CS1007: Object Oriented Design and Programming in Java

Lecture #5

Jan 31

Shlomo Hershkop
shlomo@cs.columbia.edu

1

Outline

- Feedback
- UML
- More UML and OOD

- Reading
  - Chapter 2

2

Feedback

- If you feel that the pace is too fast, or just don't get something, feel free to stop by office hours
- Will reformat slides online so its easier to read
- For exception questions
  - Please stop by office hours
- How to take notes
  - Will be using handouts for code/diagrams

3

Definition: UML

- It is a language
  - Syntax & Semantics
- When we model a concept there rules on how things can be put together and what it means when they are organized in a specific way

4

Applications

- Software design
- System requirements
- Documenting system process

5

View

- UML provides a view
  - Many views of system
  - Don’t stuff everything into one huge diagram
  - Specific diagram types can express specific concept
  - Sometimes multiple diagrams can apply
    - Pick best which you think can express the idea
- We will be covering a simple UML

6

Modeling

- What is modeling?
- Means to capture idea, relationship, decision, and requirements in a well defined notation.

7

Diagrams

- Visual representation of concepts and relationships
- Structural Diagrams
- Behavior Diagrams

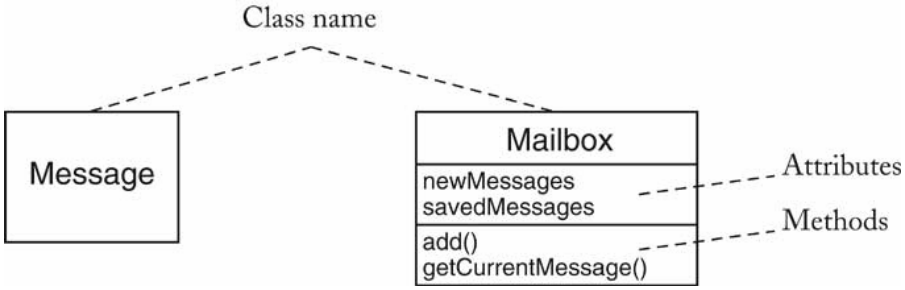
8

Class Diagrams

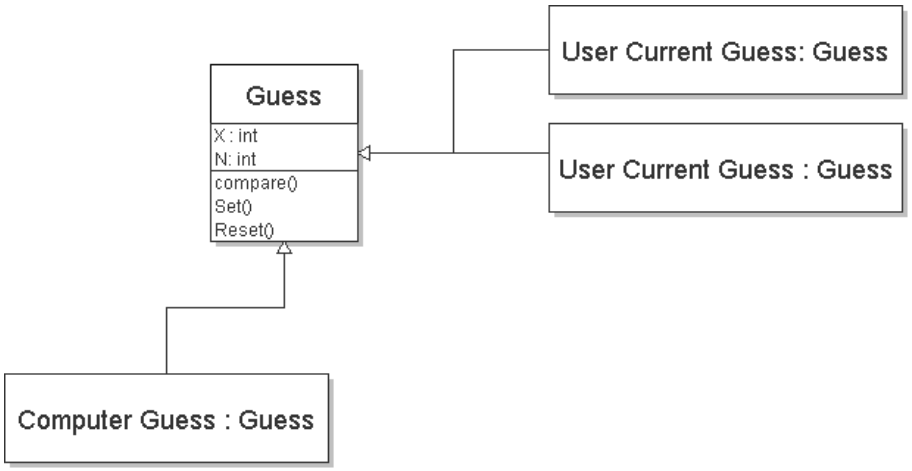
- Classes and interfaces to capture details about the entities that make up your system and static relationships.
- Rectangle with class name
- Optional compartments
  - Variables
  - Methods (sometimes shorthand)
- Include main Variables and methods

9

Class Diagram



10



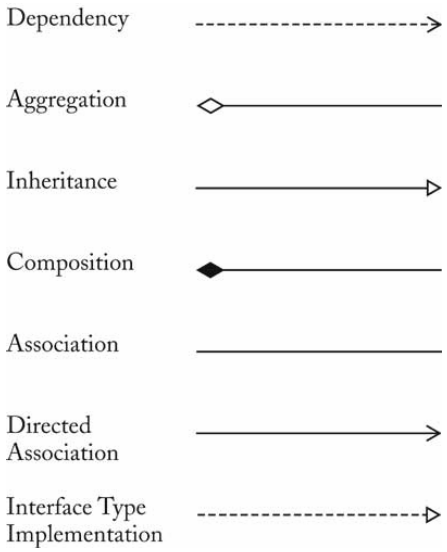
11

Relationships

- When describe classes, want to diagram how they are related
- Expressed in different types of connectors
- We covered 3 last time
  - Dependency
  - Aggregation
  - Inheritance
- Will redo, and cover others today

