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 # 02

OBJECTIVES:

- **Papyrus Software Installation guide (Recap from Week-01)**
- **To understand Domain Model and User Stories.**
- **Hands-on Domain Model Practice.**

INTRODUCTION

USER STORIES

A user story can be defined as the well-expressed requirement from the perspective of the end user and his/her goals. A user story has become popular in agile for several reasons, most important reasons being clarification of the requirement from the user point of view, identification of the user goals, and why we need that requirement in the first place.

User stories can be handwritten or can be written in a simple card.

The format of the User Story is as follows:

As a < **role**>

I need <**requirement or feature**>

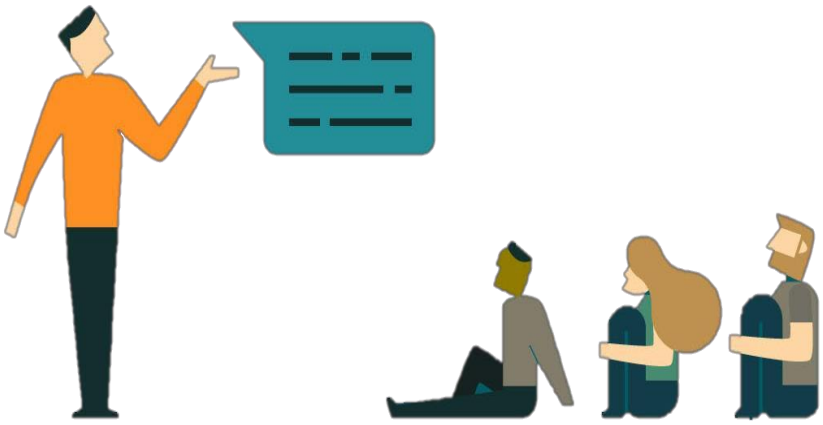
So that <**goal / value**>

Example:

As a **Marketing Director,**

I need **to improve customer service**

So that **we retain our customers.**



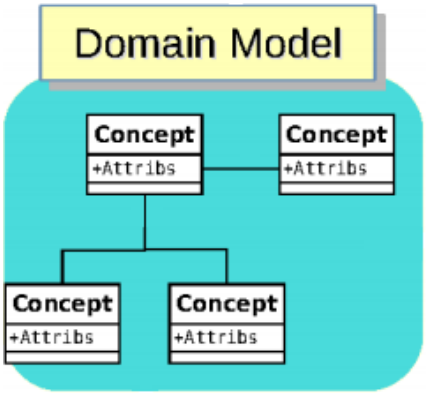
DOMAIN MODEL

The domain model (also known as the conceptual model, domain object model or analysis object model) is created to decompose the problem in hand to real world concepts or objects. The domain model identifies a set of conceptual classes (ideas, things or objects in the domain). This becomes the basis of the design of the software.

Domain model covers:

- The conceptual classes
- Attributes of the conceptual classes
- Associations between the conceptual classes

It should be noted that no operations are shown in the domain model.



