

# OLD QUESTIONS PART 1

```
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 [2pts] Imagine that you try to commit your changes into the repository, but you get an out-of-date error.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



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>) request.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() %> minutes <br><br>
```

```
<% } %>
```

```
</body>
```

```
</html>
```



## Appendix

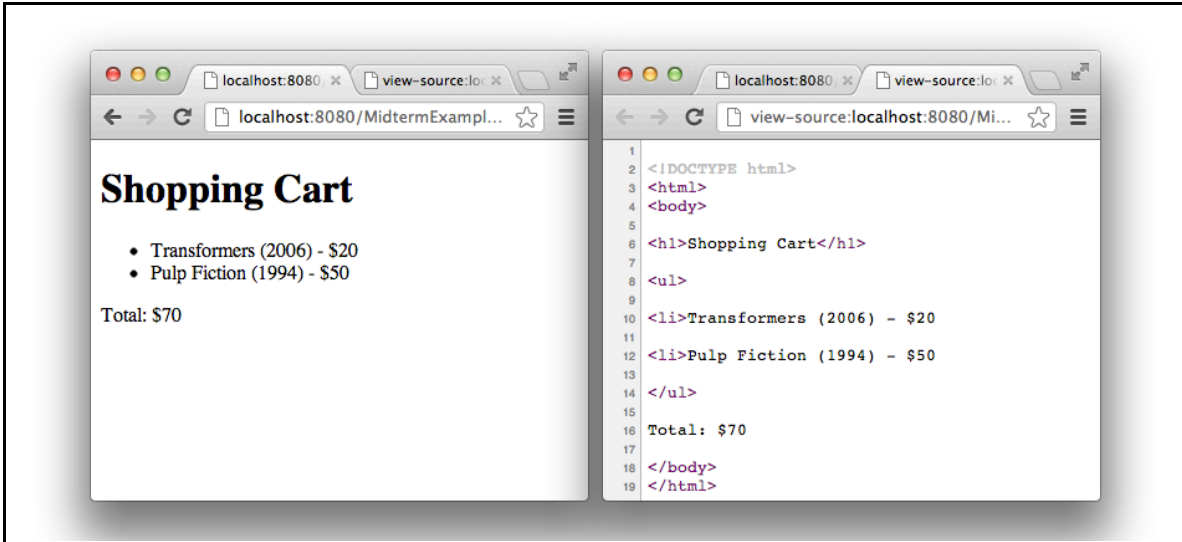


Figure 4. Sample output of a display shopping cart feature. On the right is the raw HTML.

