MAHARISHI UNIVERSITY of MANAGEMENT

Engaging the Managing Intelligence of Nature

Computer Science Department

CS401 Modern Programming Practices (MPP)

Professor Renuka Mohanraj

© 2015
Maharishi University of Management, Fairfield, Iowa

All rights reserved. No part of this slide presentation may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying or recording, or by any information storage and retrieval system, without permission in writing from Maharishi University of Management.

Lecture 1: The OO Paradigm for Building Software Solutions

Unlocking the Blueprint of Creation

Wholeness Statement

In the OO paradigm of programming, execution of a program involves objects interacting with objects.

Analysis is the process of understanding user requirements and discovering which objects are involved in the problem domain.

Design turns these discovered objects into a web of *classes* from which a fully functioning software system is built.

Each object has a type, which is embodied in a Java *class.*

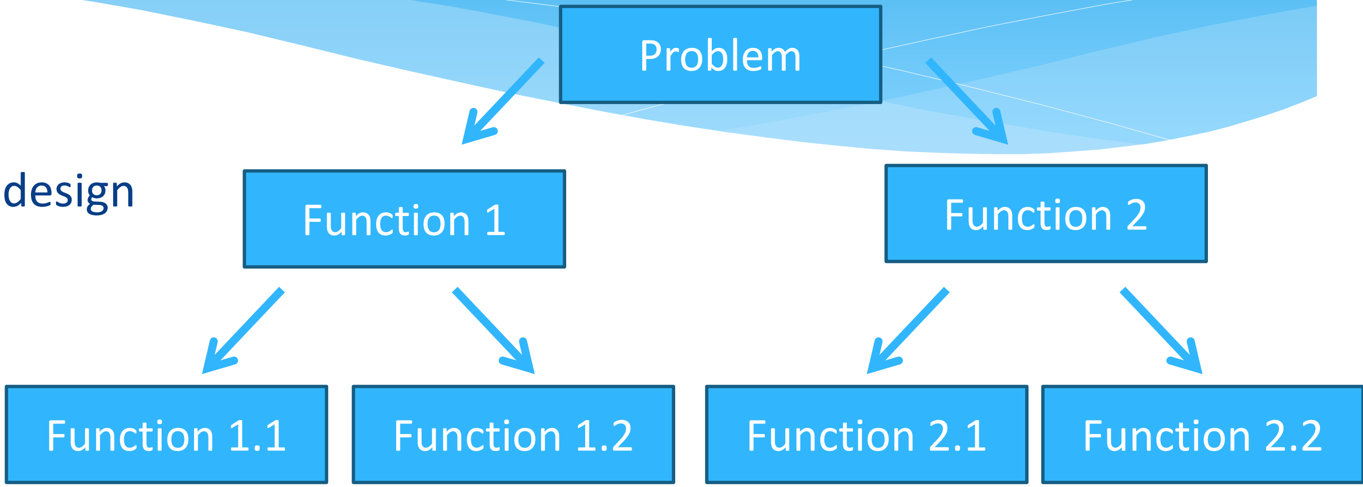
The intelligence underlying the functioning of any software object resides in its underlying class, which is the silent basis for the dynamic behavior of objects.

Likewise, pure consciousness is the silent level of intelligence that underlies all expressions of intelligence in the form of thoughts and actions in life.

