
MAHARISHI UNIVERSITY of MANAGEMENT
Engaging the Managing Intelligence of Nature
Computer Science Department
CS401 Modern Programming Practices (MPP)
Professor Joe Bruen

Lecture 1: Intro

Course Overview

Review of OO Principles

Course Objectives

Unlocking our full potential

- Problem solving through OO analysis and design.
- Understanding the OOP paradigm.
- Gain proficiency in OOP using Java.
- Explore and create UML diagrams.
- Use UML diagrams to create Java code.
- Learn to work alone and as a group.
- Develop presentation skills.
- Continue developing your full potential through regular TM practice.

Be Professional:

- Be on time
- Do not bring any food or drink into the classroom
- No jeans or T-Shirts

Course Policies and Procedures

- Review Syllabus and Calendar in our Sakai Course Management System.

  https://online.cs.mum.edu/

- Course Lectures will be posted under Resources in Sakai after each lecture.
- Note emphasis on being professional in attendance, dress, and class participation.

- Note Academic Honesty Policy

Project Development and Demos

- Each group will share their design and implementation of their project with demos.
- Most of us learn:
  - 10% of what we read
  - 20% of what we hear
  - 30% of what we see
  - 50% of what we see and hear
  - 70% of what we talk over with others
  - 80% of what we use and do in a job
  - 90% of what we teach someone else
- Placement reports that presentations are great interview preparation.

Lab Assignments

- The purpose of all labs and homework is to provide an opportunity for learning with a minimum pressure for graded outcomes.
- For the purpose of learning, all students should attempt their part of the lab individually, but are also encouraged to work in pairs or small groups.
- Discussions among fellow students are very beneficial to the learning process in order to clarify ones own understanding and to remove any road blocks that may slow down progress.

Professional Etiquette

Consider MUM to be your employer, and your professor to be your technical manager. Failure to attend class, to be punctual, or to dress professionally can result in reduced grade, dismissal from class, or outright failure. If you are sick and cannot attend class send an email just like you would do for work (generally before class.)

Courtesy during group meditation is a part of professional etiquette. No typing, texting, reading, or talking during meditation. We take 2 minutes of silence after meditation before starting activity. Everyone is expected to allow their neighbors this 2 minutes of silence.

Wholeness Statement

Object Oriented technology offers support for abstraction at conceptual level.

The hierarchical organization, along with the ability to modify and reuse enable us to manage the increasing complexity of software development.

By using more and more of the intelligence of Nature, we are able to successfully manage all complexities in life, and live a life of success, harmony and fulfillment.

Core Theory of OO

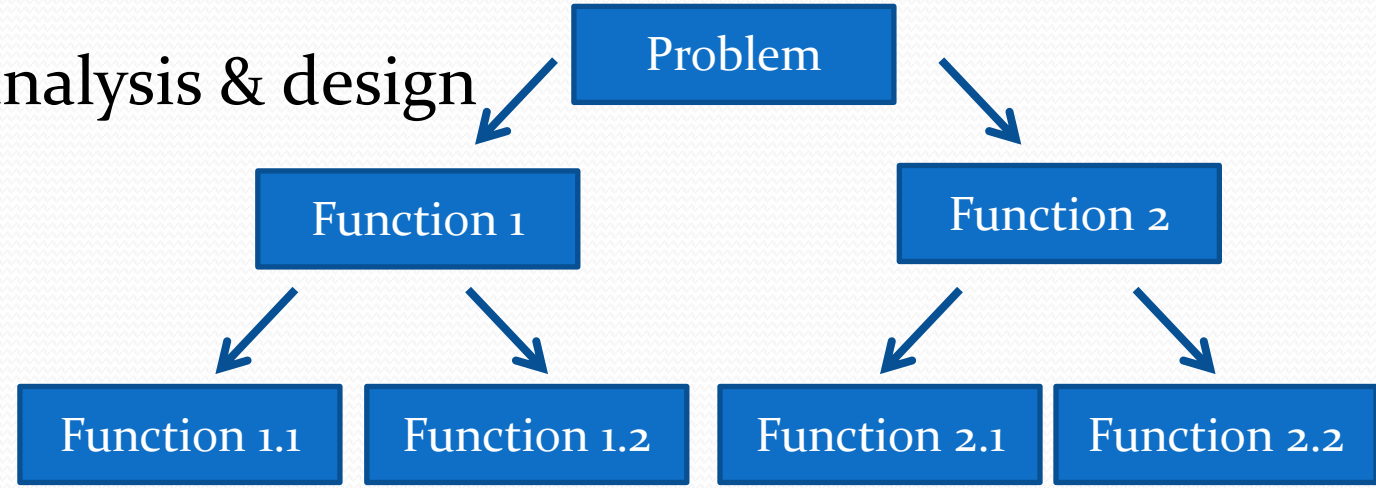
What is Object Oriented, why do we use it?

