



This work is licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 3.0](https://creativecommons.org/licenses/by-nc-sa/3.0/)

Created by: Amir Alfoly

Email: [amir.alfoly@gmail.com](mailto:amir.alfoly@gmail.com)

# ArgoUML Class Diagram Tutorial

---

## 1 Introduction

This tutorial is intended to show how to use **ArgoUML** modeling tool to create a class diagram and later converts it to source code. Although the tutorial is captured using ArgoUML on Windows Vista, the concepts and steps are the same if ArgoUML is used on any version of Windows OS.

## 2 Installing ArgoUML

For those who don't have ArgoUML installed or want to start using it, you can visit the website: <http://argouml.tigris.org/>, and download the latest version, this tutorial is based on version

**0.30.2**

Since ArgoUML is built on java, it requires Java Runtime Environment which is provided with the installation package for Windows operating systems.

## 3 Starting the application

1. Open Windows Start Menu
2. Open All Programs
3. Search for and open "ArgoUML->ArgoUML"
4. The application should start and shows the following main screen.

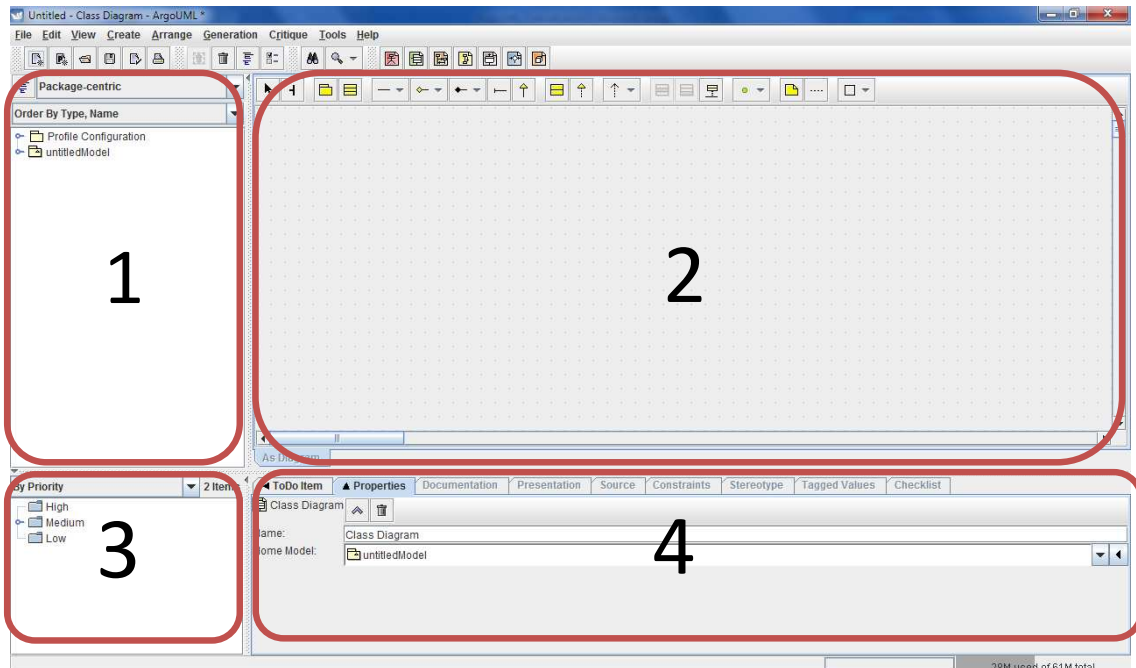


Figure 1: ArgoUML Main Screen

## 4 ArgoUML Main Screen

The top of the window contains a menu bar with commands available. In the File menu you can save the project or open another project instead.

As shown in figure 1, the main default screen is divided into 4 parts:

1. **Upper left: Tree model for diagrams and objects:** shows a tree model of diagrams and objects. This view can be adapted to your needs by filtering the objects that are shown, and the structure in which they are shown.
2. **Upper right: Current Diagram:** shows the current diagram (one at a time). You can drag and drop the objects in the diagrams, and you can use the quick-links that appear when hovering over a selected object to create new objects connected to the already present objects.
3. **Lower left: To Do list:** list of all **To Do** items for this model.
4. **Lower right: Details of the selected object:** contains various details of the currently selected object, Select the object in one of the upper levels and choose what details you want to examine using the tabs.

