## DCUreg

# DUBLIN CITY UNIVERSITY

**SEMESTER 2** **SOLUTIONS** **2014/2015**

**MODULE:** EE417 – Web Application Development

### PROGRAMME(S):

|  |  |  |
| --- | --- | --- |
|  | MEN | MEng in Electronic Systems |
|  | MTC | MEng in Telecommunications Engineering |
|  | ECSA | Study Abroad (Engineering & Computing) |
|  | DME | B.Eng. in Digital Media Engineering |
|  | ICE | BEng Info and Communications Engineering |
|  |  |  |

**YEAR OF STUDY:** 4,C,X

**EXAMINERS:**

|  |  |
| --- | --- |
| Mr David Molloy | (Ext:8426) |
| Prof. Gerard Parr |  |
| Prof. Zhili Sun |  |
| Prof. Sakir Sezer |  |

**TIME ALLOWED:** 3 Hours

**INSTRUCTIONS:** Answer 4 questions. All questions carry equal marks.

**PLEASE DO NOT TURN OVER THIS PAGE UNTIL YOU ARE INSTRUCTED TO DO SO**

The use of programmable or text storing calculators is expressly forbidden.

Please note that where a candidate answers more than the required number of questions, the examiner will mark all questions attempted and then select the highest scoring ones.

***Requirements for this paper (Please mark (X) as appropriate)***

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | |  | |  | |  | |  | | *Log Tables*  *Graph Paper*  *Dictionaries*  *Statistical Tables* | |  | | --- | |  | |  | |  | |  | | *Thermodynamic Tables*  *Actuarial Tables*  *MCQ Only – Do not publish*  *Attached Answer Sheet* |

## QUESTION 1 (Solutions) [TOTAL MARKS: 25]

### Q 1(a) [12 Marks]

**package** test;

**import** org.junit.Test;

**import** **static** org.junit.Assert.assertEquals;

**public** **class** MyTest {

@Test

**public** **void** testSumOk() { // Should succeed

System.***out***.println("MyTest: testSumOk()");

assertEquals("10", Calculator.*sum*("6", "4"));

}

@Test

**public** **void** testSumFailure() { // Should fail

System.***out***.println("MyTest: testSumFailure()");

assertEquals("Error", Calculator.*sum*("a", "b"));

assertEquals("Error", Calculator.*multiply*("a", "b"));

}

@Test

**public** **void** testMultiplyOk() { // Should succeed

System.***out***.println("MyTest: testMultiplyOk()");

assertEquals("24", Calculator.*multiply*("6", "4"));

}

@Test

**public** **void** testMultiplyNull() { // Null value - should fail

System.***out***.println("MyTest: testMultiplyNull()");

assertEquals("Error", Calculator.*multiply*("6", **null**));

}

}

### Q 1(b) [7 Marks]

**package** test;

**class** Calculator {

**public** **static** **void** main(String[] args) {

System.***out***.println("Welcome to the Calculator. This will calculate A\*B, A+B");

**if** (args.length!=2) {

System.***out***.println("Please provide the format: java Calculator 20 4");

System.*exit*(0);

}

**else** {

System.***out***.println("You provided A=" + args[0] + " and B=" + args[1]);

System.***out***.println("Multiple = " + *multiply*(args[0], args[1]));

System.***out***.println("Sum = " + *sum*(args[0], args[1]));

}

}

**public** **static** String sum(String a, String b) {

**try** {

**int** num1 = (**new** Integer(a)).intValue();

**int** num2 = (**new** Integer(b)).intValue();

**return** (num1 + num2) + "";

} **catch** (Exception e) { **return** "Error"; }

}

**public** **static** String multiply(String a, String b) {

**try** {

**int** num1 = (**new** Integer(a)).intValue();

**int** num2 = (**new** Integer(b)).intValue();

**return** (num1 \* num2) + "";

} **catch** (Exception e) { **return** "Error"; }

}

}

### Q 1(c) [4 Marks]

**Software as a Service:** In the SaaS business model, consumers are provided access to use the provider's applications, which are running on a cloud infrastructure.  The Cloud providers manage the infrastructure which runs the software, but the client simply uses a thin client interface, such as a web browser to use the software.  Clients are provided with access to application software but do not have to worry about installation and set up.  Clients typically pay for the software on a "pay-as-you-go" basis or at no charge, where other streams of revenue (such as advertising) can be generated.

**Examples:**Google Apps, Dropbox, Facebook, Twitter, any web application on a cloud infrastructure

**Infrastructure as a Service:** IaaS is the foundation block of cloud computing.  Rather than purchasing or renting space in a costly datacenter, cloud consumers rent space in a virtual data center from a cloud provider.  With IaaS the "raw materials" are provided and consumers will pay for only the resources they consume.  These resources include CPU cores, memory, hard disk storage and bandwidth.  Consumers are provided with the capability of modifying the resources available to them, typically through interfaces provided by the cloud provider.