In this course:

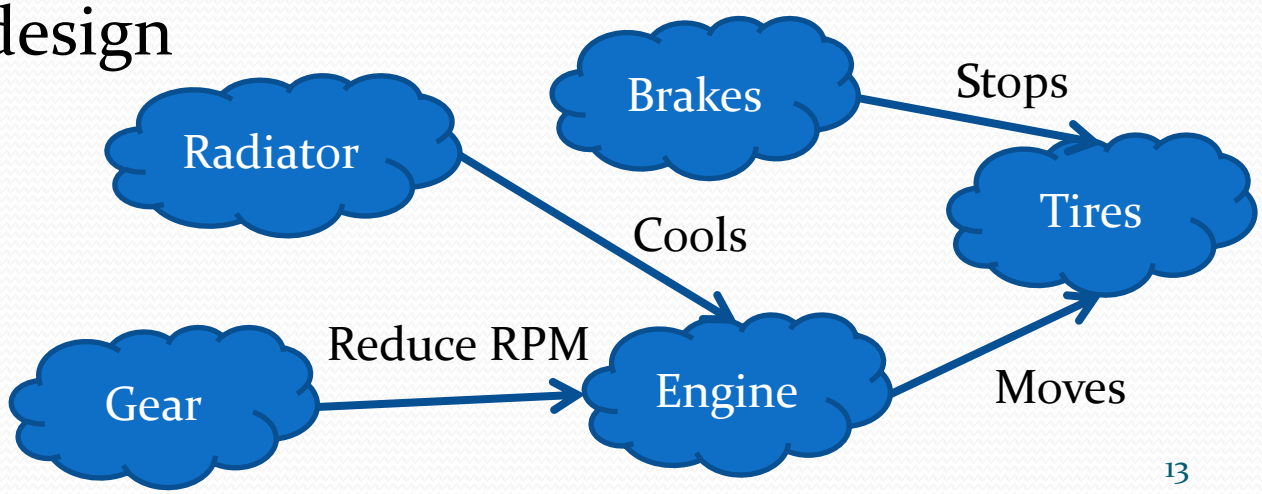
Section	Focus
Analysis	What?
Design & Implementation	How?

Origin of OO

- Procedural analysis & design



- OO analysis & design



Main Point 1

The OO paradigm is based on the assumption that there is order and intelligence in real world systems, which can be abstracted into a model of objects that send messages to each other.

OO does not address the problem on the level of the problem, but instead aims to model the underlying reality of the situation, from which the solution will naturally arise.

Unity in Diversity in Nature

- A uniform system may look nice, but may not be so helpful.
  - If all the fruits taste the same, what will a person do if that is not the taste she prefers ☹
- Nature has diversity everywhere, and that is what makes it a perfect home for everyone and everything.
- We all see the diversity in nature, but we also know there is a unity… right?

Unity in Diversity in Software

- A good system must have diversities within unity to produce a highly coherent system.
- Building software involves a series of 'divide and reunite' iterations
  - Decompose 'divide' to understand "Analysis"
  - Compose 'reunite' to build "Design"
- Your code consists of Diverse objects, but in an object oriented world we must have a unity some place, why? How?

Software Description

- How would you describe a software system?
  - How do engineers describe a building before building it?
  - Using models
    - Booch
    - Rumbaugh
    - Jacobson
    - …
  - UML (Unified Modeling Language)
- UML features and benefits

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 must facilitate
  - Description of the problem
  - Description of the solution

UML Models

- Static Model — Captures static structure
- State Model — Dynamic behavior of objects
- Use case Model — Describes user requirements
- Interaction Model — Scenarios and message flow
- Implementation Model — Shows working units
- Deployment Model — Process allocation details

UML Diagrams

- Class Diagram
- Sequence Diagram
- Object Diagram
- Collaboration Diagram
- Statechart Diagram
- Activity Diagram
- Use Case Diagram
- Component Diagram
- Deployment Diagram

What we will learn

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

Main Point 2

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

We understand and document (model) the underlying principles (classes) instead of the many complex and diverse instances (objects) that the principles bring forth.

Every person can manage all complexity by transcending to the most abstract field, which manages all complexity in nature.

