# UML Overview

Dr. Isaac Griffith Idaho State University

# Outcomes

Idaho State University | Computer Science

After today's lecture you will:

- Have and understanding of the different types of UML Diagrams

# A Reminder

Idaho State University | Computer Science


### Central role of Objects
- Objects are first-class
- Data and operations
- Processes rely on Objects

### Notion of a class
- Classes **type** Objects
- Categorize, classify, and define hierarchies

### Abstract Spec of Function
- **Interface -** Abstract specs that do not restrict function
- **Abstract classes**

### Standard Solutions
- **Design Patterns**
- All phases of SDLC

### A Language to define the System
- **UML**
- Similar to Blueprints

### Analysis process to model a system
- Specs → **Conceptual Design**
- Conceptual Classes → **Implementation Classes**

### Notions of Extendability and Adaptability
- Extendability → **Inheritance**
- Adaptability → **Composition**

# Introducing UML

## CS 2263

# Describing OO Systems

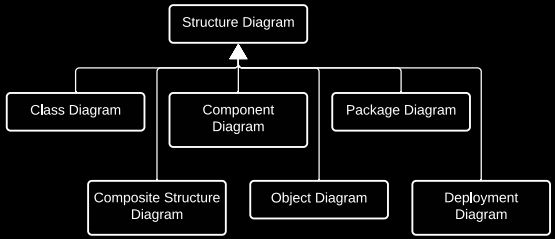
Idaho State University | Computer Science

- The Unified Modeling Language (UML) is a standard for documenting OO systems
- UML provides a graphical notation for documenting the artifacts (classes, objects, and packages) of an OO system.
- UML diagrams can be divided into 3 categories
  - **Structural Diagrams**
  - **Behavior Diagrams**
  - **Interaction Diagrams**

# Structural Diagrams

Idaho State University | Computer Science

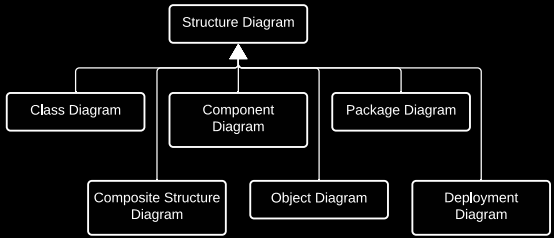
- Show the static architecture of the system irrespective of time.
- Structural Diagrams may be any of the following:
  - **Class Diagrams** - shows classes, methods and fields
  - **Composite Structure Diagrams** - provides a means for presenting the details of a structural element (i.e., a class)
  - **Component Diagrams** - Shows the details of components (software entities that satisfy functional requirements)



# Structural Diagrams



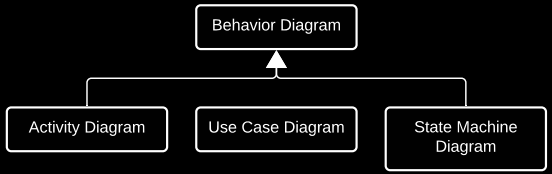
- Show the static architecture of the system irrespective of time.
- Structural Diagrams may be any of the following:
  - **Deployment Diagrams** - Shows the assignment of executable files to computing elements and the communication between entities.
  - **Object Diagrams** - Shows how objects related at runtime
  - **Package Diagrams** - Shows packages and the dependencies between them.



# Behavior Diagrams



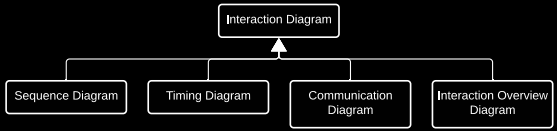
- Depict the behavior of a system or business process
- Behavior diagrams may be any of the following:
  - **Activity Diagrams** - similar to a flowchart and shows the events of an activity
  - **Use Case Diagrams** - shows the interaction involved in a use case.
  - **State Machine Diagrams** - shows the sequence of states that an object goes through in its lifetime