Origin of OO

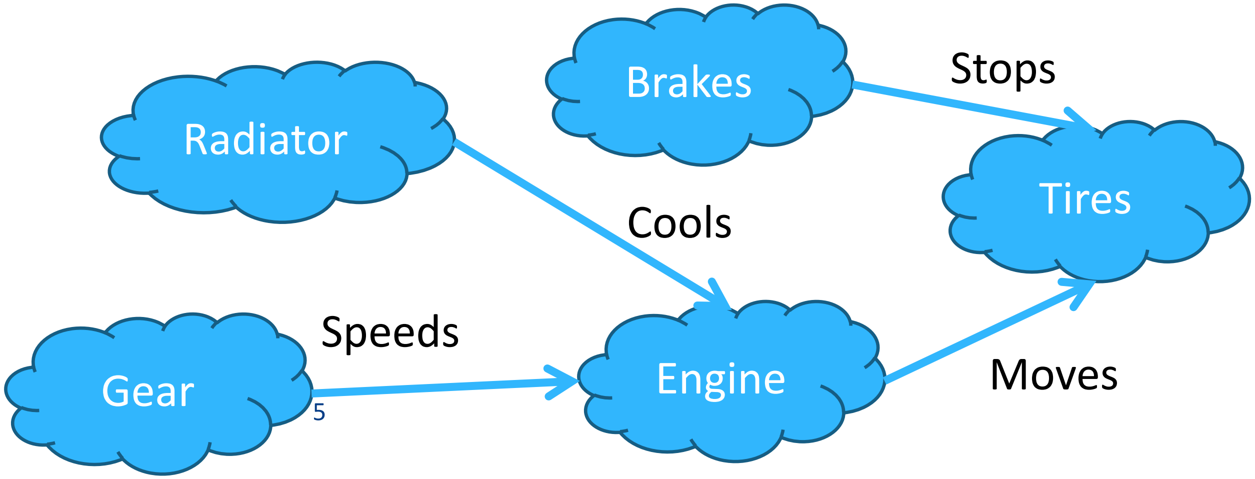
- Procedural analysis & design

In early days, programmers adapted real-world problems into “computer logic“



- OO analysis & design

Using OO, real-world objects are represented by software objects; real-world behavior by sending messages between objects



The Goal

- We want to build a software system based on objects interacting with objects
  Demo: `lesson1.lecture.objectdemo`
- Example: Think of how a car works
- If we can achieve this, there are obvious benefits:
  - Easy to maintain
  - Easy to extend and reuse
  - Easy to understand
- In order to achieve the goal, the OO paradigm sets forth certain principles to follow in design and implementation

Object Oriented Principles

- Objects have *state, behavior,* and *identity*
- Encapsulation and Data Hiding
- Inheritance (Generalization)
- Polymorphism and Late Binding
- Delegation and Propagation (Detail in Lesson-4)

UML Supports OO Principles

UML (Unified Modeling Language) allows us to build a map of “objects-interacting-with-objects”. It provides a language of diagrams for

1. Understanding user requirements (analysis) , where we first discover the “objects” for a new system
2. Developing and communicating a design for building a software solution , where we craft objects carefully so they can work together to create a software system

A UML Diagram Is the Core Element of a Model

- A Model is
  - an abstract description of a system or process
  - simplified representation that enables understanding and/or simulation of the system.
- A model facilitates
  - Description of the problem
  - Description of the solution

UML Models

- Static Model
  - Captures static structure / Characteristics of the system
  - Eg : Class diagram, Object diagram(static view of an interaction)
- Use case Model
  - Describes user requirements. Eg : Use Case diagram
- **Interaction Model**
  - **Scenarios and message flow.**
  - **Eg : Sequence diagram, Communication diagram, Timing diagram**
- Implementation Model
  - Shows working units. Eg : Component diagram
- Deployment Model
  - Process allocation details Eg: Deployment diagram

UML Diagrams

- **Class Diagram –** shows attributes and operations in each of the (primary) classes of the system as well as relationships between them.
- **Sequence Diagram –** shows the flow of communication between the running objects of the system, driven by the use cases of the system (e.g.: In an ATM system, a use case "withdraw money"; a sequence diagram will show how the system processes the request to withdraw money).
- **Object Diagram –** shows how, at a particular moment in time, all the instances of the classes communicate with each other
- **Communication Diagram –** shows the same information as a sequence diagram but arranged in a way to exhibit areas of strong and weak collaboration
- **Activity Diagram** – shows procedural logic, business logic, and workflow; these diagrams are capable of modeling parallel processing
- **Use Case Diagram –** shows in a single diagram all the use cases for the system
- **Component Diagram –** part of the implementation model; shows how well-defined pieces of the system fit together
- **Deployment Diagram –** Shows a system's physical layout, revealing which pieces of software run on what pieces of hardware.