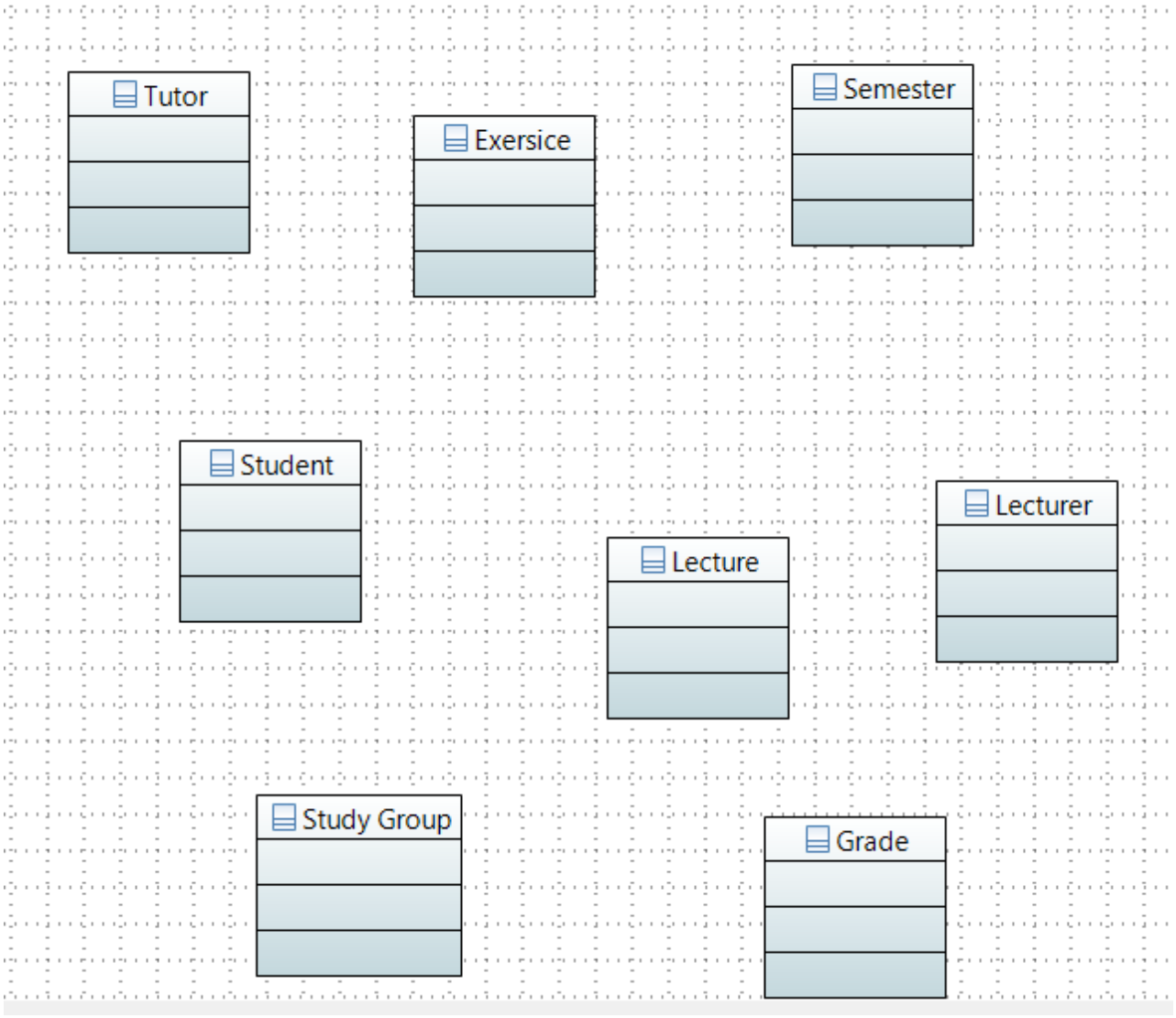
STEPS TO CREATE A DOMAIN MODEL

1. Create User Stories
2. Identify candidate conceptual classes
3. Draw them in a UML domain model
4. Add associations necessary to record the relationships that must be retained
5. Add attributes necessary for information to be preserved
6. Use existing names for things, the vocabulary of the domain

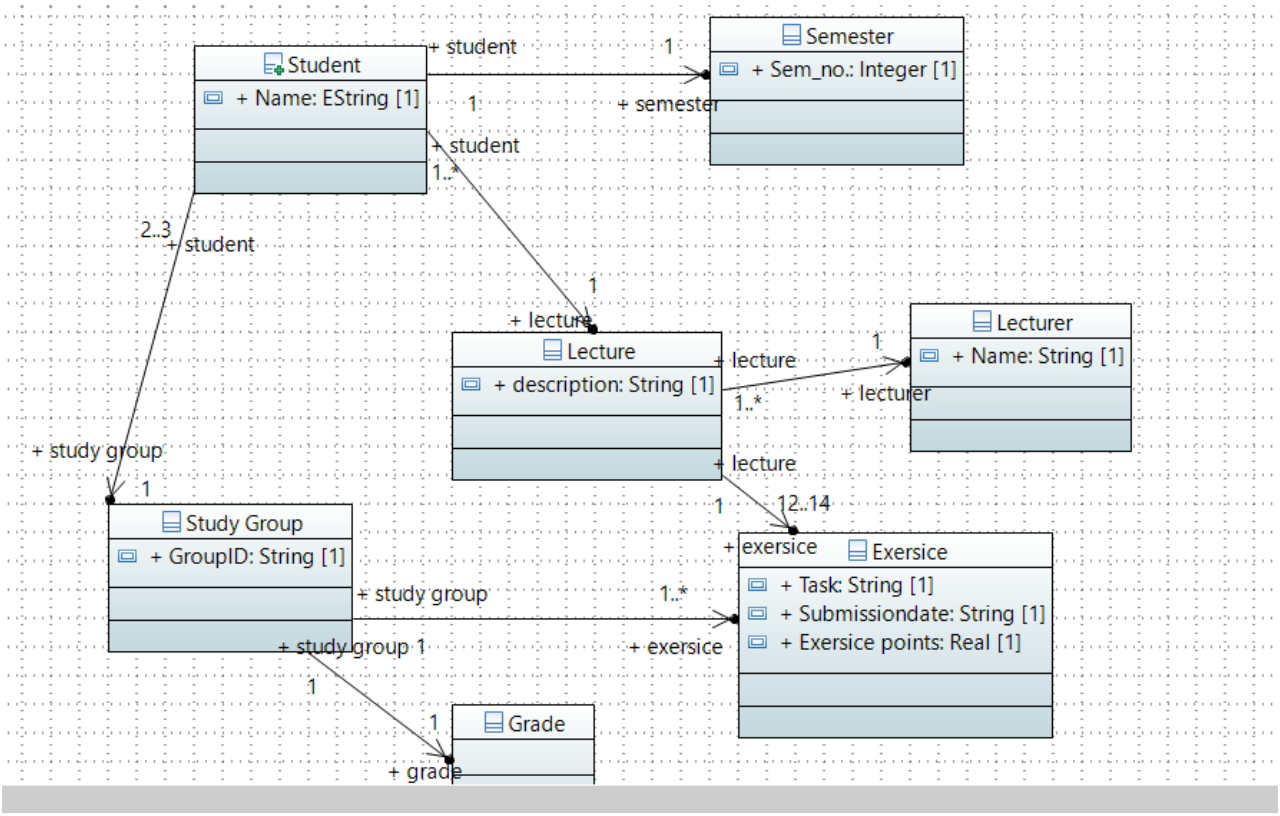
EXAMPLE (University class)