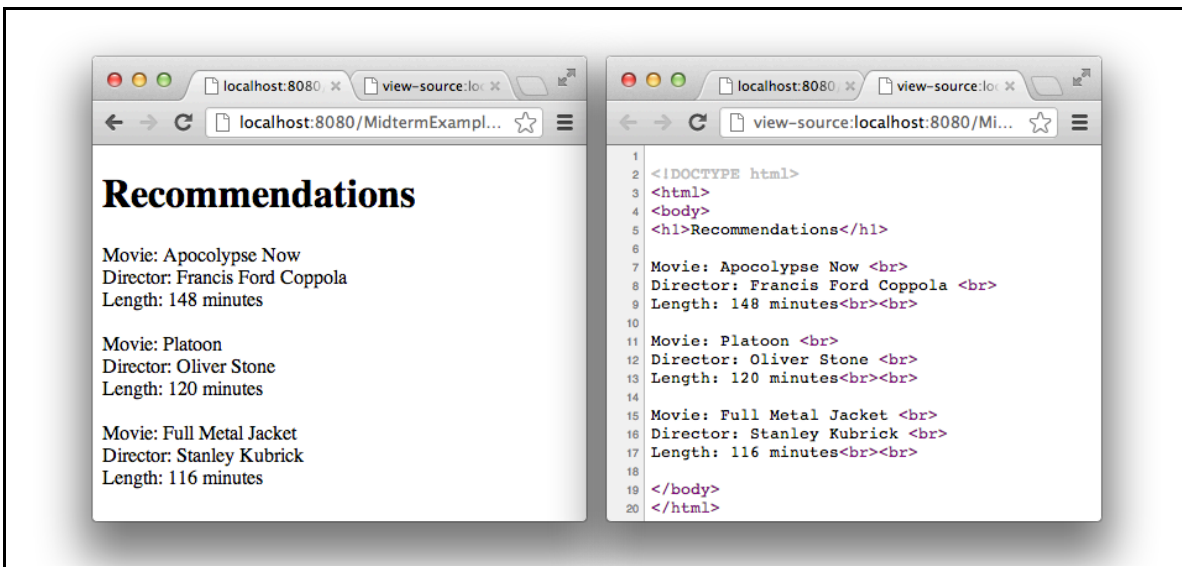


Figure 5. Sample output of a movie-recommendation feature. On the right is the raw HTML.

## 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

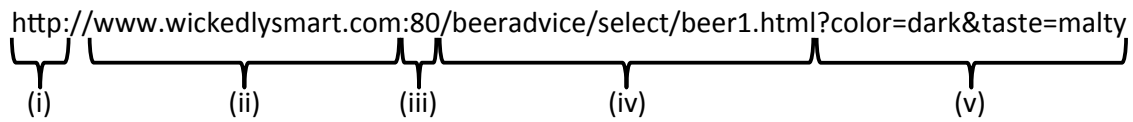
- public int size()
  - Returns the number of movies in the cart.
- public Movie get(int i)
  - Returns the i+1th movie in the cart (i.e., starts at 0 like array indices).

# OLD QUESTIONS PART 2

For each of the following, circle the answer that best answers the question.

1. [3pts] Which of the following is not a problem in software engineering?
  - ~~a. Gathering complete and valid requirements~~
  - ~~b. Estimating the cost of a software project~~
  - ~~c. Removing defects from software~~
  - ~~d. Maintaining software over time~~
  - e. None of the above (i.e., all are problems)
  
2. [3pts] Which of the following is not an agile value?
  - ~~a. Individuals and interactions over processes and tools~~
  - ~~b. Working software over comprehensive documentation~~
  - ~~c. Customer collaboration over contract negotiation~~
  - ~~d. Responding to change over following a plan~~
  - e. None of the above (i.e., all are agile values)
  
3. [3pts] Which of the following problems does iterative development directly address?
  - a. Unstable requirements
  - ~~b. Design erosion~~
  - ~~c. Program comprehension~~
  - ~~d. All of the above~~
  - ~~e. None of the above~~
  
4. [3pts] Which of the following is not true about GET and POST?
  - a. GET is generally idempotent, and POST is generally not
  - b. POST has a data payload, and GET does not
  - c. Both GET and POST requests have request headers
  - d. GET is an HTTP method, and POST is an HTTPS method
  - e. None of the above
  
5. [3pts] Which of the following correctly defines what it means to be an idempotent operation?
  - a. Has side effects
  - b. Has one parameter
  - c. Can only be applied once per session
  - d. Can be applied multiple times without changing the result beyond the initial application
  - e. None of the above

7. [5pts] For the following URL, match the part of the URL with the name for that part.



- |      |                              |                         |
|------|------------------------------|-------------------------|
| i.   | <u>          b          </u> | a. Resource (with path) |
| ii.  | <u>          e          </u> | b. Protocol             |
| iii. | <u>          c          </u> | c. Port                 |
| iv.  | <u>          a          </u> | d. Parameters           |
| v.   | <u>          d          </u> | e. Server               |

8. [3pts] Which SVN client operation creates a working copy?

- a. checkout
- b. commit
- c. update
- d. revert
- e. None of the above

9. [3pts] Which SVN client operation may report a conflict and then do nothing?

- a. checkout
- b. commit
- c. update
- d. revert
- e. None of the above

10. [3pts] Which SVN client operation should you run as the first step toward resolving a conflict, assuming your goal is to do a merge?