[*See the sample diagrams in the materials accompanying this lecture*]

What we will learn

- Modeling
  - Convert a problem statement to a UML diagram.
  - Types of Diagrams: Class Diagrams, Sequence Diagrams and Object Diagram
  - Convert UML Diagrams to OO code
- Coding
  - Some fundamental programming concepts
  - Best practices for code
- Development of Consciousness
  - Regular practice of TM
  - Connecting CS to SCI and back to CS

Main Point 1

Software is by nature complex, and the only way to manage this complexity is through *abstraction*.

Abstraction is at work when we discover the objects in the problem domain during analysis, and work with these to build a system during design. Abstraction is also at work in creating maps of our objects in the form of UML diagrams.

In a similar way, to manage the complexities of life itself the technique is to saturate awareness with its more abstract levels so that all the details of any situation are appreciated from the broadest perspective. The abstract levels of awareness are experienced in the process of transcending.

Quiz - 1

1. Deriving a new class from an Existing class is called ______________.

a. Encapsulation

b. Inheritance

Object-Oriented Model of a System

- **OO Model should have a one-to-one relationship with a real world problem domain. Generally there are two kinds of UML model: *static* and *dynamic***
- Static Model:
  - Shows the system structure
  - Emphasizes the parts that make up the system
  - classes of objects, including attributes, methods, and structural relationships
- Dynamic Model:
  - It shows the system behaviors
  - details of object collaborations and interactions
- Today we focus on the Static Model – the main example of static model in UML is the *Class Model* with its Class Diagrams.

Models for Analysis and Design

- We begin modeling the system with *analysis – understanding* the problem, the need, the user requirements.
- In this process, we *discover* the classes that will eventually be developed into a fully functioning system.
- Analysis is concerned with *what.*
- Then we move on to *design* – transforming analysis classes into software objects. During design, we are concerned with finding the best way to make the parts that have been identified during analysis function together.
- Design is concerned with *how.*

Analysis Phase: Problem Description

We have been asked to develop an automated Student Registration System (SRS) for the university. This system will enable students to register online for courses each semester, as well as track their progress toward completion of their degree.

When a student first enrolls at the university, he/she uses the SRS to create a plan of study that lists the courses he/she plans on taking to satisfy a particular degree program, and chooses a faculty advisor. The SRS will verify whether or not the proposed plan of study satisfies the requirements of the degree that the student is seeking.

Once a plan of study has been established, then, during the registration period preceding each semester, students are able to view the schedule of classes online, and choose whichever classes they wish to attend, indicating the preferred section (day of the week and time of day) if the class is offered by more than one professor. The SRS will verify whether or not the student has satisfied the necessary prerequisites for each requested course by referring to the student's online transcript of courses completed and grades received (the student may review his/her transcript online at any time).

Assuming that (a) the prerequisites for the requested course(s) are satisfied, (b) the course(s) meet(s) one of the student's plan of study requirements, and (c) there is room available in each of the class(es), the student is enrolled in the class(es).

If (a) and (b) are satisfied, but (c) is not, the student is placed on a first-come, first-served wait list. If a class/section that he/she was previously waitlisted for becomes available (either because some other student has dropped the class or because the seating capacity for the class has been increased), the student is automatically enrolled in the waitlisted class, and an email message to that effect is sent to the student. It is the student's responsibility to drop the class if it is no longer desired; otherwise, he/she will be billed for the course.

Students may drop a class up to the end of the first week of the semester in which the class is being taught.

Identifying Noun phrases

Create a list of all the *noun phrases* from the problem description.

Examples:

- student
- plan of study
- wait list

Problem Description

We have been asked to develop an automated Student Registration System (SRS) for the university. This system will enable students to register online for courses each semester, as well as track their progress toward completion of their degree.

When a student first enrolls at the university, he/she uses the SRS to set forth a plan of study as to which courses he/she plans on taking to satisfy a particular degree program, and chooses a faculty advisor. The SRS will verify whether or not the proposed plan of study satisfies the requirements of the degree that the student is seeking.

Once a plan of study has been established, then, during the registration period preceding each semester, students are able to view the schedule of classes online, and choose whichever classes they wish to attend, indicating the preferred section (day of the week and time of day) if the class is offered by more than one professor. The SRS will verify whether or not the student has satisfied the necessary prerequisites for each requested course by referring to the student's online transcript of courses completed and grades received (the student may review his/her transcript online at any time).

