CMSC 132:
OBJECT-ORIENTED PROGRAMMING II

UML (Unified Modeling Language)

Department of Computer Science
University of Maryland, College Park
UML (Unified Modeling Language)

- UML is a modeling language for object-oriented software that allow us to specify, visualize, construct, and document systems
- Use UML to help
  - Visualize design of software
  - Provide abstract model of software
- UML provides a number of diagrams that
  - Describe a model of all or part of system
  - From a particular point of view
  - With varying level of abstraction
UML Diagrams

- UML provides a number of diagrams that
  - Describe a model of all or part of system
  - From a particular point of view
  - With varying level of abstraction
  - Using certain set of notations
Class Diagrams

• We want to use **class diagrams** to describe our designs

• **Class diagram**
  • Represents (static) structure of system
  • It displays
    • Information for class
    • Relationships between classes
Class diagrams represent the structure of a system. They visually depict the various classes, their attributes, operations, and the associations between them. In the diagram:

- **Class**: SimpleWatch, LCDDisplay, Battery, Time
- **Attributes**: BlinkIdx, blinkSeconds(), blinkMinutes(), blinkHours(), stopBlinking(), refresh()
- **Operations**: load(), now()
- **Multiplicity**: 1, 2

The diagram illustrates the relationships and interactions between these classes, which are essential for understanding the system's structure.
Class Diagrams

• Information for class contains
  • Name
  • State
  • Behavior
Class Diagram

• Class name is required
• Other information optional
  • State, behavior
  • Types, visibility…

(a) Clock

(b) Clock
  secs:int
  mins:int
  hours:int
  setTime()
  adjustTime()
  reset()

(c) Clock
  secs:int
  mins:int
  hours:int
  setTime()
  adjustTime():void
  reset():void
UML Class Diagrams ↔ Java Code

- Different representation of same information
  - Name, state, behavior of class
  - Relationships between classes
- Should be able to derive one from the other
- Motivation
  - **UML ⇒ Java**
    - Implement code based on design written in UML
  - **Java ⇒ UML**
    - Create UML to document design of existing code
Java → UML : Clock Example

- Java
  ```java
class Clock {
    // name
    // state
    int seconds;
    int minutes;
    int hours;
    // behavior
    void start();
    void adjustTime();
    void reset();
  }
  ```

Java Code  Class Diagram
UML Class Diagram Notation

- Type  \(\Rightarrow\) type name preceded by colon : 
- Visibility  \(\Rightarrow\) prefix symbol
  - + public
  - - private
  - # protected
  - ~ package
- Static  \(\Rightarrow\) underline
- Types of relationships
  - Generalization
    - Inheritance
    - Implementation
  - Association
    - Dependency
Java → UML: Clock Example

- Java
  
  ```java
  class Clock {
      // name
      // state
      private int seconds;
      private int minutes;
      private int hours;
      // behavior
      public void setTime();
      public void adjustTime(int value);
      public void reset();
  }
  ```

- Class Diagram
  
  **Clock**
  - seconds : int
  - minutes : int
  - hours : int
  + setTime( ) : void
  + adjustTime( ) : void
  + reset( ) : void
Generalization

• Denotes inheritance between classes
  • Can view as “is a” relationship

• Example
  • Lecturer is a person (Lecturer extends Person class)

• Types of generalization
  • Subclass extends superclass
    • Solid line ending in (open) triangle
  • Class implements interface
    • Dotted line ending in (open) triangle
Generalization Example

- Inheritance

Laptop, Desktop, PDA inherit state & behavior from Computer
Generalization Example

- Abstract Classes are represented by italicizing the name

  Abstract class Shape

- Interfaces are prefaced with <<interface>>

  Laptop implements DVDplayer interface
Association

- Denotes interaction between two classes
- Example
  - Lecturer teaches course
    - Indicates relationship between Lecturer & Course
Association w/ Navigation

- Navigation information
  - Relationship between classes may be directional
    - Only class A can send messages to class B
    - Arrowhead indicates direction of relationship

- Example

```java
class Course {
    Lecturer TheBoss;
}
```

```java
class Lecturer {
}
```
Association w/o Navigation

- Undirected edge
  - Relationship between classes may be bi-directional
  - Direction of relationship may be unknown
- Examples

```java
class Course {
    Lecturer TheBoss;
}

class Lecturer {
    Course [] class;
}

class Foo

class Bar
```
Permanent Association

- Permanent / structural association
  - Class A contains reference to class B in data field
  - Can view as “has a” relationship
  - Also referred to as composition

- Example

```
class A {
    B x;
}

class B {
    ...
}
```

A has a B
Temporary Association (Dependency)

- A transitory relationship between classes
  - Always directed (class A depends on B)
  - Indicates change in class B may affect class A
  - Can view as “uses a” relationship
  - Represented by dotted line with arrowhead

- Example

A \depends \text{on} B
Dependency

- Dependence may be caused by
  - Local variable
  - Parameter
  - Return value

- Example

```java
class A {
    B foo(B x) {
        B y = new( );
        ...
    }
}

class B {
    ...
    ...
    ...
}
```
Inner/Nested Classes

- Anchor (cross inside a circle) associated with enclosing class
UML Examples

• Read UML class diagram
  • Try to understand relationships
  • Practice converting to / from Java code

• Examples
  • Computer disk organization
  • Banking system
  • Home heating system
  • Printing system
UML Example – Computer System

- Try to read & understand UML diagram

- CPU is associated with Controllers
- DiskDrive is associated with SCSIController
- SCSIController is a (type of) Controller
• Bank associated with Accounts
• Checking, Savings, MoneyMarket are type of Accounts
UML Example – Home Heating System

- Thermostat associated with (has a) Room
- Thermostat associated with (has a) Heater
- ElectricHeater is a specialized Heater
- AubeTH101D is a specialized Thermostat
UML Example – Library System

- Try to read & understand UML diagram

- Books are associated with (has some) Pages
- Patron & Shelf depend on (temporarily use) Books
UML → Java : Computer System

- UML

- Java

  class Controller {
  }

  class SCSIController extends Controller {
  }
UML → Java : Computer System

- **UML**

- **Java**
  - Design code using all available information in UML...
Java

class CPU {
    Controller myCtlrs[ ];
}
class Controller {
    CPU myCPU;
}
class SCSIController extends Controller {
    DiskDrive myDrive[4];
}
Class DiskDrive {
    SCSIController mySCSI;
}
Java → UML : Printing System

- Java

  class Registry {
    PrintQueue findQueue();
  }
  
  class PrintQueue {
    List printJobs;
    Printer myPrinter;
    Registry myRegistry;
    void newJob();
    int length();
    Resources getResource();
  }
Java → UML: Printing System

- Java

```java
class Printer {
    Resources myResources;
    Job curJob;
    void print();
    boolean busy();
    boolean on();
}

class Job {
    Job(Registry r) {
        ...
    }
}
```
Java → UML: Printing System

- Java
  - All together

```
Registry
  findQueue(): PrintQueue

PrintQueue
  printJobs: List
  myPrinter: Printer
  myRegistry: Registry
  newJob(): void
  length(): int
  getResource(): Resources

Job

Printer
  myResources: resources
  curJob: Job
  print(): void
  busy(): boolean
  on(): boolean
```