Student ID_____Student Name_____

Web Applications Architecture and Frameworks DE

Midterm Exam May 9, 2015

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.**

3 blank pages are provided for writing the solutions and/or scratch paper. All 3 pages must be handed in with the exam

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

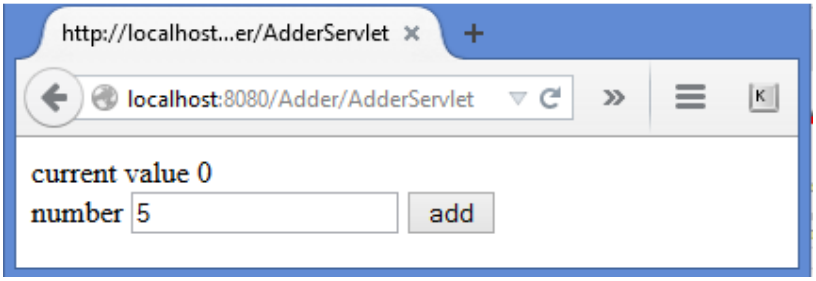
Write your name and student id at the top of this page.

Question 1: [10 points] {10 minutes}
Select true of false for each of the following statements.

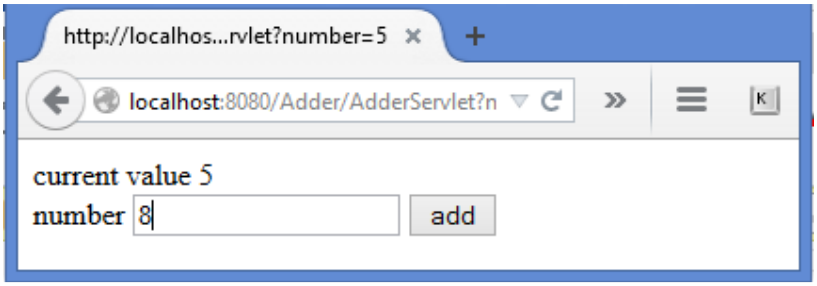
	True or False?
To access parameters sent to a servlet, we should use the Java code **request.getAttribute()**	False
There is only **one** instance of a particular Servlet available in the webcontainer	True
Request scope is always thread safe	True
A JSP page is translated into a Java servlet class	True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

Question 2: [25 points] {30 minutes}

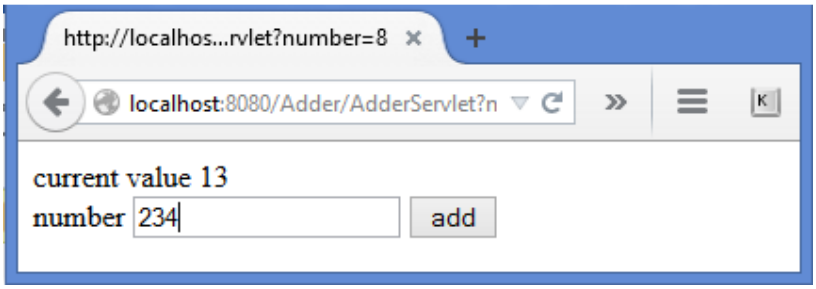
Write the following application with Servlets:



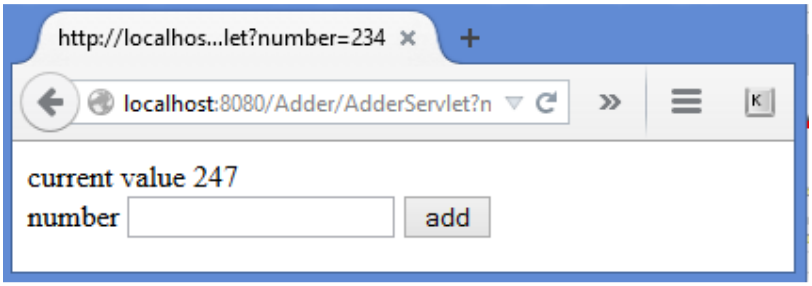
The application is a calculator that can only add numbers. The application first shows the current value of the calculator. You can enter a number (in this case 5) in the number field and when you click the add button you will see that the current value is now 5:



