Highlights of Previous Lecture

- Introduction to final project
- Introduction of the first project interface types
Homework Problems

- not declaring your classes public - be sure your classes that communicate with mine are `public class Whatever`
- not having the correct, required constructor arguments
- having multiple, conflicting copies of interface files - on classes I provide, do not have local, personal copies, just use the jar archive
- getting the `-classpath` wrong for compiling or executing - use `java -classpath reservations.jar:...` and `javac -classpath reservations.jar:...`
Homework Solution

MinimalFlightImpl.java

```java
import reservations.Flight;
import reservations.Itinerary;
import java.util.Date;
import java.util.Iterator;

public class MinimalFlightImpl implements Flight {

    Itinerary itinerary = null;

    public MinimalFlightImpl(Itinerary i) {
        itinerary = i;
        itinerary.register(this);
    }

    public Itinerary getItinerary() {
        return itinerary;
    }

    public Iterator getReservationIterator() {
        return null;
    }

    public Date getLeavingTime() {
        if (itinerary != null)
            return itinerary.getDepartureTime();
        else
            return null;
    }
}
```
public String toString() {
    return "MinimalFlight itinerary:" + itinerary.toString();
}
Final Project - Description

The class final project will be to develop an airline reservation system for a small airline that has some number of Flights corresponding to a number of Itineraries. Reservation Requests to book (create) Reservations on flights between various destinations will be made by Passengers.
Final Project - Flight and Itinerary objects

... a small airline that has a number of Flights corresponding to a number of Itineraries. ... 

- You will be provided with information on starting and destination airports and departure times in the form of Itinerary objects documented here.
- You need to associate these Itinerary objects with your implementations of the Flight interface as documented here.
Final Project - ReservationRequests and Reservation objects

... ReservationRequests to book (create) Reservations on flights between various destinations will be made by Passengers.

- You will be provided with a sequence of `ReservationRequest` objects as documented [here](#).
- You will be expected to match up ReservationRequest Itineraries with the Itineraries of your Flights and create objects implementing the `Reservation` interface documented [here](#).
- You will need to associate the resulting collection of Reservations with your Flight objects
Final project - Goals

1. Set up collections of Flights
2. Maintain information about reservation availability on flights
3. Set up collections of Reservations on Flights
4. Respond to reservation requests
5. Support the Flight and Reservation interfaces so that my code can invoke methods defined by them.
6. **Create solutions that demonstrate Object Oriented programming skills**

For convenience, [here](#) is documentation about all these interfaces and classes.
Final Project - Delivery

- The final project is due by 5:00pm, Monday, December 10. **no extensions.** Allow time for email delivery problems. You can submit multiple copies; I will only evaluate the last one received.
- You must submit your final project as a jar archive. Please be sure the archive is structured as I’ve required before:
  - the jar archive file name must identify you
  - Be sure the jar archive is received by cs332@cs.gmu.edu by the deadline.
  - CC: yourself, detach the jar archive attachment, and be sure the jar contains what you think it should: `jar tf your_jar.jar`
- Place all your **source** files in a directory whose name will identify you, either your email address or your GMU ID. If your project defines any Java packages, please place those package directories **below** the directory that identifies you. If you are not defining java packages, all source files should reside in your top level directory.
- Include a README file in your jar archive explaining how to run your program and any details necessary to compile your source files. You may also include a brief statement about your design or any other aspects of your project you want me to pay particular attention to.
- Do **not** submit your *.class files. Do **not** submit copies of my jar archives. Just your source code and any README file.
Final Project - Overview

- Your code will contain the `main` method to control reservation system operations.
- While I provide some test code that you can use to test if our classes can communicate, you will probably have some setup and initialization that you will want to design. Control of how that gets going is up to you to start in your main method.
- You will be required to call some of my code to get inputs:
  - Itineraries
  - ReservationRequests
Final Project - Minimums

To get a B grade on the final project you would have to write code to at least accomplish the following minimum capabilities:

- Create `Flight` (implementing) objects for each of the `Itinerary` objects that are made available via the `ItineraryReader`
- Create `Reservation` objects in response to each of the `ReservationRequest` objects that will be made available to you by a `ReservationRequestGenerator`
- Add each of the reservations to a collection associated with the appropriate `Flight`.
- Provide non-null, simple, useful responses to each of the methods in the `Flight` and `Reservation` interfaces.
- The main goal is to support
  - the `getReservationIterator()` method of the `Flight` interface (which should result in a series of objects implementing the Reservation interface);
  - and the `getFlight()` method of the `Reservation` interface.
- For a minimum implementation your `Seat`, `SeatCategory`, `SeatPrice` implementations can always be the same. For example all your flights have "open, economy class seating" and only one price.
- Do all of the above demonstrating object oriented principles
Final Project - Options

To get better than a B, you need to support at least **two or more** of the following options (in an object oriented way):

1. Implement different collections of seats for a flight corresponding to different SeatCategories.
2. Implement different sized planes with different strategies for assigning reservations, but keeping as much common code as possible defined only once.
3. Provide a more sophisticated implementation of different SeatCategories that will attempt to maximize profit for a flight: e.g., be sure there are always FirstClass and BusinessClass seats available and/or as flight departure time approaches, lower the prices on seats to "encourage" passengers to book flights on your flights.
4. Assign seats to passenger’s reservation requests and create an implementation of the Seat interface that will return suitable information about the assigned seat.
5. You could choose to adjust the leaving time of flights to be other than the departure time provided with Itinerary objects. You may not make those departure times earlier but you can make the leaving time up to three hours later so as to cram more reservations on to the flights and thus increase profits.
6. You could choose to book passengers on a set of connecting flights rather than on a single flight. Note that you will have to do some tricky work to get the resulting Reservation to reflect a multi-part Itinerary without modifying the Reservation interface. **The Itinerary start and destination airports you have seen so far do not support this. I will provide some more realistic Itinerary data that could be used to support the idea of connecting flights.**
7. Implement some kind of frequent flyer bonus system to reward repeat passengers.
8. Some other capability that you can convince me will take sufficient effort (on your part, and doesn’t require me to provide extra data/software) and demonstrates object oriented principles.
Final Project - Grading Guidelines