### Examples: Amazon EC2, Rackspace, Windows Azure

### Q 1(d) [2 Marks]

## URLRewriting allows the passing of a session id by rewriting the URL to pass details, either in the format /app/function/ABCD123 or /app/function?id=ABCD123. Unfortunately, it will not work for static documents and will only function for a series of dynamic documents or forms.

## [End of Solution for Question 1]

## 

## QUESTION 2 (Solutions) [TOTAL MARKS: 25]

### Q 2(a) [10 Marks]

**package** edu.ee.beans;

**import** java.util.Date;

**import** javax.persistence.Entity;

**import** javax.persistence.GeneratedValue;

**import** javax.persistence.Id;

**import** javax.persistence.Table;

@Entity

@Table (name="Accounts")

**public** **class** Account {

**private** **int** id;

**private** **boolean** active;

**private** String accountType;

**private** String currency;

**private** **float** balance;

**private** Date creationDate;

**public** Account(**boolean** active, String accountType, String currency,

**float** balance, Date creationDate) {

**super**();

//this.id = id;

**this**.active = active;

**this**.accountType = accountType;

**this**.currency = currency;

**this**.balance = balance;

**this**.creationDate = creationDate;

}

**public** Account() { **super**(); }

@Id

@GeneratedValue

**public** **int** getId() { **return** id; }

**public** **void** setId(**int** id) { **this**.id = id; }

**public** **boolean** isActive() { **return** active; }

**public** **void** setActive(**boolean** active) { **this**.active = active; }

**public** String getAccountType() { **return** accountType; }

**public** **void** setAccountType(String accountType) {

**this**.accountType = accountType;

}

**public** String getCurrency() { **return** currency; }

**public** **void** setCurrency(String currency) { **this**.currency = currency; }

**public** **float** getBalance() { **return** balance; }

**public** **void** setBalance(**float** balance) { **this**.balance = balance; }

**public** Date getCreationDate() { **return** creationDate; }

**public** **void** setCreationDate(Date creationDate) {

**this**.creationDate = creationDate;

}

}

### Q 2(b) [7 Marks]

### Modify the HibernateUtil to add the main method to do the following:

**public** **static** **void** main(String args[]) {

System.***err***.println("Recreating database!");

HibernateUtil.*recreateDatabase*();

Session session = *beginTransaction*();

Account account1=**new** Account(**true**,"Current","Euro",(**float**)123.45,**new** Date());

session.save(account1);

Account account2=**new** Account(**true**,"Savings","Euro",(**float**)43.05,**new** Date());

session.save(account2);

Account account3=**new** Account(**false**,"Current","Sterling",(**float**)0,**new** Date());

session.save(account3);

*commitTransaction*();

System.***err***.println("Completed!");

}

### Q 2(c) [4 Marks]

* **Top Down Web Service**: A top down web service is one where you create the WSDL file and use this file to automatically generate the Java classes from it
* **Bottom Up Web Service:**A bottom up web service is where you create a Java bean and then use a web services wizard to automatically generate the WSDL and publish the web service

### Q 2(d) [4 Marks]

The WSDL is a document written in XML, describing a web service, under the following:

* **What a service does -**the methods that the service provides
* **How a service is accessed -**details of the data formats and protocols necessary to access a service operations
* **Where a service is located -**details of the protocol specific network address, such as a URL

## [End of Solution for Question 2]

## QUESTION 3 (Solutions) [TOTAL MARKS: 25]

### Q 3(a) [17 Marks]

***Login.html (4 Marks)***

<html>

<head>

<title>Secure Site</title>

</head>

<body>

<h2>Please login:</h2>

<form method=*"POST"* action=*"/temp/LoginServlet"*

name=*"myform"*>

<br/>Username: <input name=*"username"* />

<br/>Password: <input type=*"password"* name=*"password"* />

<br/>

<input type=*"submit"* value=*"Login"* />

</form>

</body>

</html>

***LoginServlet.java (7 Marks)***

**package** org.dcu;

**import** java.io.\*;

**import** javax.servlet.ServletException;

**import** javax.servlet.annotation.WebServlet;

**import** javax.servlet.http.Cookie;

**import** javax.servlet.http.HttpServlet;

**import** javax.servlet.http.HttpServletRequest;

**import** javax.servlet.http.HttpServletResponse;

@WebServlet("/LoginServlet")x`

**public** **class** LoginServlet **extends** HttpServlet {

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

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

response.setContentType("text/html");

