Grade
/ 25

Jordan University of Science and Technology
Software Engineering Department
SE432 "Software Engineering for Web Applications"

Fall 2021, Midterm Exam **December 4th, 2021**

Name: **ID#:** **Section:** **Serial#:**

Part 1. Short Answer Questions (12 points) [CLO1, CLO2]

1. Why is the Transmission Control Protocol a reliable protocol? (**1 pt**)

   An acknowledgement should be sent to confirm the receipt of packets along with the checksum to make sure that the data is correct.

2. Explain the role of the Domain Name Service (DNS) (**1 pt**)

   To get the IP address of the host

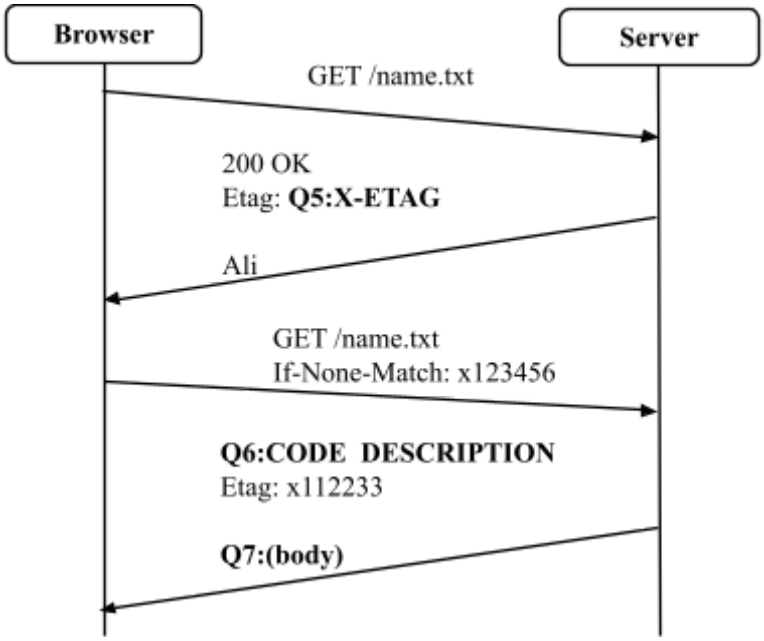
3. Which header is sent with the 302 Moved Temporarily response status? (**1 pt**)

   Location

4. When public caching is used, where does the caching of files occur? (**1 pt**)

   Proxy server

Based on the following scenario, answer questions 5, 6 and 7.



5. What is the value of **X-ETAG** (**1 pt**)

   x123456

6. What is the response Status Code and Description (**CODE DESCRIPTION**) (**1 pt**)

   200 OK

7. Is the **body** in the second response empty? (**1 pt**)

   No, since the E-tag value is new.

8. When using sessions, and in case the browser enables cookies, why is it better to use non-persistent cookies? (**1 pt**)

Because the cookies will be deleted from the browser's memory once the session is over. Thus, the session information is discarded when the session is over (or times out), or if the browser is closed.

9. Briefly, explain the purpose of Filters in web applications. (**1 pt)**

They reduce the complexity of the application by implementing the Chain of Responsibility design pattern. They encourage the separation of concerns design principle. Each filter will be responsible for a specific functionality which reduces the complexity of the servlet.

10. A POST request is sent to host *www.se432-fall2021.com* at *Port* 9000? The request is sent when submitting a form that contains two parameters (*course* and *exam* where the value of *course* is *se432* and the value of *exam* is *midterm exam*) to a page called *simpleexam*. The request is part of the session 123456 where *jsessionid* is sent as a cookie**.** What is the generated HTTP request message**? (3 pts)**

POST /simpleexam HTTP/1.1 0.5pt
Host: www.se432-fall2021.com:9000 0.5pt
Cookie: jsessionid=123456 1pt

course=se432&exam=midterm 1pt

Part 2. Servlet Programming (13 points) [CLO3]

The rate.html and purchase.html forms are used to submit requests to the TotalCostCalculator servlet. You have the Purchase, Purchases, and TaxRate classes.

Requirements:

1. The admin uses the rate.html form to set a new sales tax rate for purchases:
    a. The sales tax rate should be added as an attribute to the ServletContext as a TaxRate object.
    b. Every time the admin submits a new tax rate, it will replace the TaxRate object in the ServletContext object.
    c. When the admin updates the tax rate, he shall get the "Updated the tax rate value successfully!" message along with the rate.html form (see Figure 1 below).
    d. If the admin enters an invalid number or a rate value that is not between 0.05 and 0.20, he should get the "Please enter a valid rate value!" message along with the rate.html form.
2. The user can submit as many purchase costs using the purchase.html form:
    a. The server shall calculate the total cost using the rate in the TaxRate object using the calculateTotal method.
    b. The server shall maintain all the submitted purchase costs on the user session.
    c. When the user submits the purchase cost value, he should receive the Total Cost value, the previous purchases, and the purchase.html form in the response (see Figure 2 below).
    d. When the user submits a purchase cost and in case there is no TaxRate object in the ServletContext, a TaxRate object shall be created and added as an attribute to the ServletContext.

e. When the user enters an invalid number or a negative purchase value he should get the “Please enter a valid cost value!” message along with the purchase.html form

You need to complete the servlet doGet and doPost methods to reflect the above requirements

