

Student ID _____ Student Name _____

WAA Midterm Exam

Midterm Exam April 15, 2017

PRIVATE AND CONFIDENTIAL

1. Allotted exam duration is 2 hours.
2. Closed book/notes.
3. No personal items including electronic devices (cell phones, computers, calculators, PDAs).
4. Cell phones must be turned in to your proctor before beginning exam.
5. No additional papers are allowed. Sufficient blank paper is included in the exam packet.
6. Exams are copyrighted and may not be copied or transferred.
7. Restroom and other personal breaks are not permitted.
8. Total exam including questions and scratch paper must be returned to the proctor.

6 blank pages are provided for writing the solutions and/or scratch paper. All 6 pages must be handed in with the exam

BE VERY CAREFUL WITH THE GIVEN 2 HOURS AND USE YOUR TIME WISELY. THE ALLOTTED TIME IS GIVEN FOR EVERY QUESTION.

Write your name and student id at the top of this page.

Question 1: [10 points] {10 minutes}

Select true or false for each of the following statements.

	True or False?
To access parameters sent to a servlet, we should use the Java code <code>request.getAttribute()</code>	False
There is only one instance of a particular Servlet available in the webcontainer	True
Request scope is always thread safe	True
A JSP page is translated into a Java servlet class	True
Every servlet has an init() method	True
Application scope is thread safe	False

Question 2 SCI [5 points] {10 minutes}

Describe how we can relate the concept of JSF to the principles of SCI. Your answer should be about half a page, but should not exceed one page (handwritten). The number of points you get for this question depend on how well you explain the relationship between JSF and the principles of SCI.

Your answer:

Question 3. Servlets [25 points] {25 minutes}

In the servlet labs you wrote a guessnumber servlet according the following description:

Now write a 'guessnumber' servlet. The servlet creates a random number between 1 and 10.

The servlet then asks the user to guess the number.

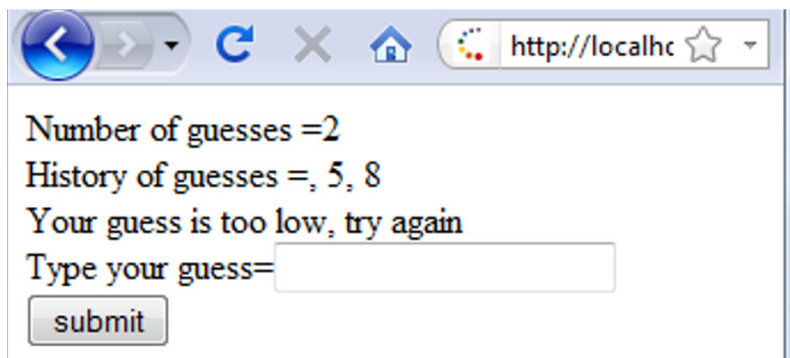
The user enters a number, and the servlet shows if the number was

- ***Too low, try again***
- ***Too high, try again***
- ***Correct, congratulations!***

The code `Math.random()*10;` creates a random number between 1 and 10.

Write the code of this servlet, but now extend the servlet code with the following functionality:

- The servlet also shows the number of guesses made.
- The servlet also shows the history of all guesses made



Number of guesses =2
History of guesses =, 5, 8
Your guess is too low, try again
Type your guess=

Complete the partial given code on the next page:

```

public class guesssservlet extends HttpServlet {

    private int number;

    public void init() {
        number = 8;
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("<body>");
        HttpSession session = request.getSession();
        // both the number of guesses and the history need to be placed in the session
        Integer nrofGuesses = (Integer)session.getAttribute("nrofGuesses");
        List<Integer> history = (List<Integer>)session.getAttribute("history");
        if (nrofGuesses == null){
            nrofGuesses = new Integer(0);
            history = new ArrayList<Integer>();
            session.setAttribute("nrofGuesses", nrofGuesses);
            session.setAttribute("history", history);
        }
        // get the guess
        String sguess = request.getParameter("guess");
        if (sguess != null) {
            int guess = Integer.parseInt(sguess);
            nrofGuesses++;
            session.setAttribute("nrofGuesses", nrofGuesses);
            history.add(new Integer(guess));
            out.println("Number of guesses =" + nrofGuesses + "<br/>");
            out.println("history of guesses =");
            for (Integer i : history){
                out.print(", " + i);
            }
            out.println("<br/>");
            // check the guess
            if (guess == number) {
                out.println("Congratulations, your guess is correct!");
            } else if (guess < number) {
                out.println("Your guess is too low, try higher");
            } else if (guess > number) {
                out.println("Your guess is too high, try lower");
            }
        }

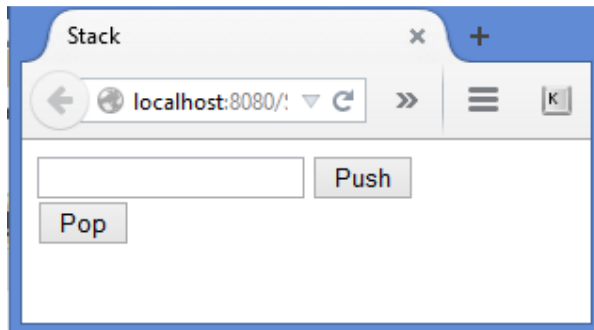
        out.println("<form method=GET action=guesssservlet>");
        out.println("Type your guess =<input type=text name=guess> <br>");
        out.println("<input type=submit value='submit'>");
        out.println("</form>");
    }
}

