CS1007: Object Oriented Design and Programming in Java

Lecture #6
Feb 2
Shlomo Hershkop
shlomo@cs.columbia.edu

Outline

• Feedback
• Wrap up last time stuff
• Actual Code
• More Background material
  – Some Theory
  – Encapsulation
  – Inheritance
  – Interface
• Class design

• Reading
  – Chapter 2.5-end, 3.1
Feedback

• Interface questions
  – We will cover code examples today
  – What if you choose a random class to implement mouseListener
• If something is confusing…see me after class or next office hours…each lecture builds on the last
• Voicemail VS other example
  – Limited time for class
  – Want to make sure everyone understands the book, this will come into play when we do code review

Feedback

• Explanation of what we did last time (and today) for UML + voice system
• Please TELL me if you need more time to copy down something….you might not be the only one
• Software engineering in general is abstract, so we cover it on theory, doc, code levels
Feedback

• UML design requirements on the HWs?
  – Will be told which docs you need to generate
• How many diagrams /use cases necessary?
  – Usually we will do specific ones on hw
• Java inheritance, Javadoc
  – Will cover today
• Interface/Extend class

Announcement

• For the homework, you should be adding methods and/or constructors as you see fit….the assignment is just bare minimum
• Q: Don’t know where to begin…..
• Q: Need programming help…. 
• A: OH = Office Hours
From last time

- We covered basic UML building blocks
- Started to sketch out UML/CRC of a voice mail system
  - Result in a few basic pieces and relationship
  - Will now look at charts
  - If you don’t follow a step, please stop me and ask

UML Class Diagram for Mail System

- CRC collaborators yield dependencies
- Mailbox depends on MessageQueue
- Message doesn't depend on Mailbox
- Connection depends on Telephone, MailSystem, Message, Mailbox
- Telephone depends on Connection
Dependency Relationships

- A mail system has mailboxes
- A mailbox has two message queues
- A message queue has some number of messages
- A connection has a current mailbox.
- A connection has references to a mail system and a telephone

Aggregation Relationships
UML Class Diagram for Voice Mail System

Sequence Diagram for Use Case: Leave a message
Interpreting this Sequence Diagram

- Each key press results in separate call to dial, but only one is shown
- Connection wants to get greeting to play
- Each mailbox knows its greeting
- Connection must find mailbox object:
  - Call findMailbox on MailSystem object
- Parameters are not displayed (e.g. mailbox number)
- Return values are not displayed (e.g. found mailbox)
- Note that connection holds on to that mailbox over multiple calls
More javadoc

- For javadoc to work the javadoc style comments
  1. Immediately precede
     - Public class
     - Public method
     - Public item
  2. /**     */
Tags

• @param ParamName description
• @return description of return value
• @throws ExceptionType explanation

• @author yourname
• @version version info
• @see Package.Class

Running javadoc

• javadoc –d doc_dir package_name

• Near the classes

• Can run on single class
  – Javadoc something.java

• Javadoc -?
• Will look at all the components of the code in chapter 2.

```java
01: /**
02:   A message left by the caller.
03: */
04: public class Message
05: {
06:   /**
07:     Construct a Message object.
08:     @param messageText the message text
09: */
10:   public Message(String messageText)
11:   {
12:     text = messageText;
13:   }
14: }
15: /**
16:   Get the message text.
17:   @return message text
18: */
19: public String getText()
20: {
21:   return text;
22: }
23: }
```
For MessageQueue

```java
/**
* Get the total number of messages in the queue.
* @return the total number of messages in the queue
*/
public int size()
{
    return queue.size();
}

/**
* Get message at head.
* @return message that is at the head of the queue, or null
* if the queue is empty
*/
public Message peek()
{
    if (queue.size() == 0) return null;
    else return queue.get(0);
}

private ArrayList<Message> queue;
```

Tester

```java
import java.util.Scanner;

/**
* This program tests the mail system. A single phone
* communicates with the program through
* System.in/System.out.
*/
public class MailSystemTester
{
    public static void main(String[] args)
    {
        MailSystem system = new MailSystem(MAILBOX_COUNT);
        Scanner console = new Scanner(System.in);
        Telephone p = new Telephone(console);
        Connection c = new Connection(system, p);
        p.run(c);
    }
    private static final int MAILBOX_COUNT = 20;
```
Violet

• Simple UML system
• Will demo
• Be sure to download and play with it before hw2

Background

• Interface
• Theory
• Class hierarchies
Interface in Java

- Just like a class is a type so it an interface
- Used to define a behavior
- Can not create an instance
- A class can choose to implement an interface thus in a sense instantiating the interface
- Idea of something…..

Example

- Music player control
- What would we expect to see supported?
- What does it mean in context?
Interface definition

• Generally set public access
• Contains method signatures
  - public boolean isActive();
  - public void startCount(int count);
• Can also contain defined constants
  - public static final int START = 0;

```java
public interface musicControl {

    /**
     * Will rewind the player n moves
     * @param n number of moves to move back
     */
    public void setRewind(int n);

    public void fastForward();
    public void play();
    public void pause();
    public void jumpShuffle();
    public void jumpForward();
    public void jumpBackward();
}
```
public interface musicControl {

    public static int SHUFFLE = 0;
    public static int FORWARD = 1;
    public static int BACKWARD = 2;

    /**
     * Will rewind the player n moves
     * @param n number of moves to move back
     */
    public void setRewind(int n);

    public void fastForward();

    public void play();

    public void pause();

    public void jump(int n);
}

How to use the interface

• Class says “implements ..”
  - public class cheapIPOD implements musicPlayer {
• If more than one, use commas to separate
• Must define all interface methods
• Interface is a type, so that if the class implements it, I can pass it to a method
  which expects the interface type

    public void calculate(musicPlayer x)
public class cheapIpod implements musicPlayer{
    ...
    public void jump(int j) {

        if( j == musicPlayer.SHUFFLE) { ...}
        else if(j == musicPlayer.FORWARD){..}
        else if(j == musicPlayer.BACKWARD){..}
        else { throw new illegalCommandException(...) } 
    }
}

Chaining interfaces

• Can derive one interface from another
public interface basicMail {...}
public interface blackberryMail extends basicMail {..}
WARNING

- Java couldn’t care less about what the interface methods are supposed to do

- Who’s job is it?

- How can it be accomplished?

Designing Objects

- Secret of OOD is to balance Abstraction and Encapsulation
Next Time

• Do homework assignment
• Read chapter 3-3.3