- a. checkout
- b. commit
- c. update
- d. revert
- e. None of the above

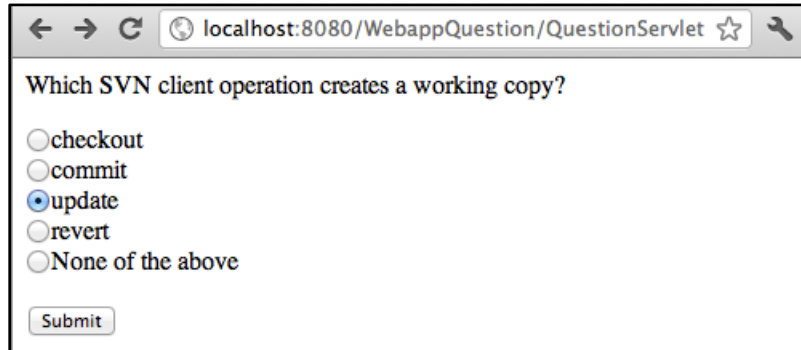
11. [6pts] For each of the following artifacts, tell which role in an MVC architecture the artifact would typically play. Give the full name of the role. (Giving only the letter M, V, or C will earn you only partial credit.)

a. A JSP View

b. A plain old Java class Model

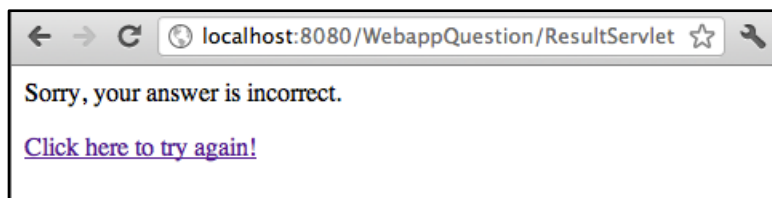
c. A servlet class Controller

12. [15pts] For this problem, you must create a servlet for a quizzing web app. The app already has a `QuestionServlet` servlet that responds to HTTP GET requests with a trivia question that looks like this:



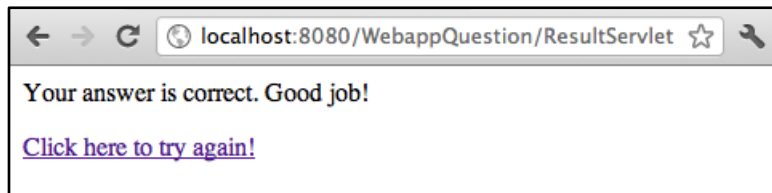
A screenshot of a web browser window. The address bar shows `localhost:8080/WebappQuestion/QuestionServlet`. The page content is a trivia question: "Which SVN client operation creates a working copy?". Below the question are five radio button options: "checkout", "commit", "update" (which is selected), "revert", and "None of the above". At the bottom left of the form is a "Submit" button.

Your job is to create a `ResultServlet` servlet that receives the user's answer when he/she presses the submit button, and that responds with a message like this:



A screenshot of a web browser window. The address bar shows `localhost:8080/WebappQuestion/ResultServlet`. The page content displays the message "Sorry, your answer is incorrect." followed by a blue hyperlink that says "Click here to try again!".

or like this:



A screenshot of a web browser window. The address bar shows `localhost:8080/WebappQuestion/ResultServlet`. The page content displays the message "Your answer is correct. Good job!" followed by a blue hyperlink that says "Click here to try again!".

Relevant excerpts from the Java API and the HTML that `QuestionServlet` generated for the above trivia question is given on page 11. Your `ResultServlet` must use the `QuestionManager` class (described on page 12) to retrieve the correct answers to questions.

You need not include `import` statements or the `web.xml` file in your solution. Just write the Java code for the `ResultServlet` class. Also, I do not expect you to use JSPs in your solution.

Write your answer on the next page.