```

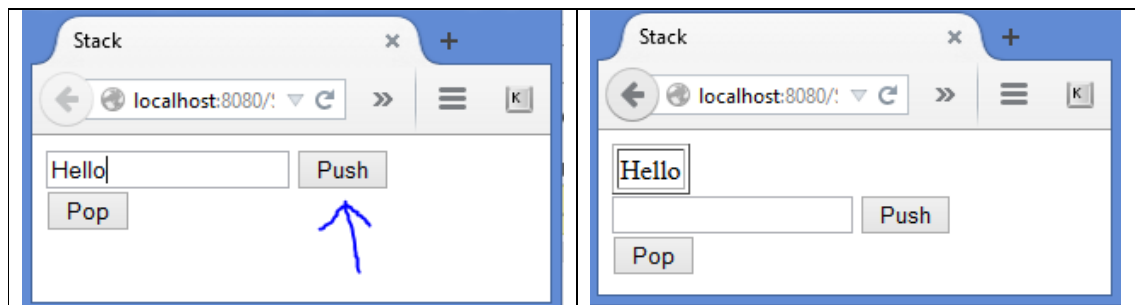
```
        out.println("</body>");  
        out.println("</html>");  
    }  
}
```

Question 4.MVC [30 points] {40 minutes}

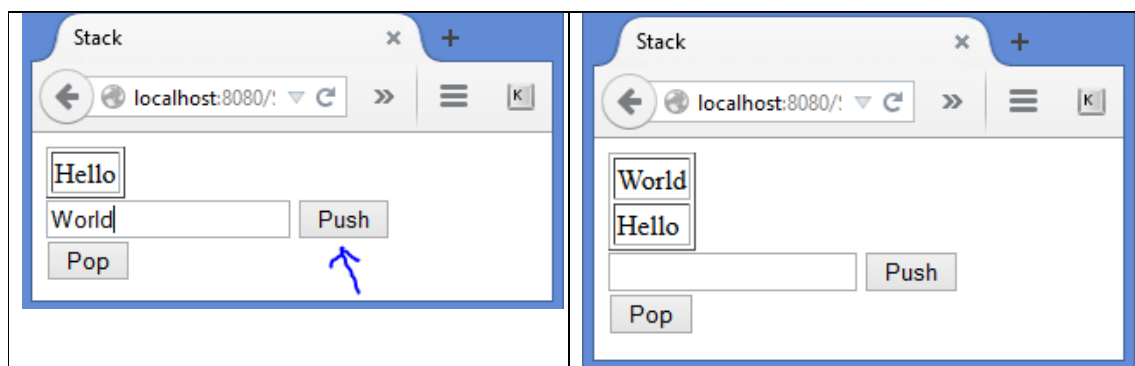
Write a stack application using Servlets and JSP's according the MVC structure. When you start the application you see the following page:



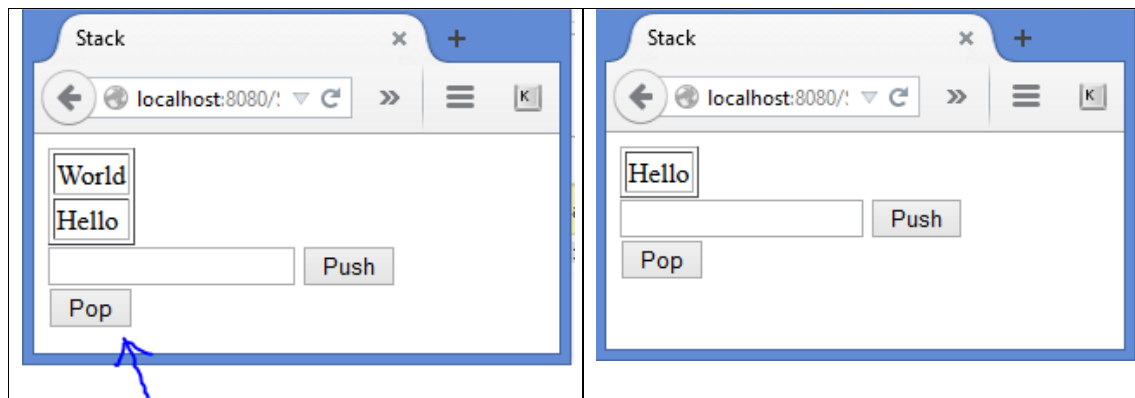
You can push a string on the stack and you can pop a string of the stack. When you start the application, the stack is empty.