Assuming that (a) the prerequisites for the requested course(s) are satisfied, (b) the course(s) meet(s) one of the student's plan of study requirements, and (c) there is room available in each of the class(es), the student is enrolled in the class(es).

If (a) and (b) are satisfied, but (c) is not, the student is placed on a first-come, first-served wait list. If a class/section that he/she was previously waitlisted for becomes available (either because some other student has dropped the class or because the seating capacity for the class has been increased), the student is automatically enrolled in the waitlisted class, and an email message to that effect is sent to the student. It is the student's responsibility to drop the class if it is no longer desired; otherwise, he/she will be billed for the course.

Students may drop a class up to the end of the first week of the semester in which the class is being taught.

Step 1 : List of Noun & Noun Phrases

system
students
courses
semester
progress
completion
degree
student
university
plan of study
courses
degree program
faculty advisor
plan of study
requirements of degree
student
plan of study
registration period

semester
students
schedule of classes
classes
preferred section
day of the week
time of day
class
professor
student
prerequisites
requested course
student
transcript
courses completed
grades received
student
transcript

student
waitlisted class
email message
student
responsibility
class
course
Students
class
end

NOTES:
1. Many duplicates
2. Prefer singular to plural ("student" instead of "students")

Step 2: Sort and Eliminate Duplicates

class
class/section that he/she was previously wait-listed for student
completion
course
courses completed
day of the week
degree
degree program
email message
end
faculty advisor
first-come, first served wait list
grades received
plan of study
plan of study requirements
preferred section
prerequisites

(continued)

professor
progress
registration period
requested course
requirements of degree
responsibility
room
schedule of classes
seating capacity
semester
student
system
time of day
transcript
university
waitlisted class

Step 3 : Streamline the List Further

- Eliminate terms that do not seem to be objects
- such as: ‘completion’, ‘end’, ‘progress’, ‘responsibility’, ‘registration period’ and ‘requirements of the degree’. (Most of these indicate *relationships*; ‘requirements’ will be wrapped into ‘plan of study requirements’.)
- Eliminate reference to the system itself (SRS). Eg: “university” our system will (probably) not need to maintain/modify information about the university itself.
- Retain list of eliminated terms, so you can use them later if necessary.

Final List of Noun Phrases

class
class/section that he/she was
previously wait-listed for
course
courses completed
day of the week
degree
degree program
email message
faculty advisor
first-come, firstserved wait list
grades received
plan of study
plan of study requirements
preferred section
prerequisites
professor
requested course
room
schedule of classes
seating capacity
section
semester
student
system
time of day
transcript
waitlisted class

Main Point 2

The OO approach to building software solutions is to represent objects and behavior in the problem domain with software objects and behavior.

One of the first steps in this process is to *locate* the objects implicit in the problem statement, and this is done by examining *nouns* and *noun phrases* in the problem statement.

These words and phrases link the real world situation to the abstract realm of software objects.

Likewise, linking individual awareness to its abstract foundation in fully expanded awareness is the basis for creating solutions to the real-world challenges of life.

Quiz - 2

1. Class diagram is an example of dynamic model. This statement is _______.

a. true b. false

Problem Description – In class Exercise, continued

The next step is to group together terms that are closely related, that belong together, that can be classified with a single concept.

- Example: *class, course, waitlisted class* belong together
- Note: Sometimes this step requires the assistance of a domain expert because sometimes there is a need to discriminate between subtle shades of meaning

Step 4 : Group "Synonyms"

We need to group apparent synonyms, to choose the one designation from among each group of synonyms that is best suited to serve as a class name.

- class
- course
- waitlisted class
- class/section that he/she was previously wait-listed for
- preferred section
- requested course
- section
- prerequisites

- day of the week

- degree
- degree program

- email message

- faculty advisor
- professor

- first-come, first served wait list

- plan of study
- plan of study requirements

- room
- schedule of classes
- seating capacity
- semester
- student
- system
- time of day

- courses completed
- grades received
- transcript

Choose Class Name

class
course
waitlisted class
class/section that he/she was
previously wait-listed for
preferred section
requested course
section
prerequisites

- Avoid choosing nouns that imply roles between objects. For example, “prerequisite” is a role in an association between two courses. “Waitlisted class” is a role in an association between a student and a course. “Preferred section” is a role in an association between a student and a course.