## 5 Designing the Class Diagram

### 5.1 The Storyline

Knowing a university system, a university hosts a number of faculties; every faculty contains a number of departments. In each department a number of courses are offered per semester, each course has an instructor and students can register for it through the Registrar office.

### 5.2 Defining the Entities

We are going to build a class diagram that represents that sample **University System**. But before building a class diagram we must pass by the activity of defining the entities in our system, we can do that by passing on nouns in the storyline, using this method we can list but not constrained to the following:

1. Faculty
2. Department
3. Course
4. Semester
5. Instructor
6. Student
7. Registrar Office

#### 5.2.1 Defining the attributes

The attributes are the properties of each class, for example a student can have the properties: full name, birth date and national id. While a course can have the properties: Code, Credit Hours. Each attribute will have a data type which defines how data will be stored in it; like a string or a number.

#### 5.2.2 Defining the functions (Operations)

The functions represent the actions that can be done by entities, for example: the student can register a course, a Registrar Office can open registration and close registration and accept course request. Each function can take multiple input parameters and return only one object.

#### 5.2.3 Defining the relations between classes (Associations)

As we may already have noticed, there exist relations between the entities presented in the storyline, for example the **Faculty *has more than one* Department**, and the **Department *has more than one* Course**. This relation can be presented in UML by what is named: **Associations**.

An **Association** links between two classes and at each end a **multiplicity** factor is defined, this multiplicity factor shows how each class relates to the other one, for example the **Faculty** must have ***one or more* Departments** and this is represented by “**1..\***” label, but the **Department** will be associated ***with only one* Faculty** and this is represented by “**1**” label. For more examples please check Table 2.

### 5.2.4 The Big Picture

So before we start modeling we will outline all our entities, their attributes, operations and associations.

Table 1: Sample Design for the University System Classes

Class Name	Attributes	Operations
Faculty	Name : <b>String</b>	void <b>AddDepartement</b> (dep : <b>Departement</b> )
Department	Name : <b>String</b>	void <b>AddCourse</b> (course : <b>Course</b> )
Course	Code : <b>String</b> CreditHours : <b>Integer</b>	
Semester	Year : <b>Integer</b> Season : <b>String</b>	void <b>RegisterCourse</b> (course : <b>Course</b> , instructor : <b>Instructor</b> )
Instructor	Name : <b>String</b> BirthDate : <b>Date</b> Room : <b>String</b>	
Student	Name : <b>String</b> BirthDate : <b>Date</b> RegistrationYear : <b>Integer</b>	
RegistrarOffice		void <b>OpenRegistration</b> (void) void <b>CloseRegistration</b> (void) void <b>RegisterCourse</b> (student : <b>Student</b> , course : <b>Course</b> , semester : <b>Semester</b> )


Table 2: Sample Associations between classes

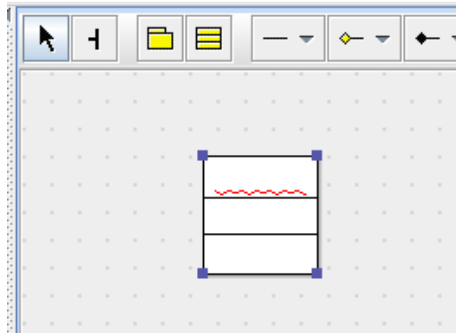
Right Side Class	Multiplicity	Multiplicity	Left Side Class
Faculty	1	1..*	Department
Faculty	1	0..*	Student
Department	1	1..*	Course
Course	0..*	0..*	Semester
Course	0..*	1	Instructor
Course	0..*	0..*	Student

## 6 Building the Class Diagram using ArgoUML

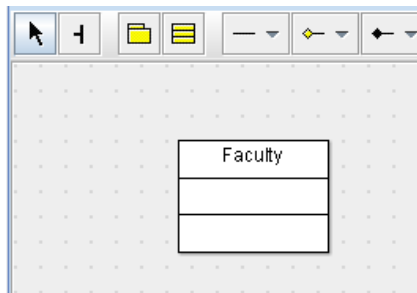
We will start to put what we have designed before into action using **ArgoUML** modeling tool, so let us start by adding a class into the **Diagram Block** (Block no. 2 in Figure 1).