- During a semester a lecturer reads one or more lectures
- A student usually attends one or more lectures
- During the semester there will be several exercises
- Each student is assigned to one particular study group for the whole semester
- Each student is graded by a tutor

- The bonus is a relative bonus that reflects the relative number of exercise points gained during the semester



Conceptual classes



Domain model

MULTIPLICITY

Place multiplicity notations near the ends of an association. These symbols indicate the number of instances of one class linked to one instance of the other class.

Multiplicity can be expressed as,

Exactly one - 1

Zero or one - 0..1

Many - 0..* or *

One or more - 1..*

Exact Number - e.g. 3..4 or 6

a complex relationship – e.g. 0..1, 3..4,6..* would mean any number of objects other than 2 or 5

VISIBILITY

Use visibility markers to signify who can access the information contained within a class.

- Public +
- Private -
- Protected#

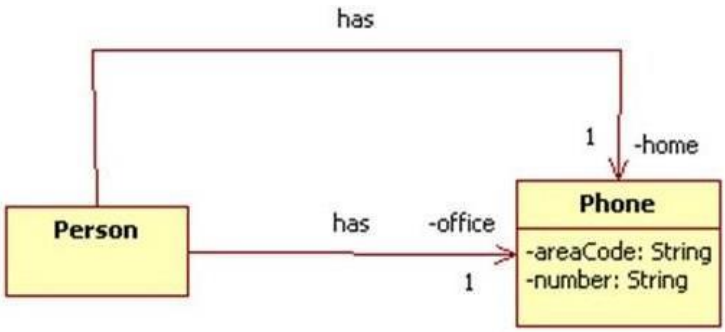
ASSOCIATION RELATIONSHIP

An association can be defined as a relationship between classes. An association relation is established, when two classes are connected to each other in any way. A class can be associated to itself too.

Examples of Associations:

Association with 1 multiplicity:

Assume some application needs to associate a home and business phone to each person in a database.

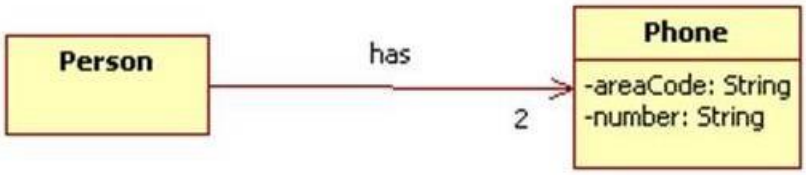


This can be represented by the following code:

```
class Person {
   private Phone home;
   private Phone office;
   // etc.
}
```

Association with more than 1 multiplicity:

Taking the example described above, we can also model the scenario as follows:

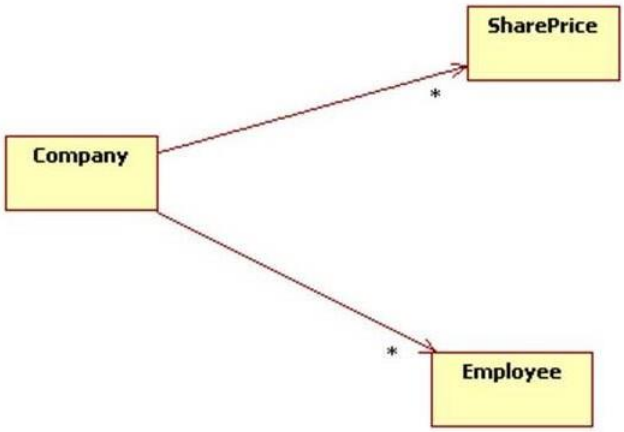


This can be represented in code as follows:

```
class Person {
   private Phone[] phones = new Phone[2];
   // etc.
}
```

Association with many (*) multiplicity:

Take the example that a company can have many employees and many share prices.



Code:

```
class Company {
   private List<Employee> employees;
   private List<SharePrice> sharePrices;
   // etc.
}
```