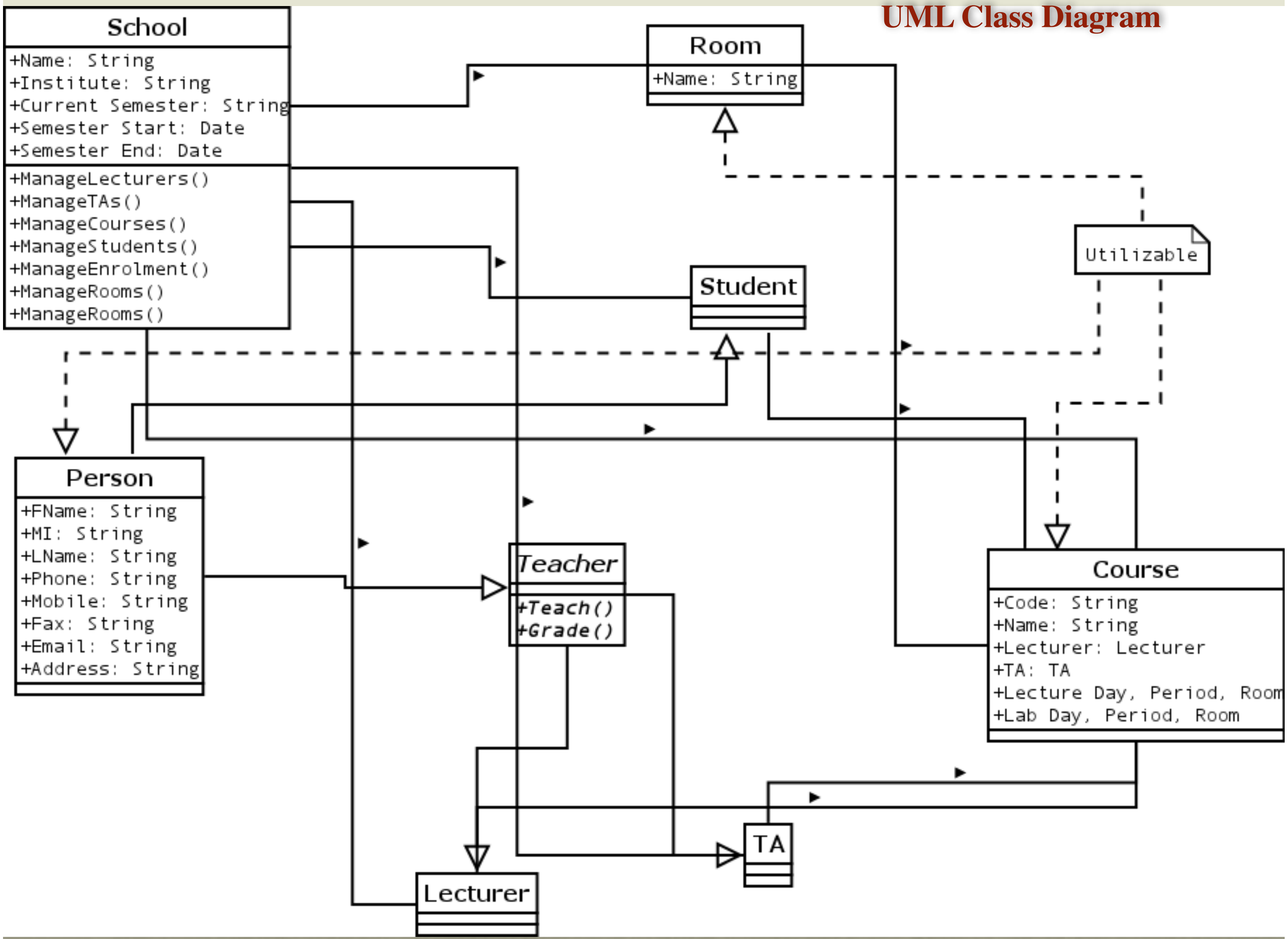
CC316: Object Oriented Programming

School Management System Case Study

Dr. Manal Helal, Fall 2015.
http://moodle.manalhelal.com

UML Class Diagram



Encapsulation & Information Hiding

- Methods (Black Box Implementation) & Methods Invocations.
- Method Overloading: Several Constructors to accept some default values, or define everything based on level of abstraction.
- Private Field and setter (mutators) and accessors (getters).

Object Association

- School is composed of: lecturers, TAs, Courses, Students, Rooms & Labs & Equipment
- Course Aggregates Students
- Course Associates with Lecturer and TA classes
- …

Inheritance

- Teacher and Student inherits from Person
- Lecturer & TA inherits from Teacher Class

Abstract Classes & Methods

- Teacher is an abstract class
- Teach and Grade are abstract methods to be implemented by concrete classes in Lecturer and TA
- Lecturer teach from slides
- TA teach from lab exercises
- Lecturer Grade Exam Questions: MCQ, Essay, … etc
- TA Grade lab exercises

Interface

- Course Class implements Cloneable interface, to clone a course to a new semester copying previous data.
- Person, Course, Room implements Utilizable interface to implement abstract methods to return no of hours worked or used:
  - WeeklyUtilisation
  - MonthlyUtilisation
  - AnnualUtilisation

Generic Programming & Polymorphism

- Calculate Weekly Utilisation for every resource in the school, invariant of class type.

```
int SchoolUtilisationReport
(ArrayList<Utilizable> resources) {

      int = 0;

      for (int i = 0; i < resources.size(); i++)

            hours += resouce.weeklyUtilisation();

      return hours;

}
```

GUI & Event Driven

- Various Frames and Controls are used.
- Action Events and Window Events are used.
- Use Window Builder Eclipse Plugin to facilitate GUI Creation and Event Handling.

Exception Handling

- In accessing files.

JAVA API USE

- GUI
- Event Handling
- Exception Handling
- File Management
- String Class
- Date Class

Modular Design & Reusability

- Model/View Separation.
    - All model classes (from UML) are created first without any user interaction.
    - Then view classes are created to call the model objects and methods.
    - In this case study, School class is done, then School GUI creates the interface for I/O, Lecturer class then Lecturer GUI, and so forth.
    - This way you can reuse the model classes for console I/O, web applications or more interface types as you will learn in the future.