When you enter a number 8 in the number field and when you click the add button you will see that the current value is now 5+8=13:



When you enter a number 234 in the number field and when you click the add button you will see that the current value is now 13+234=247:



Your implementation should follow the following requirements:

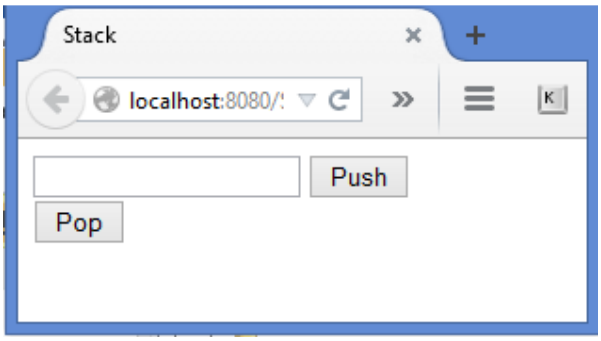
1. You are only allowed to use servlets.
2. You are **not** allowed to use JSP or JSF.
3. Do **not** write getter and setter methods

Complete the partial given code and write all other necessary classes:

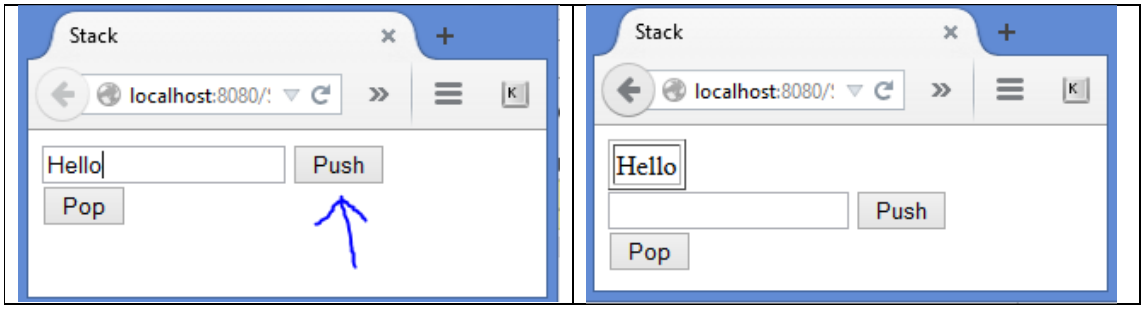
```
@WebServlet(urlPatterns = {"/AdderServlet"})
public class AdderServlet extends HttpServlet {

   @Override
   protected void doGet(HttpServletRequest request, HttpServletResponse response)
         throws ServletException, IOException {
      response.setContentType("text/html");
      PrintWriter out = response.getWriter();

      int number;
      Integer result = null;
      String snumber = request.getParameter("number");
      if (snumber != null) {
         number = Integer.parseInt(snumber);
         HttpSession session = request.getSession();
         result = (Integer) session.getAttribute("result");
         if (result == null) {
            result = new Integer(0);
            session.setAttribute("result", result);
         }
         result += number;
         session.setAttribute("result", result);
      }
      else{
         result = new Integer(0);
      }
       out.println("current value");
       out.println(result);
       out.println("<br/>");
       out.println("<form method='get' action='AdderServlet'>");
       out.println("number  <INPUT TYPE='TEXT' NAME='number'>");
       out.println("<input type='submit' value='add'>");
       out.println("</form>");
   }
}
```