More Applied OO

Reviewing of the Applied Principles of OO

Object Oriented Principles

- Encapsulation
- Information Hiding
- Inheritance
- Generalization without Polymorphism
- Generalization with Polymorphism
  - Method overriding
  - Method overloading
- Delegation
- Messaging
- Late Binding

Object Oriented Programming

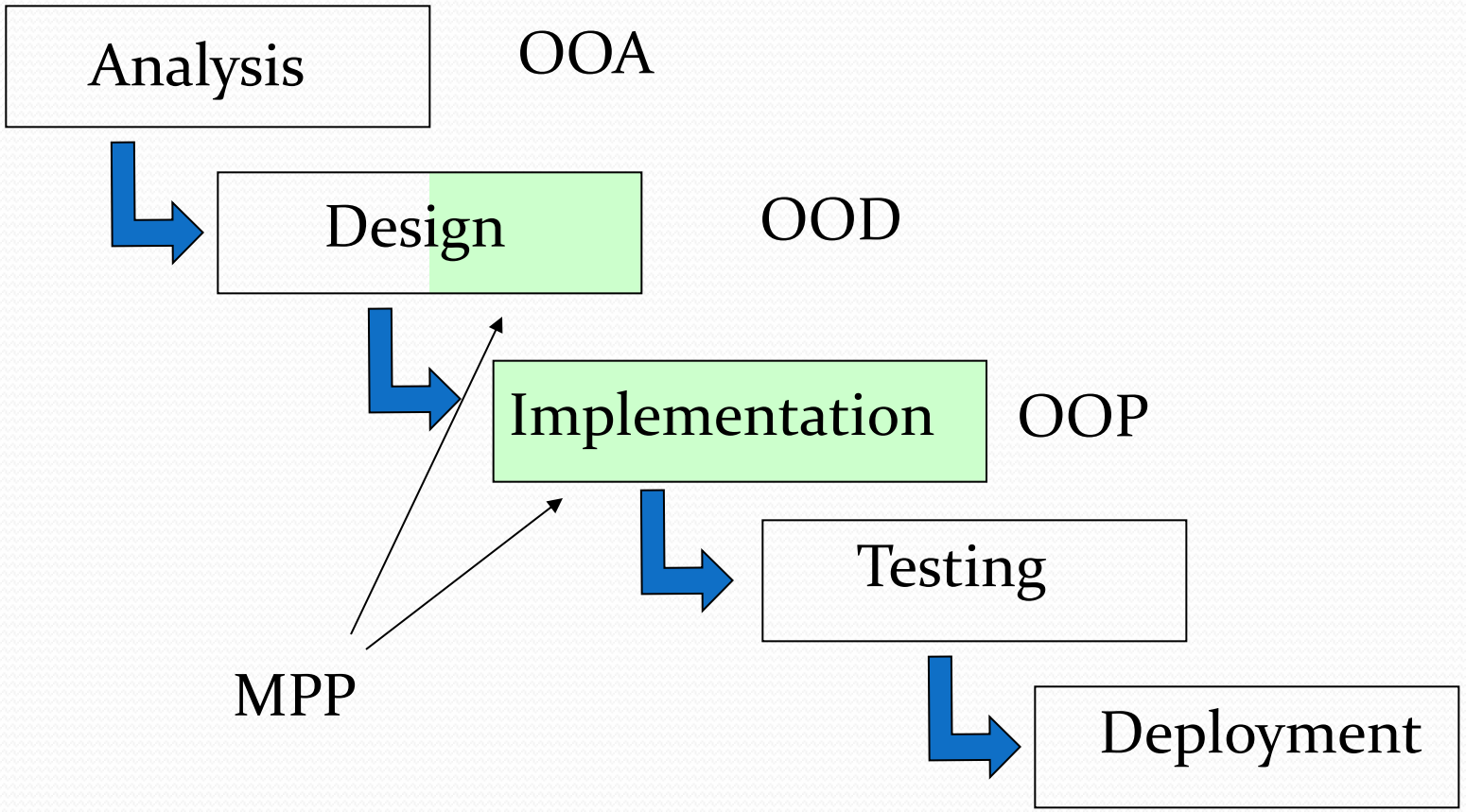
- OOP is a solution to the problem.
- Idea
  - Model the system as objects interacting with each other through messages.
  - Encapsulate object properties & behaviors. Hide implementation details.
  - Abstract object behaviors & properties.
- Benefits
  - Stable
  - Iterative construction
  - Better maintainability
  - Better reusability

Key points of OOP

- Managing complex software.
- Classes
  - Adding new data type
  - Object blueprint
  - Encapsulation
    - State
    - Behavior
  - Information Hiding
    - Hide properties
    - Hide implementation
  - Communicate through Message Passing
- Inheritance
  - Mechanism for extension and reusability.
- Polymorphism
  - Mechanism for sharing abstractions.

OO - the big picture

The different techniques in every phase are always in terms of the real world objects.