- Prefer shorter expressions to longer ones (“degree” instead of “degree program”)

(continued)

day of the week

degree
degree program

email message

faculty advisor
professor

first-come, firstserved wait list

plan of study
plan of study requirements

room
schedule of classes
seating capacity
semester
student
system
time of day

courses completed
grades received
transcript

(continued)

courses cmpleted
grades received
transcript

The notion of “transcript” *includes* “courses completed” and “grades received” although they are not actually synonyms.

Tests for a Class

- Is the class well defined? [If not, it may be a combination of concepts that could form separate classes]
- Are there any attributes for this class? [Sometimes, can think of attributes for a class but they are not relevant. Example in SRS: “University”]
- Are there any services that would be expected of objects in this class? [Typically, a class will provide services, which in Java are represented by its public methods.]
- Can this item simply be included as an attribute of another class?
  [Example: Should “room” be a class on its own, or an attribute of “section”? Which others can we treat as just attributes?]

Examples of Attributes instead of Classes

- Day of week
- Degree
- Seating capacity
- Semester
- Time of Day

Types of Classes

- Domain Classes: abstractions that the end user will recognize and that represent real-world entities.
- Implementation Classes: introduced solely behind the scenes to hold the application together (example: a dictionary to look up students based on ID number, or a special type of list to keep track of professors).
  Note: We can think of “email message” as an implementation class.
- Retain only Domain Classes; the others will be useful during design.

Final List of Classes

course
plan of study
professor
section
student
transcript

Data Dictionary of Classes

- **Course**: a semester-long series of lectures, assignments, exams, etc. that all relate to a particular subject area, and which are typically associated with a particular number of credit hours; a unit of study toward a degree. For example, ‘Software Engineering’ is a required **course** for the Master of Science Degree in Computer Science.
- **Plan of Study**: a list of the **courses** that a student intends to take to fulfill the **course** requirements for a particular degree.

(continued)

- **Professor**: a member of the faculty who teaches **sections** and/or advises **students**.
- **Section**: the offering of a particular **course** during a particular semester on a particular day of the week and at a particular time of day (for example, **course** ‘Software Engineering' is taught in the Spring 2016 semester on Mondays from 1:00 – 3:00 PM).
- **Student**: a person who is currently enrolled at the university and who is eligible to register for one or more **sections**.