PrintWriter out = response.getWriter();

String username = validateUser(request.getParameter("username"), request.getParameter("password"));

**if** (username!=**null**) { // login succeeded

out.println("Welcome " + username + ". You are now logged in!");

out.println("<br/><br/>Please click here to go to the <a href=\"AccountServlet\">AccountServlet page</a>");

Cookie cookie = **new** Cookie("username", username);

cookie.setMaxAge(3600);

response.addCookie(cookie);

}

**else** { // login failed

response.sendRedirect("login.html");

}

out.close();

}

**private** String validateUser(String username, String password) {

**if** ((username.equals("test"))&&(password.equals("test"))) {

**return** "test";

} **else** **return** **null**;

}

}

***AccountServlet.java (6 Marks)***

**package** org.dcu;

**import** java.io.\*;

**import** javax.servlet.ServletException;

**import** javax.servlet.annotation.WebServlet;

**import** javax.servlet.http.\*;

@WebServlet("/AccountServlet")

**public** **class** AccountServlet **extends** HttpServlet {

**public** **void** doGet(HttpServletRequest req, HttpServletResponse res)

**throws** ServletException, IOException {

res.setContentType("text/html");

String username = **null**;

PrintWriter out = res.getWriter();

Cookie[] cookies = req.getCookies();

**if** (cookies != **null**) {

**for** (**int** i=0; i< cookies.length; i++) {

String name = cookies[i].getName();

String value = cookies[i].getValue();

**if** (name.equals("username")) username = value;

}

}

**if** (username==**null**) { // not logged in or cookie timed out

res.sendRedirect("login.html");

} **else** { // logged in and active session

out.println("This is your secure account page! Welcome " + username);

out.println("If you are seeing this, you logged in!");

}

out.close();

}

}

### Q 3(b) [4 Marks]

## Asynchronous JavaScript and XML (Ajax) is a technique making the user interfaces of web applications more responsive and interactive. The functionality that is newly available to end-users is "in-page replacement". In-page replacement is the ability of a web page to change elements of that web page, using data from a web server, without the need to totally redraw itself. This is commonly seen in search suggestions, canvas manipulation (e.g. Google maps), drag and drop etc.

### Q 3(c) [4 Marks]

**Database Security** – one of the issues associated with providing a database in a two-tier system is that the database needs to be accessible from the location of every possible user of the system. This means that the database is unnecessarily exposed and can not be effectively firewalled. Three-tier systems, on the other hand, tend to hide their databases behind a server firewall, so that only the application server (and some local DBA machines) are allowed access.

**Code Protection –** on two tier systems, the code typically takes the form of a client-side application, such as an executable (.exe) file or a Java (.class) applet. For this to work, the binary version of the code must be downloaded to every user. While this is not a problem in itself, it means that there are many binary copies of the file in existence and malicious users may attempt to decompile the code to steal it or to discover vulnerabilities or backdoors. On a three-tier system this cannot occur as the server code is never downloaded to the client.

## [End of Solution for Question 3]

## 

## QUESTION 4 (Solutions) [TOTAL MARKS: 25]

### Q 4(a) [10 Marks]

Solution of question 4 part a

<html>

<head>

<title>Comparison Question</title>

<script type="text/javascript">

function compareValues(a,b) {

if (a>b)

document.getElementById("result").innerHTML = "First field > Second field";

else if (a==b)

document.getElementById("result").innerHTML = "First field equals Second field";

else

document.getElementById("result").innerHTML = "First field < Second field";

}

</script>

</head>

<body>

<h2>Comparison Question</h2>

<form name="form1">

First Number:

<select name="valueA" id="valA">

<option>1</option>

<option>2</option>

<option>3</option>

<option>4</option>

<option>5</option>

</select>

<br/><br/>

Second Number:

<select name="valueB" id="valB">

<option>1</option>

<option>2</option>

<option>3</option>

<option>4</option>

<option>5</option>

</select>

<br/><br/>

<input type="submit" onclick="compareValues(document.form1.valueA.value,document.form1.valueB.value);return false;" />

<div id="result">Result of comparison</div>

</form>

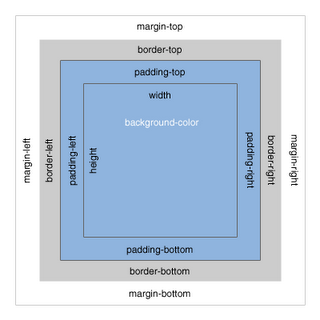
</body>

</html>

**[Question: 10 Marks]**

### Q 4(b) [5 Marks]

The box model is important when laying out CSS as there are a number of CSS properties which can affect the overall size of the “box” containing a HTML element. These are width, height, padding, margin and border.