### 6.1 Add a Class

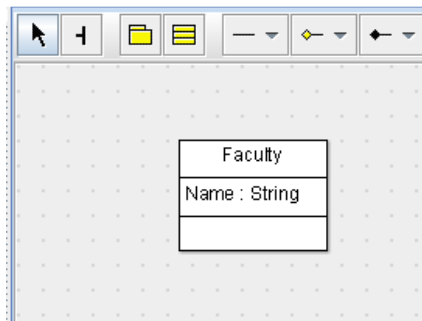
1. To add a class, click on the icon  in the toolbar located just above the diagram area
2. It will make the mouse pointer becomes **+**
3. Move the mouse pointer to the diagram area and click once, the application will create an empty class as shown in the next figure.



4. The class is partitioned into three parts:
  - a. Top Part: will contain the class name
  - b. Middle Part: will contain the class attributes
  - c. Lower Part: will contain the class functions
5. Double click on the top part and it will allow you to input the class name, type the first class name: **Faculty**, the class should look like the following figure.

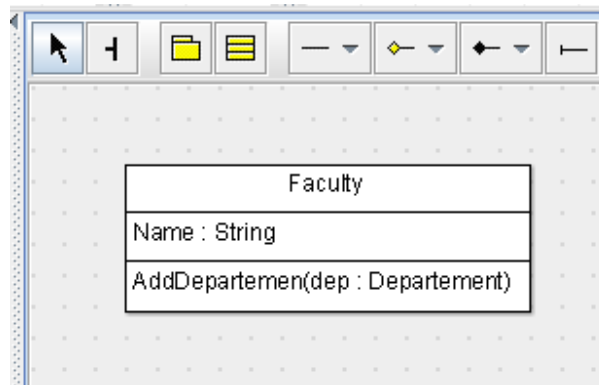


6. Now we are ready to add attributes, double-click on the middle part, the application will generate an attribute "**newAttr : Integer**" with the name "**newAttr**" and data type "**Integer**"
  - a. Change the name to: **Name**
  - b. Change the data type to: **String**
  - c. The class now should look like the following figure.