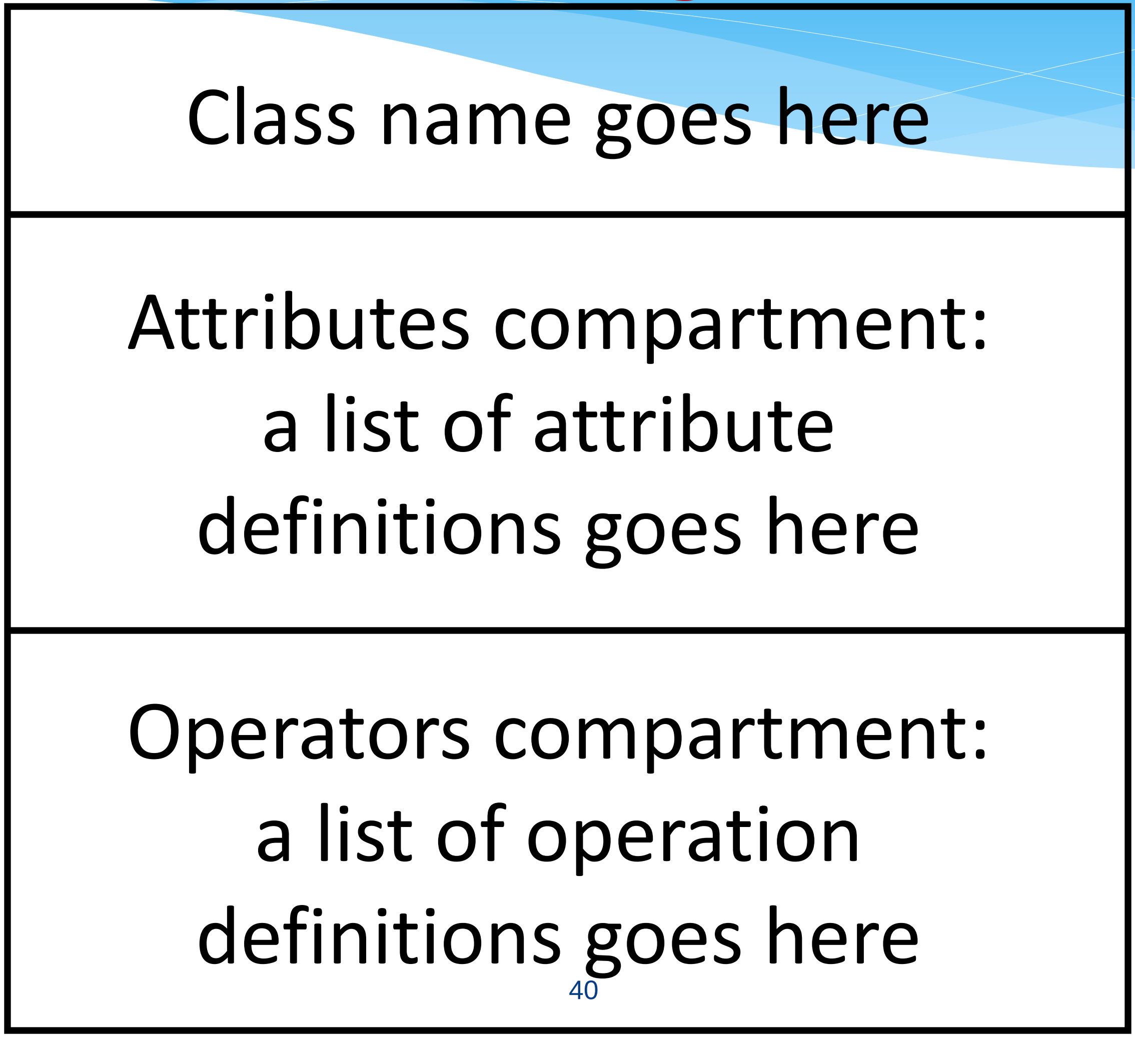
(continued)

- **Transcript**: a record of all of the **courses** taken to date by a particular **student** at this university, including which semester each **course** was taken in, the grade received, and the credits granted for the **course**, as well as reflecting an overall total number of credits earned and the **student's** grade point average (GPA).

The Class Model

- Continuing our study of the problem description and other inputs, we now specify more detail in our classes – the data and behavior that each will have – and (in later lessons) specify relationships between them
- UML makes it convenient to take this next step
- When considering such details leads us to introduce classes, attributes, operations, relationships that go beyond simply modeling the user requirements, and are concerned with *how to build the system,* we will be in the *design phase* of the project.