# Interaction Diagrams

Idaho State University | Computer Science

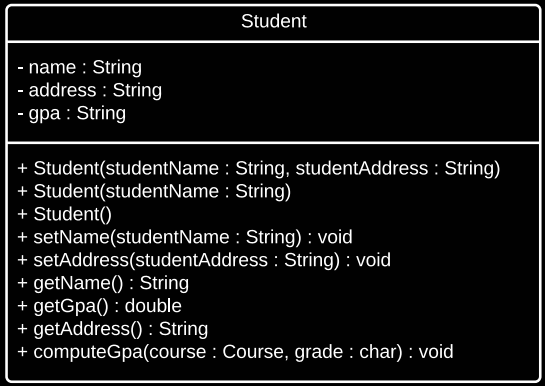
- Show the methods, interactions, and activities of the objects
- Interaction diagrams may be any of the following:
  - **Sequence Diagrams** - details how operations are carried out
  - **Timing Diagrams** - shows an object's change in state over time as it reacts to events
  - **Communication Diagrams** - has the same purpose as a sequence diagram, but with a different layout
  - **Interaction Overview Diagrams** - shows the high-level control flow in a system



# Class Diagrams

Idaho State University | Computer Science

- Each class is represented by a box or a box divided into three sections.
  - The top section displays the name of the class
  - The middle section displays the attributes (fields, properties)
  - The bottom section displays the operations (methods, functions)
- Attributes are defined as follows:
  - an access modifier (+, –, #, ~)
  - attribute name
  - colon (:) and
  - attribute type



# Class Diagrams

Idaho State University | Computer Science

- Methods are defined as follows:
  - an access modifier (+, –, #, ~)
  - method name
  - parameter list in parentheses
    - a comma-separated list of:
      `name : type`
  - colon (:) and
  - return type
- Access Modifiers

  \+ - public

  \# - protected

  ~ - package/default

  – - private

| Student |
| --- |
| - name : String<br>- address : String<br>- gpa : String |
| + Student(studentName : String, studentAddress : String)<br>+ Student(studentName : String)<br>+ Student()<br>+ setName(studentName : String) : void<br>+ setAddress(studentAddress : String) : void<br>+ getName() : String<br>+ getGpa() : double<br>+ getAddress() : String<br>+ computeGpa(course : Course, grade : char) : void |

# Associations

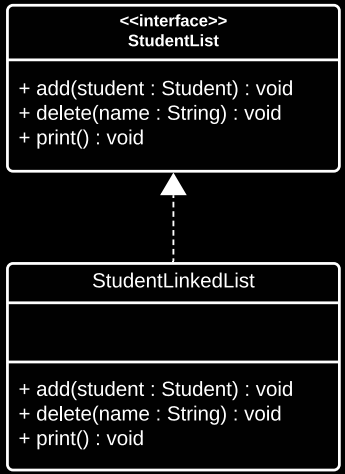
Idaho State University | Computer Science

- Associations depict some relationship between two classes
  - The example shows that the `Accounts` instance contains zero or more `BankAccount` instances
  - The numbers are called the multiplicity and indicate that the opposite side is connected to the number and number's side class.
    - That is each `BankAccount` is related to exactly 1 `Accounts`
  - The most basic association is simply a solid line with no arrows (indicating bidirectionality)

# Interfaces and Implementations



- Class diagrams can also depict interfaces and their realizations
- These relationships are depicted by **a dashed line with a white triangular arrowhead** point towards the interface.
- For example:
  - We show that the `StudentLinkedList` implements the `StudentList` interface
  - It should also be noted that the interface's operations are assumed to be **abstract**
  - Thus, the implementing class (`StudentLinkedList`) should also have the same methods



# Use Cases

Idaho State University | Computer Science

- Constructed during the Analysis Phase
- Describes a feature of an application system
  - Describes the interaction between an **actor** (human, software, or hardware)
  - Does **not** describe **how** the system carries out the task
- Uses cases may be textually describe in a table with two columns
  - First column describes what the actor does
  - Second column describes the system's response
- The use case does not depict all possible situations, but rather only the **main flow**