Write your answer to Question 12 on this page.

```
public class ResultServlet extends HttpServlet {

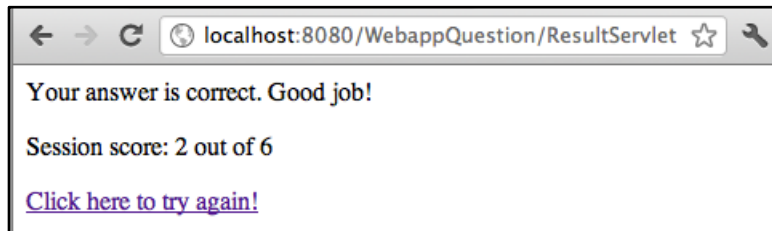
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException
    {
        Enumeration<String> params = request.getParameterNames();
        String questId = params.nextElement();
        String userAns = request.getParameter(questId);
        String correctAns = QuestionManager.getInstance().getAnswerId(questId);
        PrintWriter out = response.getWriter();

        out.println("<html>");
        out.println("<body>");
        out.println("<p>");

        if (userAns.equals(correctAns)) {
            out.println("Your answer is correct. Good job!");
        } else {
            out.println("Sorry, your answer is incorrect.");
        }

        out.println("</p>");
        out.println("<p>");
        out.println("<a href=\"QuestionServlet\">Click here to try again!</a>");
        out.println("</p>");
        out.println("</body>");
        out.println("</html>");
    }
}
```

13. [15pts] For this problem, you must enhance your previous servlet by making it keep track of the user's score for the current session:



You need not rewrite all the code from your previous solution. Represent code that has not changed with ellipses (...). As before, I do not expect you to use JSPs.

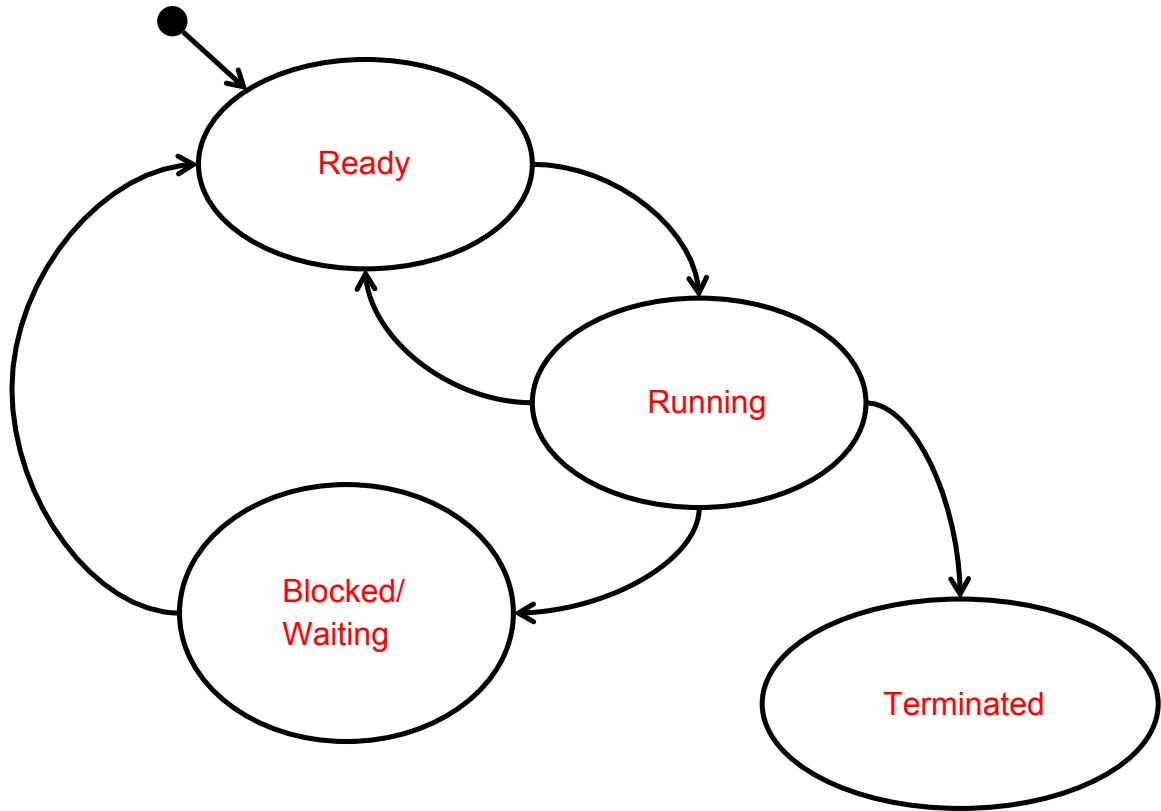
```
public class ResultServlet extends HttpServlet {

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException
    {
        ...
        HttpSession session = request.getSession();

        if (session.isNew()) {
            session.setAttribute("score", new Integer(0));
            session.setAttribute("attempts", new Integer(1));
        } else {
            session.setAttribute(
                "attempts",
                new Integer(((Integer)session.getAttribute("attempts")).intValue()+1)
            );
        }
        ...
        if (userAns.equals(correctAns)) {
            ...
            session.setAttribute(
                "score",
                new Integer(((Integer)session.getAttribute("score")).intValue()+1)
            );
        } else {
            ...
        }
        ...
        out.println("<p>");
        out.println("Session score: " +
            ((Integer)session.getAttribute("score")).toString() +
            " out of " +
            ((Integer)session.getAttribute("attempts")).toString());
        out.println("</p>");
        ...
    }
}
```