12

UML Relationships



13

Dependency

- Weakest relationship
- Something “uses a” another class
- Can assume short lived relationship between classes

14

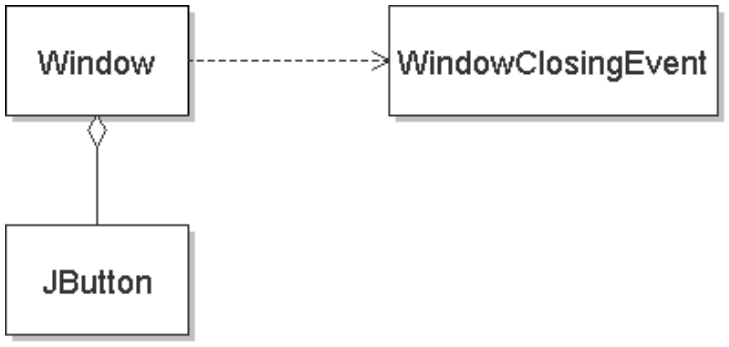
Association

- Stronger dependency
- Typically relationship over long periods of time
- Usually linked by not tied (can destroy one without destroying other).
- Example
- Window and curser
- (straight line)

15

Aggregation

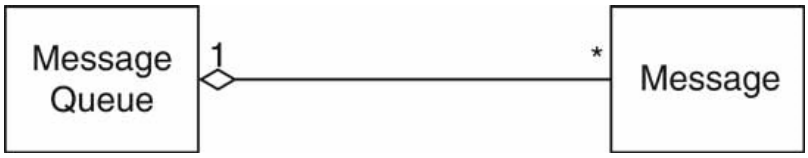
- Stronger than association
- Ownership between classes
- Window has a button



16

Multiplicities

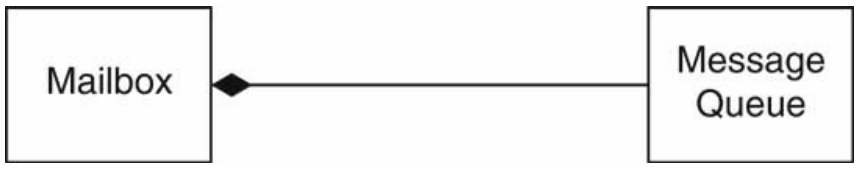
- Assume 1
- any number (0 or more): *
- one or more: 1..*
- zero or one: 0..1
- exactly one: 1



17

Composition

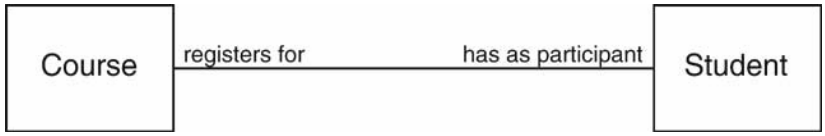
- Special form of aggregation
- Contained objects don't exist outside container
- Destroying main kills sub class
- ..is part of ….
- Example: message queues permanently contained in mail box



18

Association II

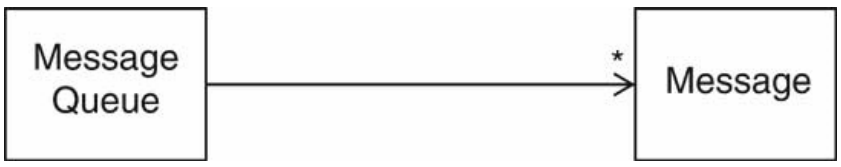
- Some designers don't like use aggregation
- More general association relationship
- Association can have roles



19

Association II

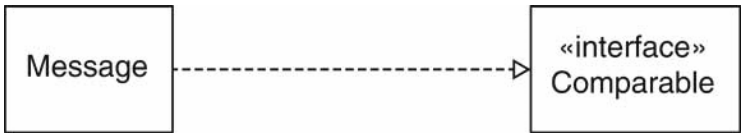
- Some associations are bidirectional
- Can navigate from either class to the other
- Example: Course has set of students, student has set of courses
- Some associations are directed
- Navigation is unidirectional
- Example: Message doesn't know about message queue containing it



20

Interface Types

- Interface type describes a set of methods
- No implementation, no state
- Class implements interface if it implements its methods
- In UML, use stereotype «interface»



21

Reminder

- Use UML to inform, not to impress
- Don't draw a single monster diagram
- Each diagram must have a specific purpose
- Omit inessential details

22

Use cases

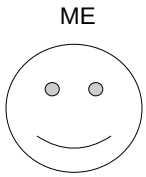
- Consists of interactions between the system and “actors” and their relationships
- Describes what the system does (not how)
- High level sketch of something of value to any actor

23

NOTE!