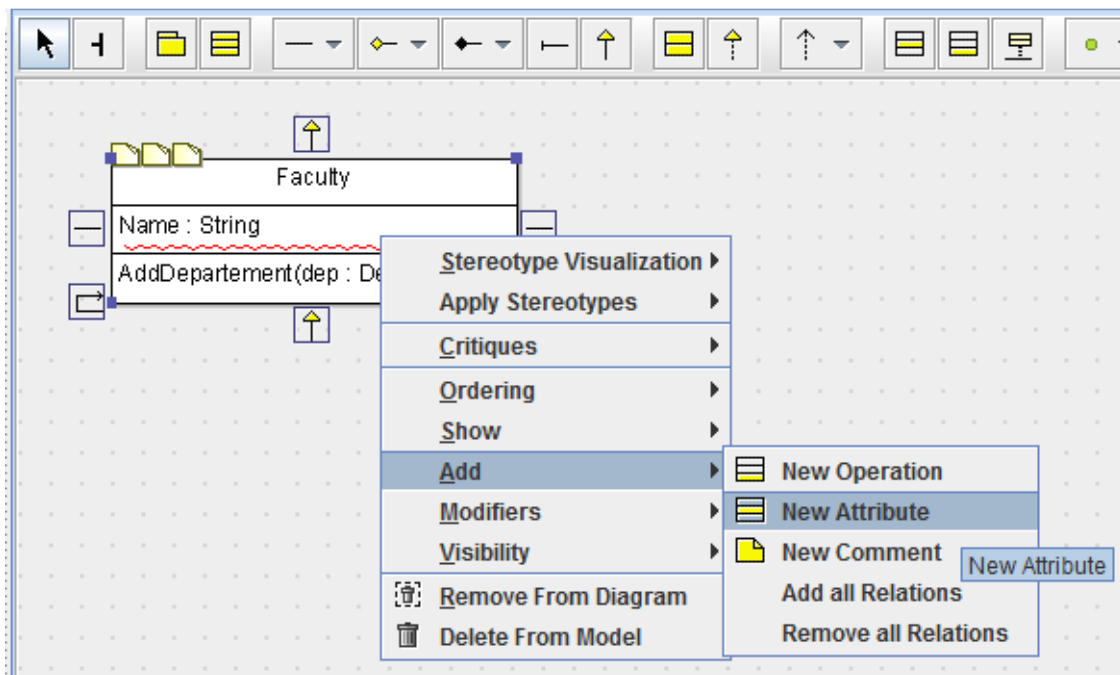


7. Now we will add our first function,
  - a. Double-click on the lower part; the application will generate a sample function definition.

- b. Change the name of the function from **newOperation** to **AddDepartement**
- c. To define the parameters type within the **brackets ()** the parameter name and data type as follows: **dep : Departement**
- d. The class should now look like the following figure



8. To add another attribute:
  - a. Right click on the attributes part and go to **Add->New Attribute** as shown in the following figure.
  - b. Repeat the steps of defining the attributes name and datatype.
9. To add another function (Operation):
  - a. Right click on the attributes part and go to **Add->New Operation** as shown in the following figure.
  - b. Repeat the steps of defining the function name and parameters.