14. [8pts] Complete this diagram of thread states in Java by labeling the states.



15. [15pts] List all the different output values that this program might produce.

```
public class MyThread implements Runnable {  
  
    public void run() {  
        MyMainThread.x += MyMainThread.y; // A  
        MyMainThread.y = 5; // B  
    }  
  
}  
  
public class MyMainThread {  
    static public int x = 1;  
    static public int y = 1;  
  
    public static void main(String[] args)  
        throws InterruptedException  
    {  
        Thread myT = new Thread(new MyThread());  
        myT.start();  
        x = 9; // C  
        y += x; // D  
        myT.join();  
        System.out.println("x = " + x);  
        System.out.println("y = " + y);  
    }  
  
}
```

ABCD: x = 9, y = 14  
ACBD: x = 9, y = 14  
ACDB: x = 9, y = 5  
CABD: x = 10, y = 15  
CADB: x = 10, y = 5  
CDAB: x = 19, y = 5

## Relevant Excerpts from the Java API

- Class HttpServlet
  - protected void service(HttpServletRequest req, HttpServletResponse resp)
  - protected void doGet(HttpServletRequest req, HttpServletResponse resp)
  - protected void doPost(HttpServletRequest req, HttpServletResponse resp)
- Interface HttpServletRequest
  - HttpSession getSession()
  - public Enumeration<String> getParameterNames()
  - public String getParameter(String name)
- Interface HttpServletResponse
  - public PrintWriter getWriter()
- Interface HttpSession
  - public boolean isNew()
  - public Object getAttribute(String name)
  - public void setAttribute(String name, Object value)
- Interface Enumeration<E>
  - boolean hasMoreElements()
  - E nextElement()

## Example of HTML Generated by QuestionServlet

```
<html>
<body>
<p>
Which SVN client operation creates a working copy?
</p>
<form method="POST" action="ResultServlet">
<input type="radio" name="q1748" value="a1">checkout<br>
<input type="radio" name="q1748" value="a4">commit<br>
<input type="radio" name="q1748" value="a2">update<br>
<input type="radio" name="q1748" value="a3">revert<br>
<input type="radio" name="q1748" value="a5">None of the above
<p>
<input type="submit" value="Submit">
</p>
</form>
</body>
</html>
```

q1748 is the ID used by QuestionManager for the question. a1–a5 are IDs that QuestionMaker uses for each answer.

## Class QuestionManager

```
public class QuestionManager {
    private static QuestionManager instance = null;

    public static QuestionManager getInstance() {
        if (instance == null) {
            instance = new QuestionManager();
        }
        return instance;
    }

    private QuestionManager() { ... }

    /**
     * @return The ID of the correct answer to questionId.
     */
    public String getAnswerId(String questionId) { ... }

    ...
}
```

QuestionManager follows the singleton pattern, which ensures that there is only one instance of the class in the system. To get the instance of QuestionManager, you must call its static getInstance() method (as opposed to using new).