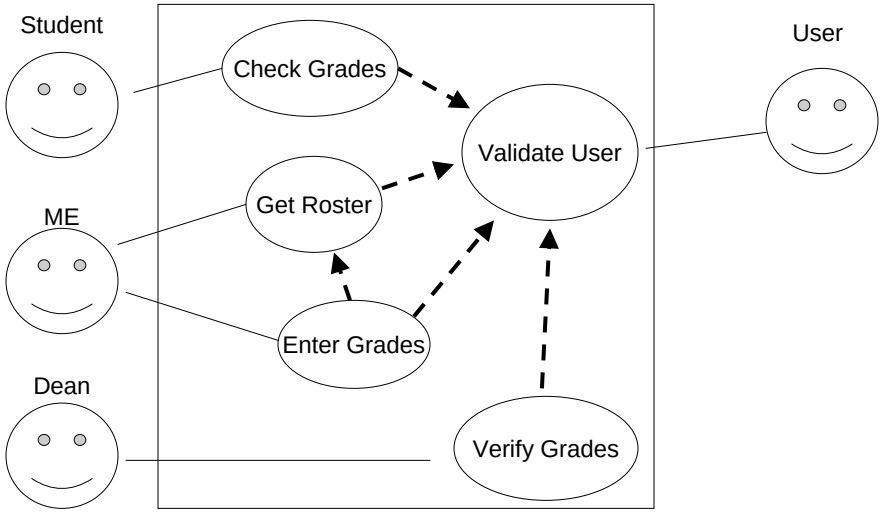
- What follows is a highly complex diagram which took your instructor hours to put together…..

24



25

Use case



26

Reach an Extension case

1. User dials main number of system
2. System speaks prompt

```
Enter mailbox number followed by #
```

3. User types extension number
4. System speaks

```
You have reached mailbox xxxx.
Please leave a message now
```

27

Leave a Message

1. Caller carries out Reach an Extension
2. Caller speaks message
3. Caller hangs up
4. System places message in mailbox

28

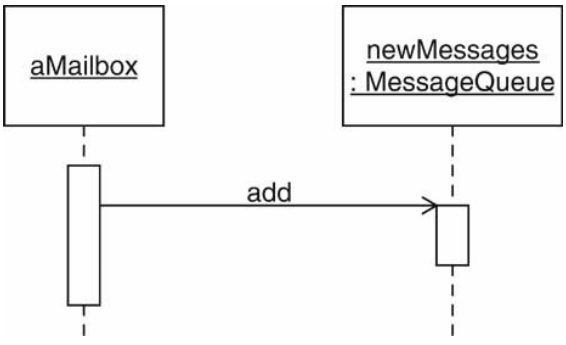
Sequence diagram

- Interaction between object over time
- Model several objects in a use case
  - Life of objects
  - Control points (method execution)