First we enter the string 'Hello' and click the Push button. We see that then the String 'Hello' is pushed on the stack.



Then we enter the string 'World' and click the Push button. We see that then the String 'World' is pushed on the top of the stack, and the stack contains the strings 'World' and 'Hello'



When we click the Pop button, we see that the top element is removed from the stack, in this case the string World is removed from the stack. It should be possible to push many elements on the stack or pop elements off the stack, and the application should show the content of the stack in the correct order (the top elements of the stack should be shown as the first element in the table).

Your implementation should follow the following requirements:

1. The application should follow the correct **Model-View-Controller** principles using Servlets, JSP's and Java classes.
2. You are **not** allowed to use JSF.
3. It is **not** allowed to write JAVA code in the JSP page (use JSTL and JSP EL).

Complete the partial given JSP file and write all other necessary classes:

stack.jsp

```
<% @page contentType="text/html" pageEncoding="UTF-8"%>
<% @taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Stack</title>
  </head>
  <body>
    <TABLE BORDER=1>

      <c:forEach var="element" items="${stack.elements}">
        <TR>
          <TD>${element}</TD>
        </TR>
      </c:forEach>
    </TABLE>
    <form method='get' action='PushServlet'>
      <input type='text' name='element' />
      <input type='submit' value='Push' />
    </form>
    <form method='get' action='PopServlet'>
      <input type='submit' value='Pop' />
    </form>
  </body>
</html>
CalculateShippingServlet.java
```

```
@WebServlet(urlPatterns = {"/ShowServlet"})
public class ShowServlet extends HttpServlet {

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

    HttpSession session = request.getSession();
    Stack stack = (Stack) session.getAttribute("stack");
    if (stack == null){
      stack = new Stack();
      session.setAttribute("stack", stack);
    }
    RequestDispatcher view =request.getRequestDispatcher("stack.jsp");
    view.forward(request, response);
  }
}

@WebServlet(urlPatterns = {"/PushServlet"})
```



```

public class PushServlet extends HttpServlet {

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

        // get the element
        String element= request.getParameter("element");

        HttpSession session = request.getSession();
        Stack stack = (Stack) session.getAttribute("stack");
        if (stack == null){
            stack = new Stack();
            session.setAttribute("stack", stack);
        }
        stack.push(element);
        RequestDispatcher view =request.getRequestDispatcher("stack.jsp");
        view.forward(request, response);
    }
}

@WebServlet(urlPatterns = {"/PopServlet"})
public class PopServlet extends HttpServlet {

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

        HttpSession session = request.getSession();
        Stack stack = (Stack) session.getAttribute("stack");
        if (stack == null){
            stack = new Stack();
            session.setAttribute("stack", stack);
        }
        stack.pop();
        RequestDispatcher view =request.getRequestDispatcher("stack.jsp");
        view.forward(request, response);
    }
}

public class Stack {
    private List<String> elements = new LinkedList<String>();

    public boolean isEmpty() {
        return (elements.size() == 0);
    }

    public void push (String object){
        elements.add(0,object);
    }
}

```

```
}

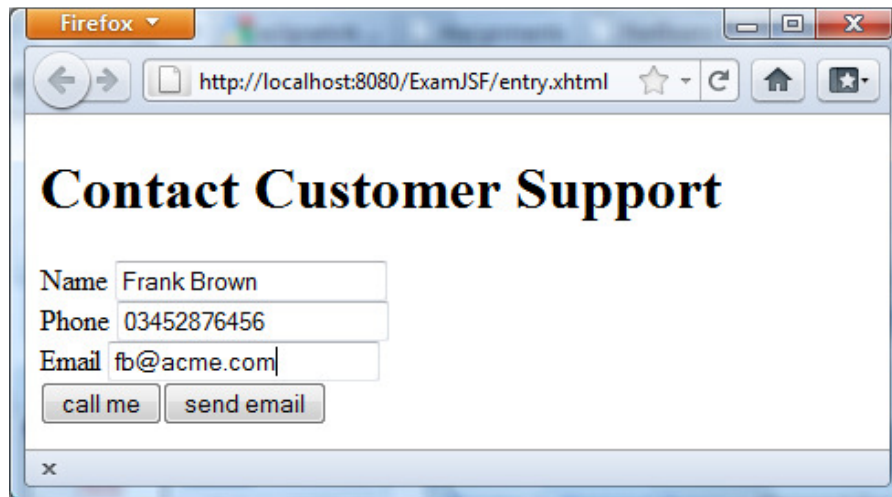
public String pop (){
    String top = top();
    elements.remove(0);
    return top;
}

public String top (){
    if (isEmpty())
        throw new IllegalStateException("stack is empty");
    return elements.get(0);
}

public List<String> getElements() {
    return elements;
}
}
```