Question 3 [30 points] {50 minutes}

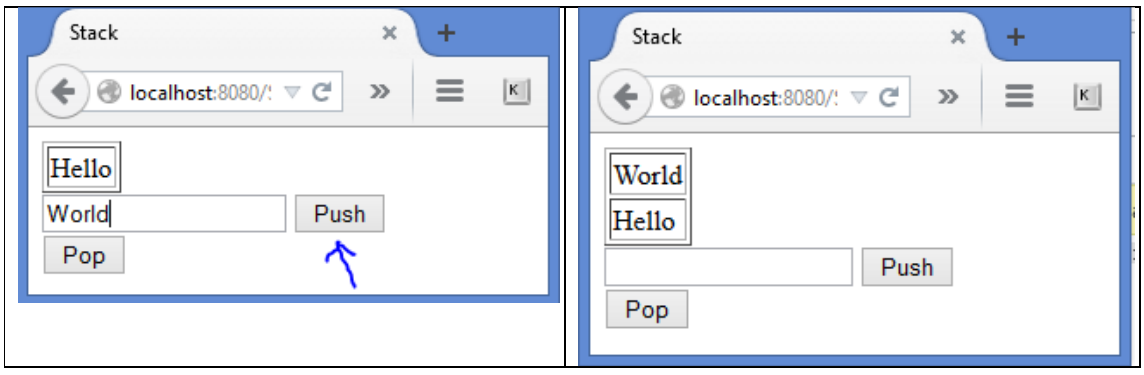
Write a stack application using Servlets and JSP's according the MVC structure. When you start the application you see the following page:



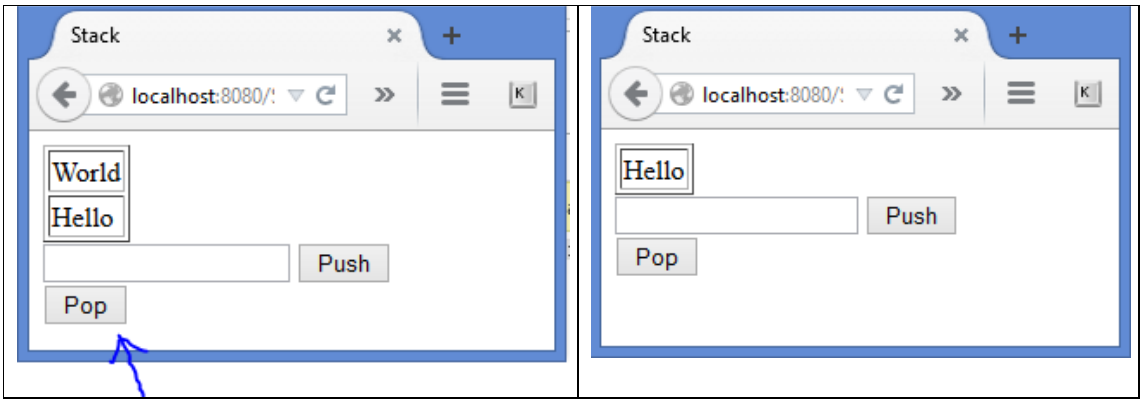
You can push a string on the stack and you can pop a string of the stack. When you start the application, the stack is empty.