29

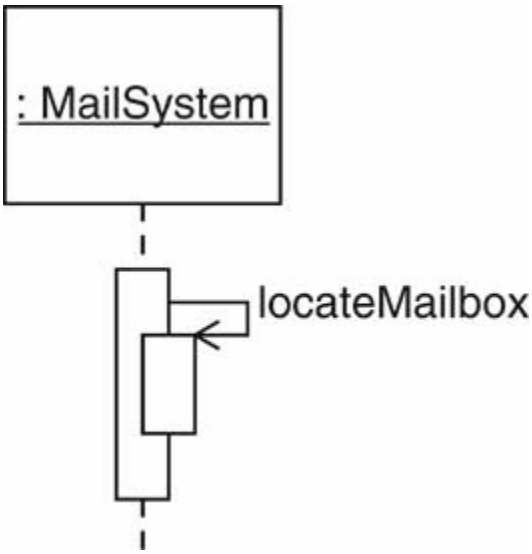
Sequence Diagrams II

- Object diagram: class name underlined



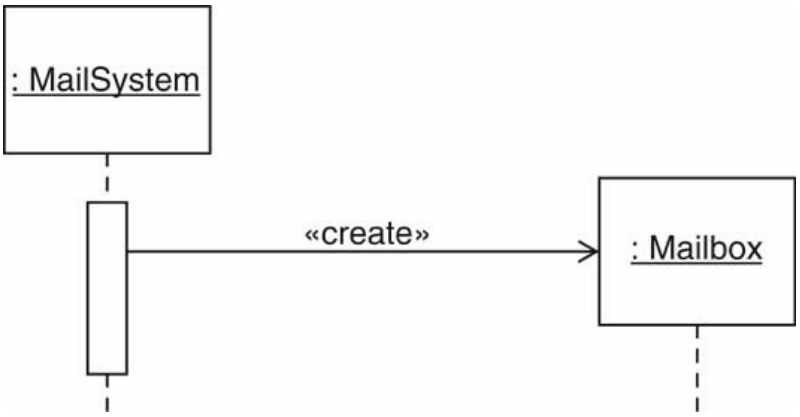
30

Method operating on object itself



31

Object Creation



32

State diagrams

- Describe system behavior
- All possible states of an object as events occur
- Not always applicable

33

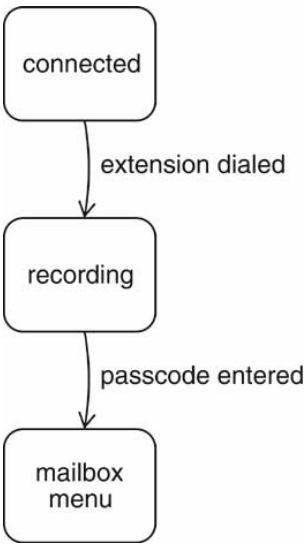
FSA background

- Finite State Automaton
- Model of behavior composed of
  - States
  - Transitions
  - Rules for transitions
  - Actions
- Inititial state, current, and end

- Studied in automata theory
  - Abstract machines and what they can solve
  - Used in AI
  - Related to formal language theory

34

State Diagram



35

Advantage

- Simple
- Easy to implement
- Given inputs, current state can predict state transition

36

Design Docs

- We will be using Javadoc comments
- Leave methods blank

```
/**
   Adds a message to the end of the new messages.
   @param aMessage a message
*/
public void addMessage(Message aMessage)
{
}
```

- run Javadoc to generate html pages
- Makes a good starting point for code later
- Good for sketching out ideas

37

Summary: UML Building Blocks

- model elements
  - (classes, interfaces, components, use cases, etc.)
- Relationships
  - (associations, generalization, dependencies, etc.)
- diagrams
  - (class diagrams, use case diagrams, interaction diagrams, etc.)

- Simple building blocks are used to create large, complex structures
  - elements, bonds and molecules in chemistry
  - components, connectors and circuit boards in hardware

38

More examples

- Will continue with components of voice mail system

39

- We want to sketch out some actions in the voicemail system
- Get a message
- Change a greeting
- What would both require? What would be related to that?

40

Retrieve Messages

1. Mailbox owner carries out **Log in**
2. Mailbox owner selects "retrieve messages" menu option
3. System plays message menu:

Press 1 to listen to the current message
Press 2 to delete the current message
Press 3 to save the current message
Press 4 to return to the mailbox menu

4. Mailbox owner selects "listen to current message"
5. System plays current new message, or, if no more new messages, current old message.
6. Note: Message is played, not removed from queue
7. System plays message menu
8. User selects "delete current message". Message is removed.
9. Continue with step 3.

41

Change Greeting

1. Mailbox owner carries out Log in
2. Mailbox owner selects "change greeting" menu option
3. Mailbox owner speaks new greeting
4. Mailbox owner presses #
5. System sets new greeting

42

Log In

1. Mailbox owner carries out `Reach an Extension`
2. Mailbox owner types password and # (Default password = mailbox number. To change, see `Change the Passcode`)
3. System plays mailbox menu:

Enter 1 to retrieve your messages.
Enter 2 to change your passcode.
Enter 3 to change your greeting.

43

Change Passcode

- Mailbox owner carries out Log in
- Mailbox owner selects "change passcode" menu option
- Mailbox owner dials new passcode
- Mailbox owner presses #
- System sets new passcode

44

Practice

- What would the use case to make a move in the mastermind game look like

45

CRC Cards

- Design method to discover classes
- By sketching out a class and roles and goals will start to discover other actors