Unified Modeling Language (UML) Class Diagram



Class name goes here

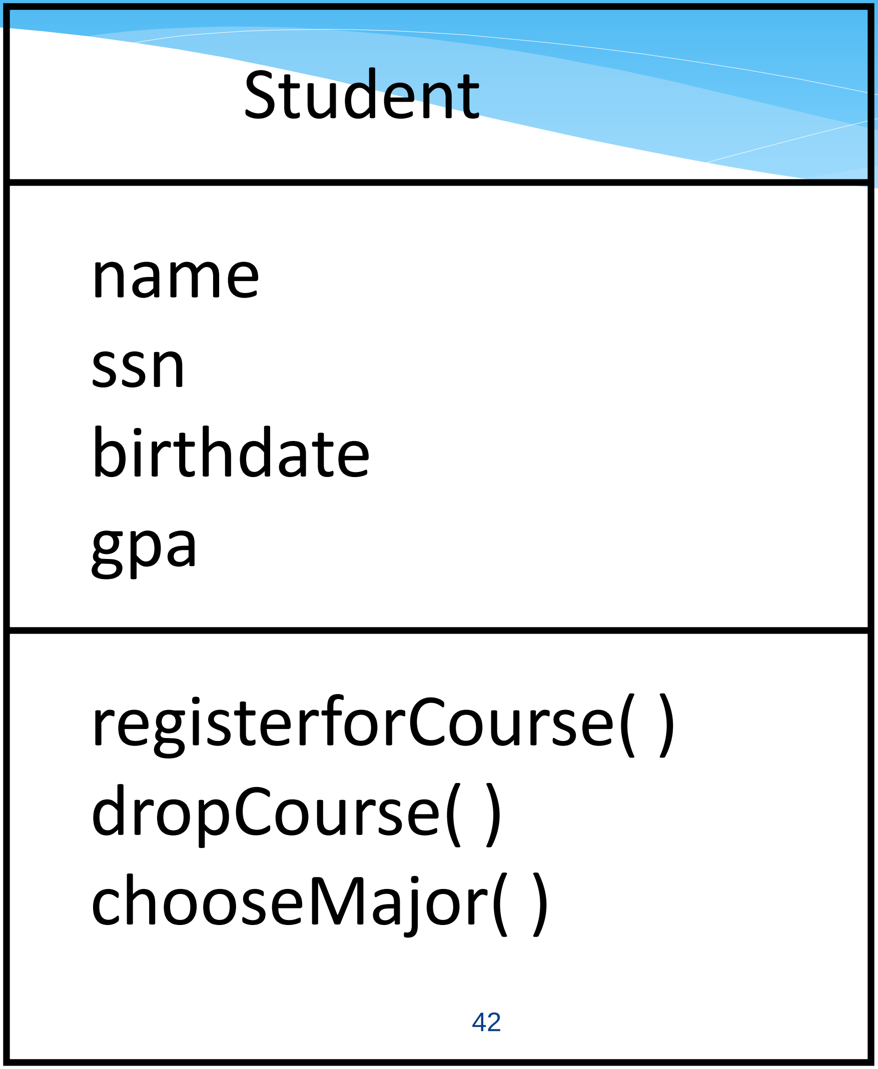
Attributes compartment: a list of attribute definitions goes here

Operators compartment: a list of operation definitions goes here

Modeling – In class Exercise

- Create a UML class for Student.
- Look back at our problem description and your group's definition of a student.
  - What attributes do we need?
  - What operations do we need?

Student Class Diagram



Student

name
ssn
birthdate
gpa

registerforCourse( )
dropCourse( )
chooseMajor( )

Identifying Attributes

- Use requirements to find attributes of domain classes
- Use your prior knowledge of the domain to help find attributes (e.g. each student has an ID number)
- Talk to the domain expert (often you're not the expert)
- Examine old SRS system already in use to find attributes
- Note: Trying to understand domain classes in this way is part of the process of *analysis.*

Making Items Attributes

- The following items can all be included as attributes of the “Section” class: day of the week, semester, time of the day, room, seating capacity.
- The item “major” could be another attribute of Student class.

Identifying Operations

- Most operations are added during design when looking at implementing specific functionality
- Some operations that are added earlier are often just computed attributes. E.g. getAge() when a class has a birthdate attribute
- It will be easier to identify operations after we introduce *relationships* between classes, represented in UML as *associations.*

Main Point 3

A class encapsulates *data,* stored as attributes, and *behavior*, represented as operations. These are the static and dynamic aspects of any class, and a UML diagram for a class provides compartments for each of these.

These two aspects of a class – data and behavior – are aspects of anything the we encounter in creation. They give expression to the reality that life, at its basis, is a field of *existence* and *intelligence.*

Connecting the Parts of Knowledge With the Wholeness of Knowledge

1. Class diagrams display the data and behaviors of a class
2. Class diagrams provide an (abstract) representation of a specific real word problem domain.

---

3. **Transcendental Consciousness** is the simplest state of awareness, where the mind goes beyond thoughts and concepts to the most abstract level of awareness.
4. **Wholeness moving within itself**: in Unity Consciousness one experiences that all objects in the universe arise from consciousness and are ultimately nothing but consciousness.