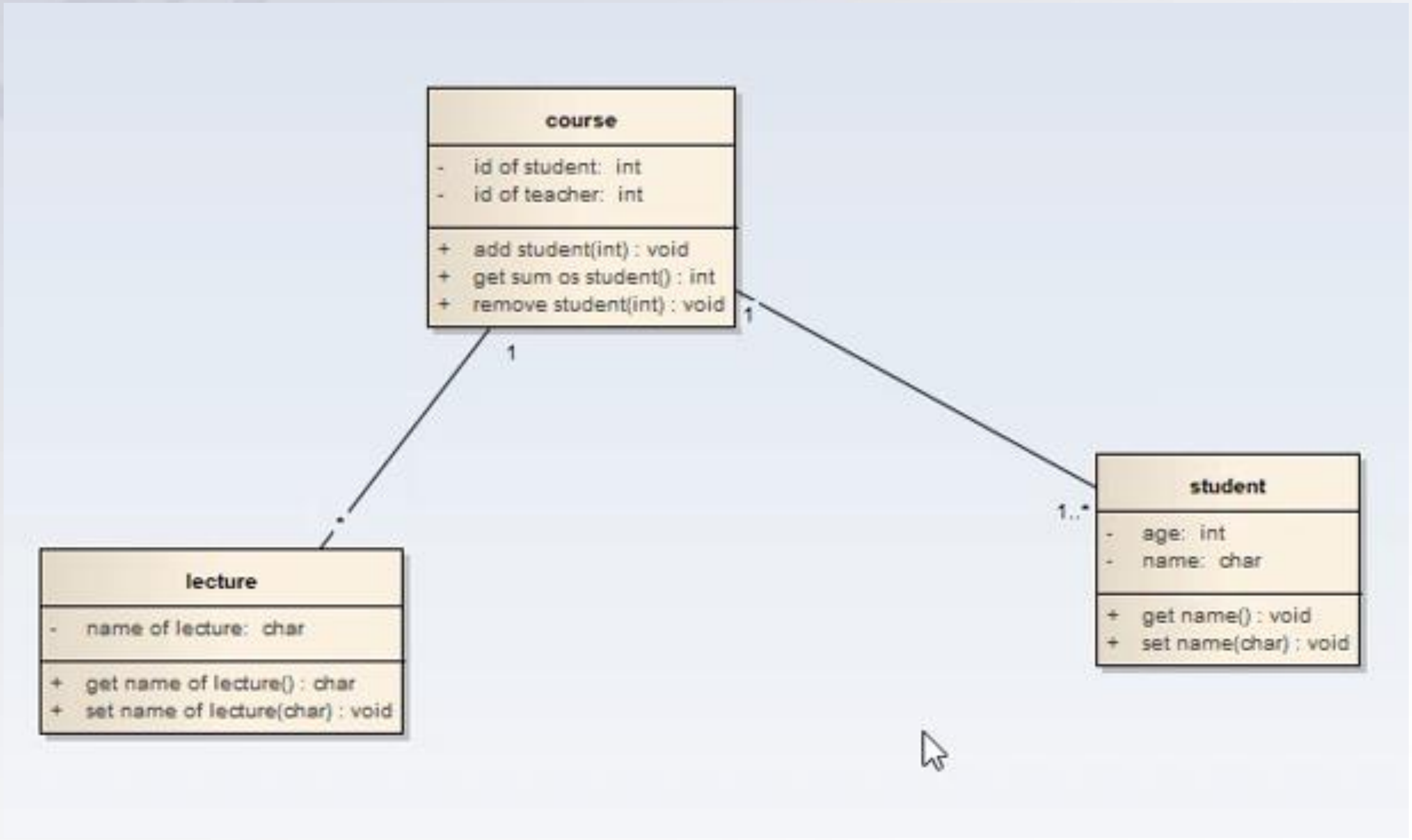
Select operation → behavior



```
int z;
z=x*y;
```

ENTERPRISE ARCHITECT

example



Example

**Design class Diagram for Car that contain:**

1-car model
2-car
3-Engine
4-GearBox
6-Door
7- Brake
8- tire
9-wheel

**and create attribute and operation for each class**

SPARX SYSTEMS

# Thank You