COMP/EECE 4081

# Midterm Exam

Fall 2012

Name: Answer Key

**Rules:**

- No potty breaks.
- Turn off cell phones.
- Closed book, closed note, closed neighbor.

**Reminders:**

- Verify that you have 14 pages.
- Don't forget to write your name.
- Read each question carefully.
- Don't forget to answer every question.

**Additional Items:**

- For questions that involve writing code, you may omit import statements unless specifically asked for them.

1. [2pts] Which one of the these is a <u>bad</u> length for an iteration?
   [✓ a.] 1 week
   b. 2 weeks
   c. 4 weeks
   d. 6 weeks
   e. All of the above

2. [2pts] What <u>flawed</u> assumption is the waterfall process model based upon?

The waterfall process modul Falsely assumes:
* that specifications are predictable and stable and
* that they can be corredly defined at the start, with low change rates.

3. [2pts] Which one of these <u>is</u> appropriate in an agile and iterative development process?
   a. Gather a complete set of requirements before designing/building anything.
   b. Implement the backend of the system first—that is, before implementing the frontend functionality with which users interact.
   c. Generate and maintain complete, detailed design documents, which comprehensively model all aspects of the design.
   [✓ d.] Implement the system incrementally, building it up bit by bit.
   e. Test the code at the end, after the system has been completely implemented.

**UC Rent DVDs**

1. User presses the checkout button.
2. System creates an SQL query and executes it on the MySQL database to retrieve the contents of the User's shopping cart.
3. System displays an itemized list of the User's selections and the total cost.
4. User provides payment information.
5. System validates the payment info.
6. System records the sale.
7. System displays a receipt for the sale.

**Figure 1. Example use case.**

4. [3pts] Does the use case depicted in Figure 1 follow the "essential style"? Justify your answer, citing specific examples from the figure.

No. The essential style should be free of UI details; however the UC refers to a "checkout button" in step 1.

5. [3pts] Does the use case depicted in Figure 1 follow the "black-box style"? Justify your answer, citing specific examples from the figure.

No. The black-box style should be free of details regarding the System's implementation; however, step 2 refers to SQL and MySQL, which are implementation concerns.

```
public class IncreaseDiscountServlet extends HttpServlet {

        protected void doPost(HttpServletRequest request,
                              HttpServletResponse response)
            throws ServletException, IOException {
A:            UserProfile user = getUserProfile(request.getParameter("user"));
B:            int discount = user.getDiscount();
C:            ++discount;
D:            user.setDiscount(discount);
              ...
        }

        ...
}
```

**Figure 2. Servlet with concurrency bug. The purpose of this servlet is to increase by 1 the level of discount that a customer receives. Note the statements labeled A–D.**

6. [4pts] The servlet code in Figure 2 contains a concurrency-related defect that can cause discount additions to be lost. Fill in the blanks below to describe a step-by-step scenario, involving two threads T1 and T2, that illustrates how a discount increase can be lost. (The first step has been filled in for you.)

   - First, thread __T1__ executes line _A__
   - Next, thread T1 executes line B
   - Next, thread T1 executes line C
   - Next, thread T2 executes line A
   - Next, thread T2 executes line B
   - Next, thread T2 executes line C
   - Next, thread T2 executes line D
   - Finally, thread T1 executes line D

7. [2pts] True or false? The above concurrency error could be corrected by making the UserProfile follow the monitor pattern (e.g., by making all its methods `synchronized`).

   a. True

   [✓ b.] False

8. [2pts] Imagine that you just joined a development team that uses svn for version control and collaboration. To start contributing to the project, what svn operation would you most likely invoke first?

   [✓ a.] checkout

   b. commit

   c. export

   d. revert

   e. update

9. [2pts] Now, imagine that you have a working copy, but other team members have pushed changes into the repository since you created the working copy. What svn operation would you use to pull those changes into your working copy?

   a. checkout

   b. commit

   c. export

   d. revert

   [✓ e.] update

10. [2pts] Imagine that you try to commit your changes into the repository, but you get an out-of-date error. What would your next svn operation most likely be?

    a. checkout

    b. commit

    c. export

    d. revert

    [✓ e.] update

**Artist**

| ArtistID | ArtistName |
|---|---|
| 21233 | Pixies |
| 55621 | Cure |
| 10311 | Smiths |