# ATM System Use Case

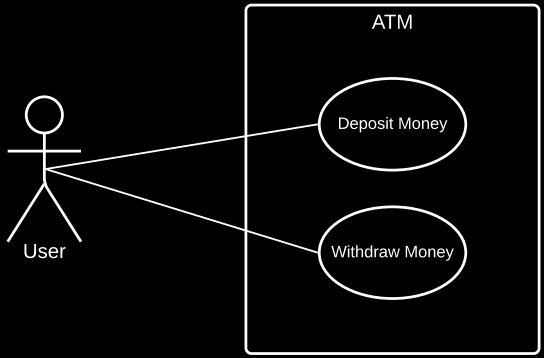
Idaho State University | Computer Science

| | Action performed by the actor | | Responses from the system |
|---|---|---|---|
| 1 | Inserts debit card into the 'Insert card' slot | | |
| | | 2 | Asks for the PIN number |
| 3 | Enters the PIN number | | |
| | | 4 | Verifies the PIN. If the PIN is invalid displays an error and goes to Step 8. Otherwise, asks for the amount |
| 5 | Enters the amount | | |
| | | 6 | Verifies that the amount can be withdrawn. If not, display an error and goes to Step 8. Otherwise, dispenses the amount and updates the balance |
| 7 | Takes the cash | | |
| | | 8 | Ejects the card |
| 9 | Takes the card | | |

# Use Case Diagrams

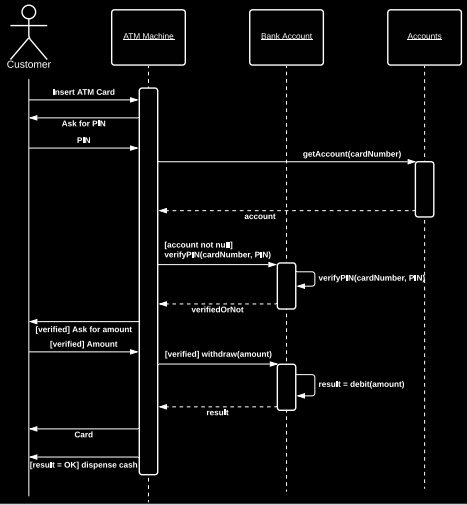
Idaho State University | Computer Science

- A Visual Diagram depicting the use cases of a system and their relationships.
- A Use Case Diagram is composed of the following components
  - A rectangle with a system name depicting the system boundaries
  - Stick figures representing actors
    - Actors to the left, **primary actors**, directly start interactions with the system
    - Actors to the right, **secondary actors**, react to system events
    - Actors, being separate from the system, must be outside the system boundary
- Lines between actors and use cases indicate association.



# Sequence Diagrams

Idaho State University | Computer Science



# UML Tools



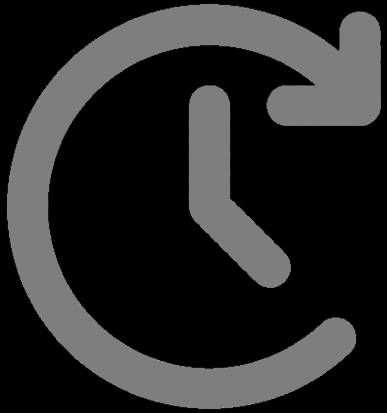
- Violet UML (free)
  - http://horstmann.com/violet/
- StarUML (semi-free)
  - http://staruml.io
- LucidCharts (free for student use)
  - http://lucidcharts.com
- Rational Rose
  - http://www.rational.com/
- There are many others, but most are commercial


CS 2263

# For Next Time



- Review Chapters 2.7
- Review this lecture
- Come to class
- Read Chapter 3
- Continue working on Homework 02



# Are there any questions?