Your basic grade depends on how much work your project gives evidence of and whether it meets the requirements. Here is what you can expect:

- Expect a D or less if your project shows very little effort. That would include evidence like source files that have obvious problems that the compiler would find immediately and that I’ll notice right away: misspelled Java keywords and language objects, for example.

- Expect a C if your project demonstrates some effort on your part but does not meet the minimal requirements: your classes must implement the Java interfaces that I required and your classes must handle the basic inputs you are provided.

You can expect a C even if your project has some small compilation or execution problems if you show sufficient evidence that you understand and can apply the concepts taught in the course.

- Expect a B if you handle the minimum requirements as outlined above and you do so showing some evidence of having learned some object oriented concepts.

- You are in A territory if you demonstrate an effort to go beyond the minimum, implementing optional capabilities in an object oriented way.
Final Project - Pluses

Here are a number of factors that will influence whether your final project grade stays at those starting points or goes up or down. First, here are some factors that will positively impact your grade; factors that I’ve emphasized or demonstrated throughout the semester:

- Evidence of a good design: using object oriented concepts **appropriately**:
  - Delegation of responsibilities to other classes
  - Encapsulation of information
  - Appropriate use of exceptions
  - Using additional interfaces and abstract data types correctly
  - Appropriate use of inheritance
- Small objects focused on a single aspect of a reservation system.
- Small methods focused on one aspect of an object’s responsibilities.
- Using appropriate, meaningful names for objects and methods,
- Well structured, consistently formatted, understandable code
- Use of design patterns appropriately
- Use of javadoc style documentation: documentation written as if the classes were intended to be re-used (and that doesn’t just say what the code says.)
- Using Java coding standards
Final Project - Minuses

Here are some factors that will negatively impact your project’s grade, regardless of how well your project’s software mimics the behavior of a reservation system:

- Code written in a procedural style
- Long blocks of code
- Code that is duplicated unnecessarily
- Unnecessary use of static fields: global data or information that should be encapsulated in objects that is not.
- Too much coupling between classes: objects or classes that depend on each other too much: A depends on B and B depends on A. (A little of this may be unavoidable.) Delegation does not imply inter-dependence, however.
- Classes whose implementations depend on the particular implementation of other classes. Insufficient encapsulation of implementation details.
- Objects that modify the data of other objects directly. Again, insufficient encapsulation.
- Dependencies on magic numbers: hard-coded limits.
- Code that is hard to understand.
- Obvious code copying between students. These are to be individual efforts
- Not meeting the final project delivery requirements: jar/zip archives correctly structured so that I can evaluate them easily.
- And last but not least, not getting the project in by the deadline.
Designing Objects

- Visualize objects as bundles of services, not bundles of data.
- Separate interface from implementation.
- Make your objects small and self-contained.
- If you find your classes becoming large and with too much responsibility, split it up into several classes.
- If you find code being duplicated in related objects
  - Create a base class and push the duplicated code into it, or
  - Move the code to an object that can be delegated to.
Designing Object Semantics

- Keep objects focused.
- Minimize the number of methods in object interfaces: make interfaces specific to a particular role
- Make sure subtypes fulfill the semantic contracts of their supertypes.

Things to Strive For

- Lots of small, private (or maybe protected), helper methods
- Relatively few public methods
- Readable code
Designing Object Fields and Methods

- Make the types of your method parameters and return values only as specific as necessary (and no more so)
- Where appropriate, use interface types
- Avoid data overloading - use type safe objects.

Corollaries

- Create objects types liberally
- Prefer constants over hard-coded literals
- Access to fields should be by way of accessor methods
Homework

Reading

- Chapter 12 - Wrapping Up Our Modeling Efforts
- Chapter 13 - A Deeper Look at Java
- Chapter 14 - Transforming Your Model into Java Code - not required for the final exam but you might find useful to read for ideas that you could apply to the final project.

Homework

The goals of this homework are

- To get some additional practice implementing an interface
- To think about the final project issues
- To get some more practice working with the jar archives I will provide in your development environment

Write a class that implements the Reservations interface

- You don’t have to be concerned with the details of the getFlight method as yet (although it would be helpful if you would at least assign some (the same) Flight object with each Reservation.
- Similarly, I am not (for this homework) concerned that you provide realistic data in response to the getPassenger, getSeat, getSeatCategory, or getSeatPrice methods.
- I will provide you with a jar file that contains code and data to generate sample ReservationRequest objects and documentation on how to use it.
- You will need to create a class or classes that contains your main method, asks for each ReservationRequest object in turn, and generates a Reservation object in response.
- Please make sure that, as you get each ReservationRequest object that you call its toString() method and print the result, followed by a call to a toString()
method on your Reservation implementation object, printing that result. **Hint:**
do not have your Reservation implementation call your Flight
implementations toString() method, which could then recursively iterate
through all the Reservations ...

This will be due by 5:00 PM, Wednesday, 28 November.