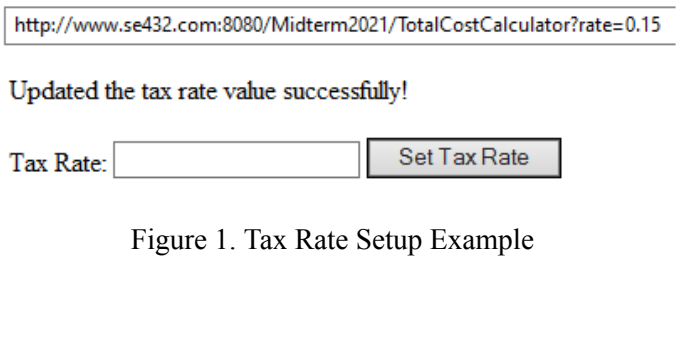


Figure 1. Tax Rate Setup Example

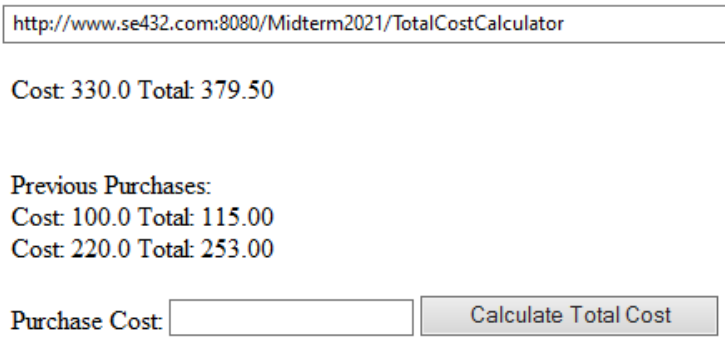


Figure 2. Tax Rate Setup Example

rate.html

```
<form action="TotalCostCalculator" method="get">
   Tax Rate: <input type="text" name="rate"/>
             <input type="submit" value="Set Tax Rate"/>
</form>
```

purchase.html

```
<form action="TotalCostCalculator" method="post">
 Purchase Cost: <input type="text" name="cost"/>
                <input type="submit" value="Calculate Total Cost"/>
</form>
```

```
package models;
public class Purchase {
  private double cost=0.0;
  private double total=0.0;
  public double getCost() {return cost;}
  public void setCost(double cost) {this.cost = cost;}
  public double getTotal() {return total;}
  public void calculateTotal(double rate){total = cost * (1 + rate);}
  public String toString(){
    return "Cost: " + cost + " Total: " + String.format("%.2f", total) + "<br>";
  }
}
```

```
package models;
import java.util.ArrayList;
public class Purchases {
      ArrayList<Purchase> purchases;
      public Purchases(){purchases = new ArrayList<Purchase>();}
      public void add(Purchase p){purchases.add(p);}
      public String print(){
            String s="<br>";
            if(!purchases.isEmpty()) s+="Previous Purchases: <br>";
            for(Purchase p: purchases){s+= p.toString();}
            return s;
      }
}
```

```java
package models;
public class TaxRate {
	private double rate = 0.15;
	public TaxRate(){}
	public TaxRate(double rate){this.rate = rate;}
	public double getRate() {return rate;}
	public void setRate(double rate) {this.rate = rate;}
}
```

```java
package servlets;
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import models.*;

public class TotalCostCalculator extends HttpServlet {

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {

	response.setContentType("text/html");

	PrintWriter out = response.getWriter();   0.5pt

	ServletContext application = request.getServletContext();

	try {


		double rate = Double.parseDouble(request.getParameter("rate"));  0.5pt

		if(rate<0.05 || rate > 0.2) throw new Exception();  0.5pt

		application.setAttribute("tax-rate", new TaxRate(rate)); 1pt

		out.println("Updated the tax rate value successfully!"); 0.5pt


	} catch (Exception e) {

		out.println("Please enter a valid rate value!"); 0.25pt

	}

	finally{

		RequestDispatcher rd = request.getRequestDispatcher("rate.html"); 0.25pt
		rd.include(request, response); 0.25pt

	}
```

```
}

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {


            response.setContentType("text/html");

            PrintWriter out = response.getWriter(); 0.5pt

            ServletContext application = request.getServletContext();

            HttpSession session = request.getSession(); 0.5pt

            try {

                  TaxRate taxRate = (TaxRate)application.getAttribute("tax-rate"); 1pt

                  Purchases purchases = (Purchases)session.getAttribute("purchases"); 1pt

                  if(purchases == null){ 0.5pt
                        purchases = new Purchases(); 0.5pt
                        session.setAttribute("purchases", purchases); 0.5pt
                  }

                  if(taxRate == null){ 0.5pt
                        taxRate = new TaxRate(); 0.5pt
                  }

                  double cost = Double.parseDouble(request.getParameter("cost")); 0.5pt

                  if(cost<0) throw new Exception(); 0.5pt

                  Purchase p = new Purchase(); 0.5pt

                  p.setCost(cost); 0.25pt

                  p.calculateTotal(taxRate.getRate()); 0.25pt

                  out.println(p.toString()); 0.25pt

                  out.println(purchases.toString()); 0.25pt

                  purchases.add(p); 0.5pt

            } catch (Exception e) {

                  out.println("Please enter a valid cost value!"); 0.25pt

            }
            finally{
```

```
            RequestDispatcher rd = request.getRequestDispatcher("purchase.html"); 0.25pt
            rd.include(request, response);  0.25pt

        }


}
```