46

CRC Cards: Mailbox

Mailbox	
keep new and saved messages	MessageQueue

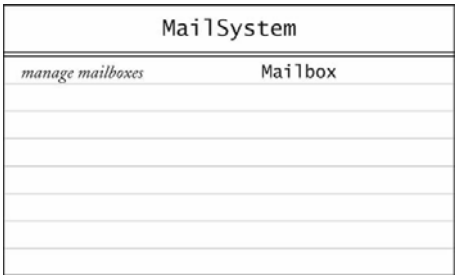
47

CRC Cards: MessageQueue

MessageQueue
add and remove messages in FIFO order

48

CRC Cards: MailSystem

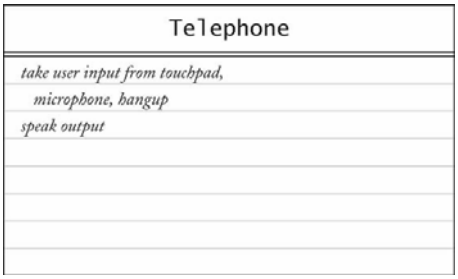


49

Telephone

- Who interacts with user?
- Telephone takes button presses, voice input
- Telephone speaks output to user

50



51

Connection

- With whom does Telephone communicate
- With MailSystem?
- What if there are multiple telephones?
- Each connection can be in different state
  - (dialing, recording, retrieving messages,...)
- Should mail system keep track of all connection states?
- Better to give this responsibility to a new class

52

Connection

Connection	
get input from telephone	Telephone
carry out user commands	MailSystem
keep track of state	

53

Analysis: Leave Message

1. User dials extension. Telephone sends number to Connection
   (Add collaborator Telephone to Connection)
2. Connection asks MailSystem to find matching Mailbox
3. Connection asks Mailbox for greeting
   (Add responsibility "manage greeting" to Mailbox,
   add collaborator Mailbox to Connection)
4. Connection asks Telephone to play greeting
5. User speaks greeting. Telephone asks Connection to record it.
   (Add responsibility "record voice input" to Connection)
6. User hangs up. Telephone notifies Connection.
7. Connection constructs Message
   (Add card for Message class,
   add collaborator Message to Connection)
8. Connection adds Message to Mailbox

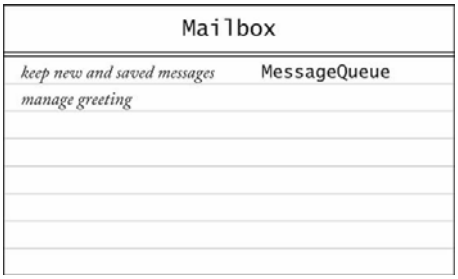
54

Telephone	
take user input from touchpad, microphone, hangup	Connection
speak output	

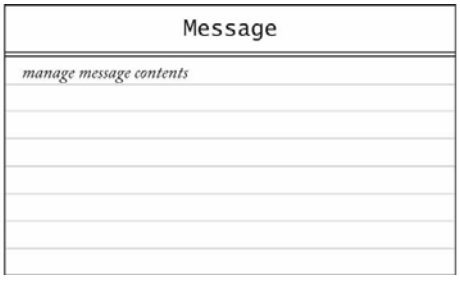
55

Connection	
get input from telephone	Telephone
carry out user commands	MailSystem
keep track of state	Mailbox
record voice input	Message

56


Mailbox
keep new and saved messages
MessageQueue
manage greeting

57


Message
manage message contents

58

Retrieve Messages

1. User types in passcode. `Telephone` notifies `Connection`
2. `Connection` asks `Mailbox` to check passcode.
   (Add responsibility "manage passcode" to Mailbox)
3. `Connection` sets current mailbox and asks `Telephone` to speak menu
4. User selects "retrieve messages". `Telephone` passes key to `Connection`
5. `Connection` asks `Telephone` to speak menu
6. User selects "listen to current message". `Telephone` passes key to `Connection`
7. `Connection` gets first message from current mailbox.
   (Add "retrieve messages" to responsibility of `Mailbox`).
   `Connection` asks `Telephone` to speak message
8. `Connection` asks `Telephone` to speak menu
9. User selects "save current message". `Telephone` passes key to `Connection`
10. `Connection` tells `Mailbox` to save message
    (Modify responsibility of Mailbox to "retrieve,save,delete messages")
11. `Connection` asks `Telephone` to speak menu

59

Result of Use Case Analysis

Mailbox	
keep new and saved messages	MessageQueue
manage greeting	
manage passcode	
retrieve, save, delete messages	

60

CRC Summary

- One card per class
- Responsibilities at high level
- Use scenario walkthroughs to fill in cards
- Usually, the first design isn't perfect.
  - (You just saw the author's third design of the mail system)

61

Next Time

- Wrap up UML
- OOP Theory background
- Considerations when choosing and designing classes

- Will cover theory and code next class
- Finish reading chapter 2 (2-2.5), start 3

62