**ArtistAlbum**

| ArtistID | AlbumID |
|---|---|
| 21233 | 75531 |
| 10311 | 00598 |
| 21233 | 31300 |

**Album**

| AlbumID | AlbumTitle | AlbumYear |
|---|---|---|
| 75531 | Surfer Rosa | 1988 |
| 00598 | Meat Is Murder | 1984 |
| 31300 | Come On Pilgrim | 1987 |

**Figure 3. Database tables for a music catalog. All columns in ArtistAlbum are foreign keys.**

11. [6pts] Consider the database tables in Figure 3. Fill the table below to match the result returned by the following query. Cross out any cells in the table that you do not need. Don't forget to label the columns.

```
SELECT ArtistName, AlbumTitle, AlbumYear FROM
(Artist INNER JOIN ArtistAlbum
 ON Artist.ArtistID = ArtistAlbum.ArtistID)
INNER JOIN Album ON ArtistAlbum.AlbumID = Album.AlbumID
ORDER BY ArtistName, AlbumYear, AlbumTitle;
```

| ArtistName | AlbumTitle | AlbumYear | |
|---|---|---|---|
| Pixies | Come On Pilgrim | 1987 | |
| Pixies | Surfer Rosa | 1988 | |
| Smiths | Meat Is Murder | 1984 | |
| | | | |
| | | | |
| | | | |

12. [2pts] A user is trying to run the `IncreaseDiscountServlet` using this URL:

http://localhost:8080/MidtermExamples/IncreaseDiscount.do

But tomcat keeps giving her a 404 error message.

Correct the following DD to fix the problem by crossing out the offending parts and inserting corrections as necessary.

```
<?xml version="1.0" encoding="UTF-8"?>

<web-app …>

 <display-name>MidtermExamples</display-name>

 <servlet>

   <display-name>IncreaseDiscountServlet</display-name>

   <servlet-name>IncreaseDiscountServlet</servlet-name>

   <servlet-class>com.dvd.IncreaseDiscountServlet</servlet-class>

 </servlet>

 <servlet-mapping>

   <servlet-name>IncreaseDiscountServlet</servlet-name>

                                      .do
   <url-pattern>/IncreaseDiscount~~Servlet~~</url-pattern>

 </servlet-mapping>

</web-app>
```

13. [3pts] Fill in each blank with either *GET* or *POST*.

- GET requests must be idempotent.
- GET requests have no data payload.
- Clicking a hyperlink in a webpage typically produces a GET request.

14. [2pts] Which one of these mechanisms is most often used to handle HTTP sessions.

    a. Bookmarks
    b. Browser history
    [✓ c.] Cookies
    d. HTTP WRITE method
    e. ServletConfig object

15. [3pts] For each of the following types of artifacts, tell which role in an MVC architecture the artifact would typically play. Give the full name of the role. (Giving only the letter M, V, or C will earn you only partial credit.)

- Plain old Java class: Model
- Servlet class: Controller
- JSP: View

For questions 16 and 17, imagine you are developing a web app for an on-line DVD rental store. Each of the questions asks you to implement a feature. In your solutions, use the classes defined in the Appendix (especially the domain ones; assume they are already implemented).

16. [15pts] For this question, your task is to implement a feature that displays the contents of a user's shopping cart. Assume that as the user browses other DVD-store pages, she adds videos to her shopping cart. You must implement either one servlet or one JSP (your choice) that takes an HTTP GET request and displays the contents of the user's shopping cart as shown in Figure 4. Assume that each session object has an attribute "cart" that refers to a ShoppingCart object. There is additional space on the next page for your answer.

JSP Solution:

```
<!DOCTYPE html>
<html>
<body>
<h1> Shopping Cart</h1>
<ul>
<% ShoppingCart cart =
        (ShoppingCart)session.getAttribute("cart");
    int total =0;
    For (int i=0; i< cart.size(); ++i) {
        Movie m = cart.get(i);
        total += m.getPrice();
%>
<li><%= m.getTitle() %>(<%= m.getYear() %>) -
    $<%= m.getPrice() %>
<% } %>
Total: $<%= total %>
```

```
</body>
</html>
```

17. [15pts] For this question, your task is to implement one servlet and one JSP that work together (as in MVC) to compute and display recommendations for movies that the user might like. Your servlet must take an HTTP POST request, which includes a "targetMovieID" parameter, and must display a list of recommendations for movies similar to the target movie (using the MovieRecommender class). The recommendations must be formatted as in Figure 5. Your code need not display the form that produced the POST request (assume some other servlet or JSP does that). There is additional space on the next page for your answer.