Question 5 JSF [30 points] {35 minutes}

Write the following Contact Customer Support application in JSF. The entry page shows the following form:



Firefox

http://localhost:8080/ExamJSF/entry.xhtml

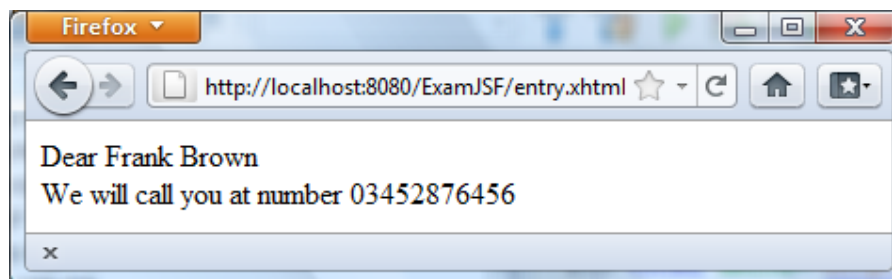
Contact Customer Support

Name

Phone

Email

If you click the *call me* button, you get the following page:



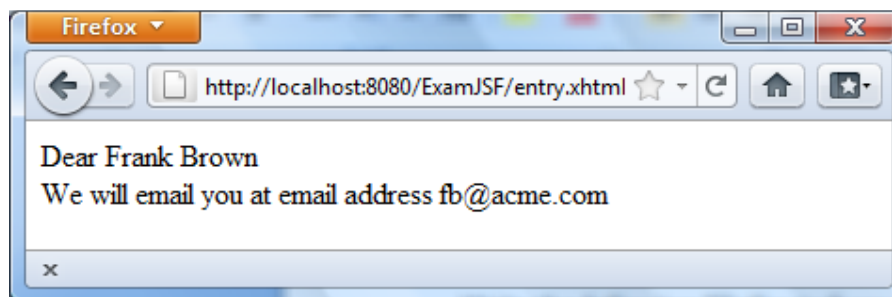
Firefox

http://localhost:8080/ExamJSF/entry.xhtml

Dear Frank Brown

We will call you at number 03452876456

If you click the *send email* button, you get the following page:



Firefox

http://localhost:8080/ExamJSF/entry.xhtml

Dear Frank Brown

We will email you at email address fb@acme.com

Complete the partial given code. Make sure you add all code that is necessary for the correct working of this application.

- The application should follow the correct **Model-View-Controller** principles using JSF.
- You are **not** allowed to use HTML, JSP's or servlets.
- For this JSF application we will use annotation based configuration, so there is no faces-config.xml file. **Add the necessary annotations to the code.**
- **Do NOT write getter and setter methods!**

entry.xhtml:

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html">
  <h:head>
    <title> Contact Customer Support</title>
  </h:head>
  <h:body>
    <h1>Contact Customer Support</h1>
    <h:form>
      Name <h:inputText value="#{customer.name}" /><br/>
      Phone <h:inputText value="#{customer.phone}" /><br/>
      Email <h:inputText value="#{customer.email}" /><br/>
      <h:commandButton value="call me" action="#{customer.callme}" />
      <h:commandButton value="send email" action="#{customer.sendmail}" />
    </h:form>
  </h:body>
</html>
```

call.xhtml:

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html">
  <h:head>
    <title>Phone contact</title>
  </h:head>
  <h:body>
    Dear <h:outputText value="#{customer.name}" /><br/>
    We will call you at number <h:outputText value="#{customer.phone}" /><br/>
  </h:body>
</html>
```

email.xhtml:

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html">
  <h:head>
    <title>Email contact</title>
  </h:head>
  <h:body>
    Dear <h:outputText value="#{customer.name}" /><br/>
    We will email you at email address <h:outputText value="#{customer.email}"
/><br/>

  </h:body>
</html>
```

Customer.java

You don't need to write out getter and setter methods.

```
@ManagedBean
@RequestScoped
public class Customer {
  private String name;
  private String phone;
  private String email;

  public String callme(){
    return "call";
  }
  public String sendmail(){
    return "email";
  }
}
```