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 code 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' />
```

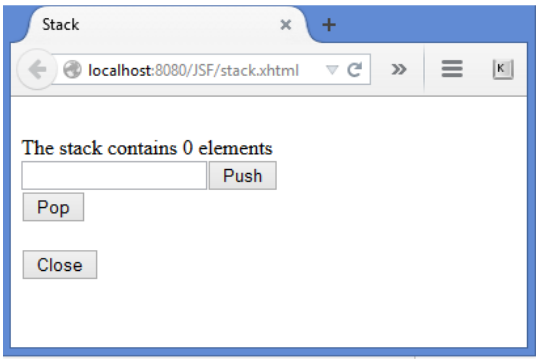
```
        </form>
        <form method='get' action='PopServlet'>
            <input type='submit' value='Pop' />
        </form>
    </body>
</html>
```

CalculateShippingServlet.java

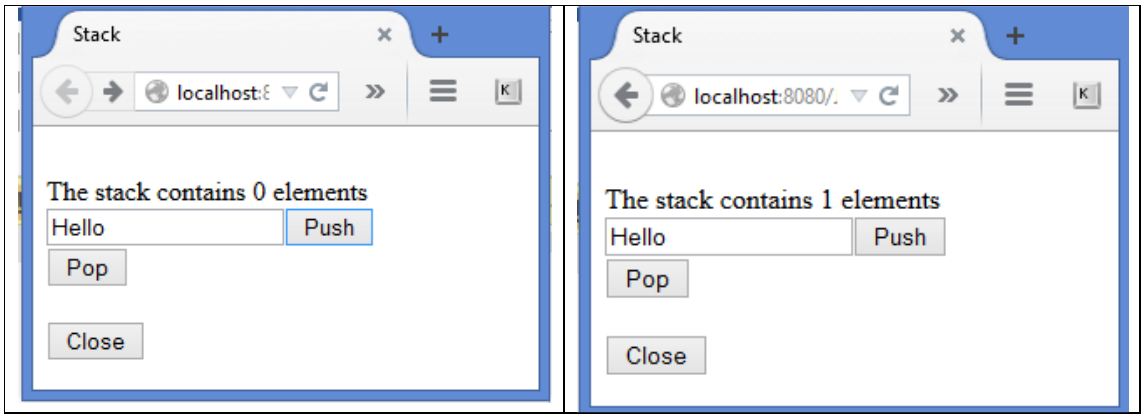
```
@WebServlet(urlPatterns = {"/ShowServlet"})
public class ShowServlet extends HttpServlet {

    @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 4. [30 points] {30 minutes}
Write the following stack application using JSF. The application starts with an empty stack.

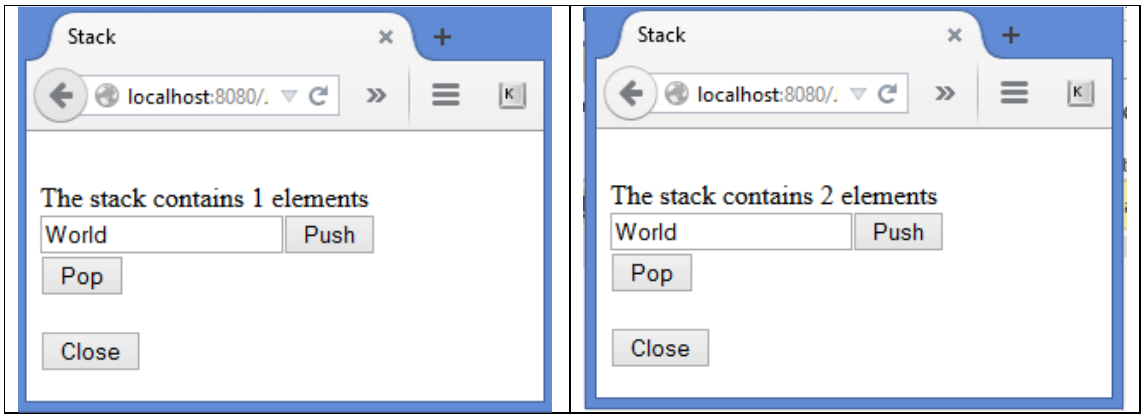


The application allows you to push strings on the stack and/or pop elements of the stack:



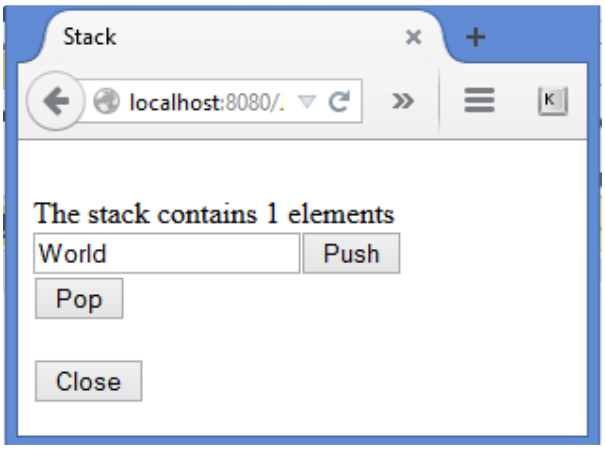
When you enter the string "Hello" and click the Push button, then the application shows that the stack contains 1 element. The content of the stack is now written to the console:
In the console we see now:

INFO: Content of the Stack:
INFO: Hello



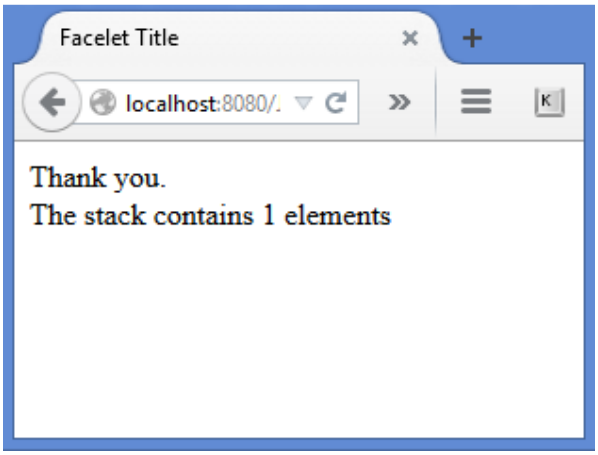
When you enter the string “World” and click the Push button, then the application shows that the stack contains 2 element. The content of the stack is now written to the console:
In the console we see now:

INFO: Content of the Stack:
INFO: World
INFO: Hello



When we click the Pop button, we see that the stack contains only 1 element en the content of the stack is written to the console:

INFO: Content of the Stack:
INFO: Hello



When we click the Close button we see a new “thank you” page that also shows the number of elements on the Stack.

Your implementation should follow the following requirements:

1. The application should follow the correct **Model-View-Controller** principles using JSF.
2. You are **not** allowed to use HTML, JSP's or servlets.
3. You do not need to validate the input.
4. 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**.
5. **Do NOT write getter and setter methods!**

Complete the partial given code and write all other necessary classes:

stack.xhtml :

```
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:h="http://xmlns.jcp.org/jsf/html">
  <h:head>
    <title>Stack</title>
  </h:head>
  <h:body>

    <br/>
    The stack contains <h:outputText value="#{stackBean.stacksize}"/> elements<br/>
    <h:form><h:inputText value="#{stackBean.element}"/>
      <h:commandButton value="Push" action="#{stackBean.push}" />
      <br />
      <h:commandButton value="Pop" action="#{stackBean.pop}" />
      <br /><br />
      <h:commandButton value="Close" action="#{stackBean.close}" />
    </h:form>
  </h:body>
</html>
```

thankyou.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://xmlns.jcp.org/jsf/html">
  <h:head>
    <title>Facelet Title</title>
  </h:head>
  <h:body>
    Thank you.<br/>
    The stack contains <h:outputText value="#{stackBean.stacksize}"/> elements
  </h:body>
</html>
```

```
@ManagedBean
@RequestScoped
public class StackBean {

   private String element;
   private int stacksize;
   @ManagedProperty(value="#{stack}")
   private Stack stack;

   public String push(){
      stack.push(element);
      stacksize=stack.getElements().size();
      printstack();
      return "";
   }

   public String pop(){
      stack.pop();
      stacksize=stack.getElements().size();
      printstack();
      return "";
   }

   public String close(){
      return "thankyou";
   }
   private void printstack(){
      System.out.println("Content of the Stack:");
      List<String> stackList = stack.getElements();
      for (String element : stackList ){
         System.out.println(element);
      }
   }
   public int getStacksize() {
      stacksize=stack.getElements().size();
      return stacksize;
   }

}
```

```
@ManagedBean
@SessionScoped
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;
  }
}
```