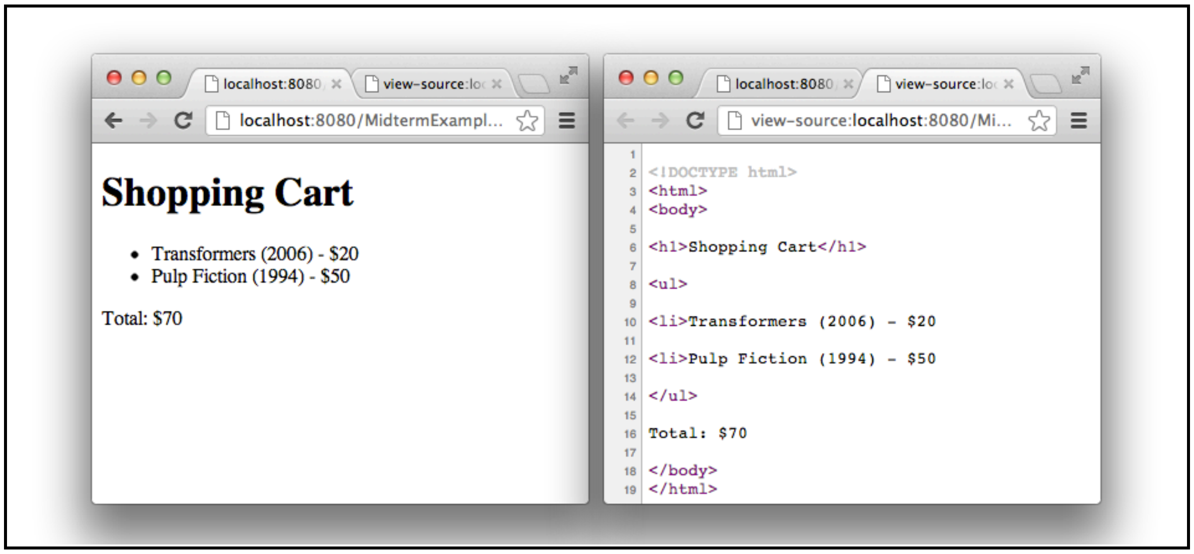
```
public class RecommendMoviesServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request,
                          HttpServletResponse resp) {
        String target = request.getParameter("targetMovieID");
        Vector<Movie> recs = MovieRecommender.recommendSimilar(target);
        request.setAttribute("recs", recs);
        RequestDispatcher view =
              request.getRequestDispatcher("recommend.jsp");
        view.forward(request, resp);
    }
}
```

recommend.jsp

```
<!DOCTYPE html>
<html>
<body>
<h1> Recommendations </h1>
```

```
<%  Vector<Movie> recs =
        (Vector<Movie>)requert.getAttribute("recs");
    For(int i=0;  i < recs.size(); ++i) {
        Movie m = recs.get(i);
%>
Movie: <%= m.getTitle() %> <br>
Director: <%= m.getDirector() %> <br>
Length: <%= m.getLength() %> minuter <br><br>
<%  }  %>
</body>
</html>
```

## Appendix


localhost:8080
view-source:lo
localhost:8080/MidtermExampl...
view-source:localhost:8080/Mi...