In Class Exercise – Lesson 1-1

- Take 15 minutes with a partner and try to come up with a model for the Student Registration System described in the following problem description.

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.

Objects - 1

- An instance of a class.
- Characteristics
  - State
    - The internal value of instance fields
    - Determines the reaction to messages it receives (invoked methods)
    - Determines the data stored after messages (change of state)
  - Behavior
    - What actions can an object perform, what methods can you invoke on it, what messages you can send it.
  - Identity
    - What makes this object different from its peers.

Objects - 2

- Object & Object Variable
- Date deadline; //doesn't refer to any object
- Initialization
  - deadline = new Date(); //construct then init
  - deadline = birthday; //init
- new Date()
  - Creates a Date object, then returns a reference to it.
- When no reference to an object exists, it is put on the garbage collection list, the memory it is occupying can be declared as free. (In Java, garbage collection is automatic)
- If object variable refers to null, that means there is no object it is referencing.
- An object may have more than one object variable referencing it.

Main Point 3

Objects are instances of classes. Similarly everything in nature is an instance on underlying concepts and principles.

Once we understand gravity we can predict that an object will drop faster on earth than it will on the moon.

By understanding the classes that make up a system, and how they relate, we understand what (state) the objects of that system represent, and how they interact (behavior) to accomplish tasks.

Abstraction

- OOP abstracts real-world concepts.
- Layers of abstraction

Encapsulation

- Data (state)
- Operations (methods, behavior)

Information Hiding

- Class must not provide access to its instance fields.
- Class must not provide implementation details.
- How to get data?
  - Use Accessor methods (getters)
- How to change data?
  - Use Mutator methods (setters)
- Implicit & Explicit Parameters
  - Method operates on objects, access instance fields.
  - Instance of the object “this” is implicit.
  - Parameters are explicit.

Constructors

- Creates an instance of the class and returns the reference.
- Have the same name as the class.
- A class can have more than one constructor.
- If a class has no constructors, JVM creates a default constructor for you that takes no parameters.
- can take zero, one or more parameters.
- Have no return value.
- Called with the new operator.
- Constructor call constructor
  - It must be the first method call
  - this(…)

Methods

- Access modifier.
- Static methods.
  - When a method doesn't need to access object state.
  - When a method only needs access to static fields.
  - Factory methods.
    - Methods that return a sub type of a type.
  - What is good about static methods?
- Parameters
  - Call by value
  - Call by reference
  - Java uses a call by value
- Overloading

Methods

- What is good about static methods? – You do not need to new() a class to use static methods.

- Can you think of some obvious method calls we are going to need for our student registration system?

Scope

- If not stated ends up being package.
- For Classes it may be good
- For attributes, be careful, why?

Scope

- If not stated ends up being package.
- For Classes it may be good
- For attributes, be careful, why?
  - You are violating information hiding.

Guidelines

- Always keep data private.
  - Add accessors & mutators if needed.
- Always initialize data.
  - Instance field are initialized but local variables are not.
- Don't use too many basic types in a class
  - Replace with a new Class
- Not all fields need accessors & mutators
  - Some fields may be set once, or never read.
- Use a standard
  - Class
    - Public features
    - Package features
    - Private features
  - Section
    - Static fields
    - Instance fields
    - Static methods
    - Instance methods
- Break up classes with too many responsibilities
  - The 10 minute rule
- Make names of classes and methods reflect responsibility
  - Noun for class name

Main Point 4

Information hiding allows us to hide the underlying reality of how an object is implemented.

This is useful because it allows us to create separate layers of functionality. A piece of code is not dependent on how another piece of code is implemented as long as it provides the needed features (greater flexibility).

The most extreme example of this in life, is where the total diversity of existence completely hides the underlying reality that it is all implemented from one and the same underlying source, the unified field.

Summary

Today we looked at the principles of OO programming

- We saw that OO is meant to model the real world. We do not focus on 'how do I solve this problem' but rather 'how can I model what is really going on'.
- We saw that we can understand complex systems through abstraction, by focusing on underlying principles rather than the things they produce
- We saw that Objects are instances of Classes, where classes can be thought of as the underlying concepts / principles that can manifest themselves
- We saw that information hiding is used to hide implementation details, thereby creating greater flexibility

Connecting the Parts of Knowledge With the Wholeness of Knowledge

1. In all phases of Object-Oriented development the program is modeled in terms of the real world problem domain.

2. OOP supports abstraction, hierarchy, change and reuse, which make it an excellent technology for developing complex software.

---

3. **Transcendental Consciousness** is the field of solutions for all problems.

4. **Wholeness moving within itself**: Unity Consciousness is the most simple and blissful state of awareness, where one experiences that all the complexity of the world is nothing other than an expression of one's own Self.