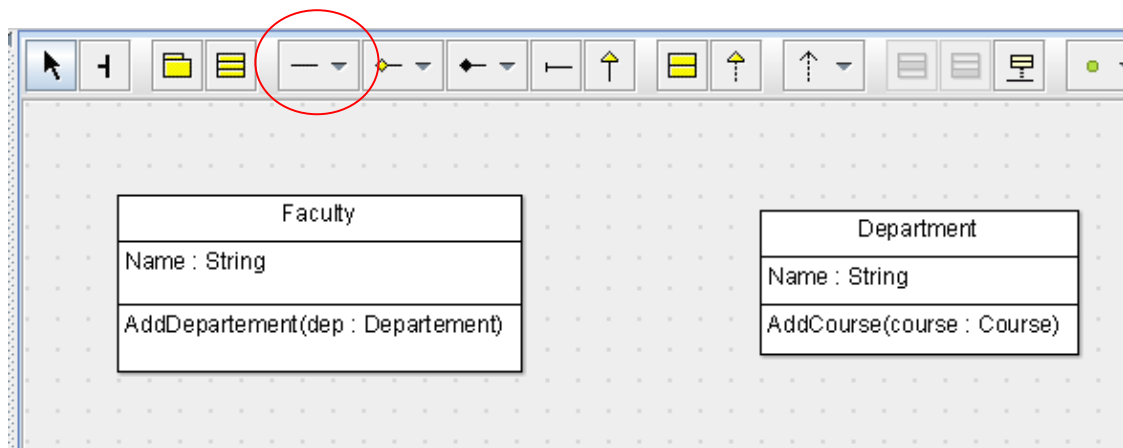


10. Repeat the above steps to define all the classes as shown in **Table 1**

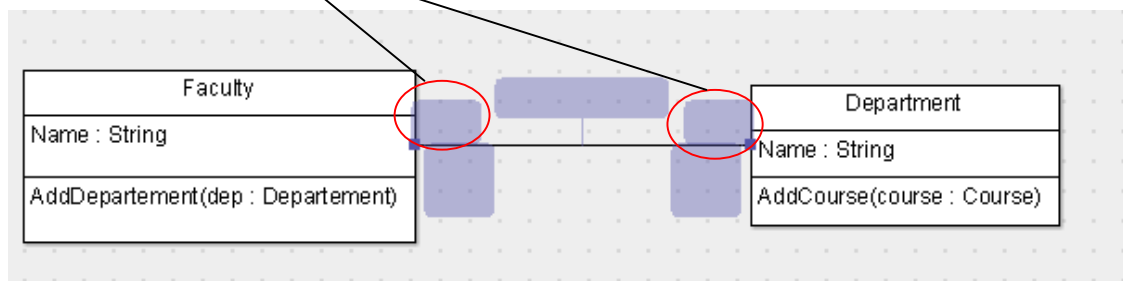
## 6.2 Add Class Associations

As shown in Table 2, we will define the associations between classes in **ArgoUML**, we will show how to define the association between the **Faculty** class and the **Department** class.

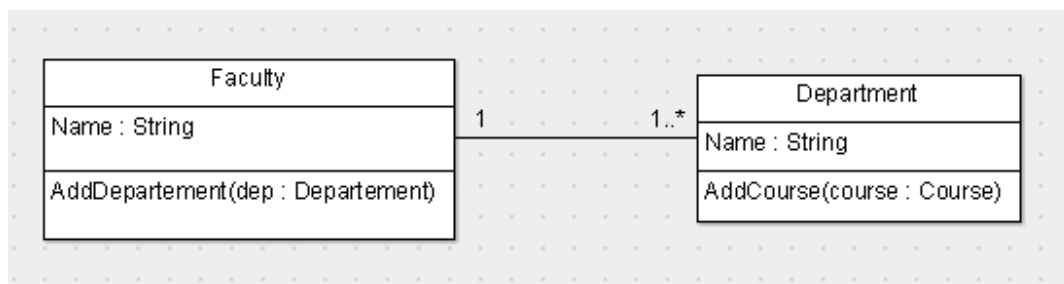
1. As shown in the below figure, we have the **Faculty** and the **Department** classes in the diagram



2. Click on the Multiplicity icon on the toolbar above the diagram and draw a line between the two classes, the diagram should look like the following figure.



3. Double click on each box labeled in red in the above figure.
  - a. Write the multiplicity factor as defined in Table 2, where it will be "1" from the side of "Faculty" and "1..\*" from the side of the department
  - b. The diagram now should look like the following figure

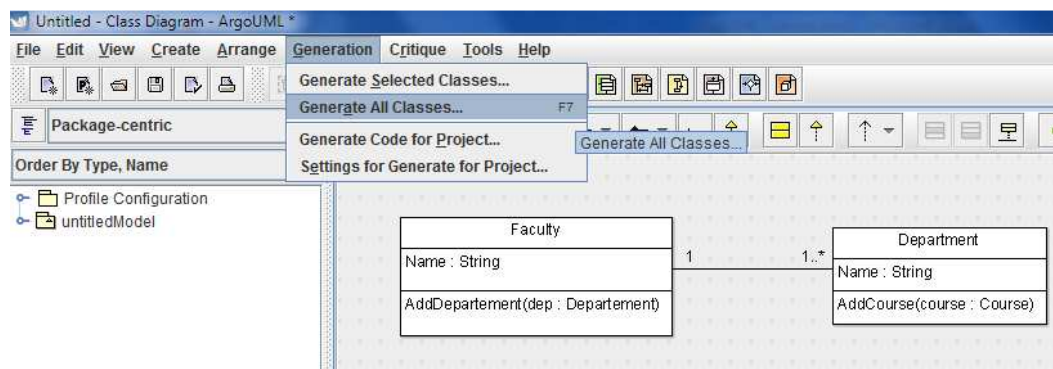


- Repeat the above steps for each Association defined in Table 2, the final diagram should show all the associations between the classes.

### 6.3 Generate Source Code from UML Diagram

ArgoUML provides a very useful tool to generate source code for the modeled classes, to do that:

- Create a folder on the **Desktop** named **GeneratedCode**
- Go to the menu item and click on: **Generation->Generate All Classes** as shown in the next figure (or press **F7**).



- Click **Select All** button to select all classes for generation
- In the **Output Directory** select the created folder **“GeneratedCode”**
- Click the button **“Generate”** then check the created files in the folder.