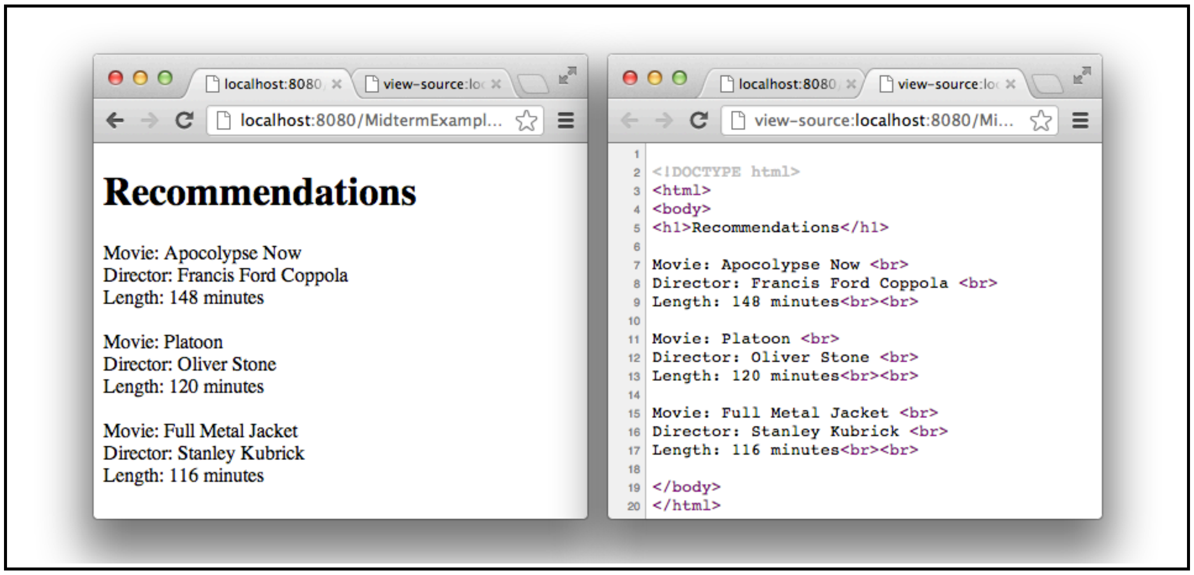
Shopping Cart

- Transformers (2006) - $20
- Pulp Fiction (1994) - $50

Total: $70

```

<!DOCTYPE html>
<html>
<body>

<h1>Shopping Cart</h1>

<ul>

<li>Transformers (2006) - $20

<li>Pulp Fiction (1994) - $50

</ul>

Total: $70

</body>
</html>
```

**Figure 4. Sample output of a display shopping cart feature. On the right is the raw HTML.**


localhost:8080
view-source:lo
localhost:8080/MidtermExampl...
view-source:localhost:8080/Mi...

Recommendations

Movie: Apocolypse Now
Director: Francis Ford Coppola
Length: 148 minutes

Movie: Platoon
Director: Oliver Stone
Length: 120 minutes

Movie: Full Metal Jacket
Director: Stanley Kubrick
Length: 116 minutes

```

<!DOCTYPE html>
<html>
<body>
<h1>Recommendations</h1>

Movie: Apocolypse Now <br>
Director: Francis Ford Coppola <br>
Length: 148 minutes<br><br>

Movie: Platoon <br>
Director: Oliver Stone <br>
Length: 120 minutes<br><br>

Movie: Full Metal Jacket <br>
Director: Stanley Kubrick <br>
Length: 116 minutes<br><br>

</body>
</html>
```

**Figure 5. Sample output of a movie-recommendation feature. On the right is the raw HTML.**

## Java API Excerpts

Class HttpServlet

- protected void doGet(HttpServletRequest request, HttpServletResponse response)
- protected void doPost(HttpServletRequest request, HttpServletResponse response)

Class HttpServletRequest

- String getParameter(String name)
- RequestDispatcher getRequestDispatcher(String path)

Class RequestDispatcher

- void forward(ServletRequest request, ServletResponse response)

Interface HttpSession

- Object getAttribute(String name)
- void setAttribute(String name, Object value)
- boolean isNew()

Class Vector<E>

- public int size()
    - Returns the number of components in this vector.
- public E get(int index)
    - Returns the element at the specified position in this Vector.

## Domain Classes

Class Movie

- public String getTitle()
- public int getLength()
    - Returns number of minutes.
- public String getDirector()
- public int getPrice()
    - Returns dollars (no cents).
- public int getYear()

Class MovieRecommender (Note the method is static.)

- public static Vector<Movie> recommendSimilar(String targetMovieID)
    - Given a movie ID, returns a list of recommended movies.

Class ShoppingCart

- public int size()
    - Returns the number of movies in the cart.
- public Movie get(int i)
    - Returns the i+1th movie in the cart (i.e., starts at 0 like array indices).