When attempting pixel perfect layout using CSS it is important to understand the box model as any modification of any of these five elements can have a significant effect on any web page which is designed.

(5 Marks, 3 diagram, 2 description)

### Q 4(c) [3 Marks]

### The World Wide Web Consortium (W3C) is the main international standards organization for the World Wide Web (abbreviated WWW or W3). W3C tries to enforce compatibility and agreement among industry members in the adoption of new standards defined by the W3C. Incompatible versions of HTML are offered by different vendors, causing inconsistency in how Web pages are displayed. The consortium tries to get all those vendors to implement a set of core principles and components which are chosen by the consortium.

### Q 4(d) [4 Marks]

A few changes would need to be made in this situation:

* Firstly, we would need to download a new driver designed to work with the MySQL database. Assume that this is called **MySQL-Connector.jar**.
* We would then need to replace the Oracle drivers file in our application with this new database driver.
* We would then need to change a few lines of our code wherever a connection is being created to indicate that we are using a different driver type.
* Finally, there could be database differences in the application of SQL and we may need to debug some statements, especially when creating the database tables again in the new database.

### Q 4(e) [3 Marks]

Less changes are needed in this situation, but still a couple:

* We would need to download and use the new MySQL-Connector.jar file
* We would need to make some configuration changes in our hibernate.cfg.xml file to indicate the new database settings and dialect we will use.

Hibernate should handle the other aspects on our behalf and allow the automatic creation of our database structures under MySQL.

## [End of Solution for Question 4]

## QUESTION 5 (Solutions) [TOTAL MARKS: 25]

### Q 5(a) [13 Marks]

1. **(10 marks)**

CREATE TABLE CUSTOMERS

(ID INTEGER,

SURNAME VARCHAR(30) NOT NULL,

FIRSTNAME VARCHAR(30) NOT NULL,

ADDRESS VARCHAR(200),

COUNTRY VARCHAR(30) DEFAULT 'Ireland',

EMAIL VARCHAR(80),

PHONE VARCHAR(25),

PRIMARY KEY (ID)

)

CREATE TABLE VEHICLES (

REGISTRATION VARCHAR(2) NOT NULL,

VEHICLETYPE VARCHAR(20) NOT NULL,

BRAND VARCHAR(80) NOT NULL,

MODEL VARCHAR(80) NOT NULL,

SEATS INTEGER,

PRIMARY KEY (REGISTRATION)

)

CREATE TABLE RENTALS (

RENTALID INTEGER,

STARTDATE VARCHAR(20) NOT NULL,

ENDDATE VARCHAR(20) NOT NULL,

CUSTOMERID INTEGER,

VEHICLEID VARCHAR(80),

PRIMARY KEY (RENTALID),

CONSTRAINT CONSTNAMEM1

FOREIGN KEY (CUSTOMERID)

REFERENCES CUSTOMERS,

CONSTRAINT CONSTNAMEM2

FOREIGN KEY (VEHICLEID)

REFERENCES VEHICLES

)

1. **(3 Marks)**

Select c.firstname, c.surname, v.registration, v.brand, v.model, r.startdate, r.enddate from customers c, vehicles v, rentals r

Where c.id=r.customerid and v.registration=r.vehicleid

### Q 5(b) [7 Marks]

----- rentals.xml -----

<?xml version="1.0"?>

<!DOCTYPE rentals SYSTEM "rentals.dtd">

<rentals>

<rental>

<customer>

<firstname>Joe</firstname>

<surname>Smith</surname>

<address>123 Fake Street, Springfield</address>

<phone>+12345678</phone>

<email>joe.smith@gmail.com</email>

</customer>

<vehicle>

<registration>15 D 12345</registration>

<vehicletype>Saloon Car</vehicletype>

<brand>Toyota</brand>

<model>Corolla</model>

<seats>5</seats>

</vehicle>

<startdate>15/05/15</startdate>

<enddate>20/05/15</enddate>

<id>1234</id>

</rental>

<rental>

<customer>

<firstname>Joe</firstname>

<surname>Smith</surname>

<address>123 Fake Street, Springfield</address>

<email>joe.smith@gmail.com</email>

</customer>

<vehicle>

<registration>14 D 55667</registration>

<vehicletype>Van</vehicletype>

<brand>Toyota</brand>

<model>Hi-Ace</model>

<seats>2</seats>

</vehicle>

<startdate>1/06/15</startdate>

<enddate>12/06/15</enddate>

<id>1234</id>

</rental>

</rentals>

### Q 5(c) [5 Marks]

------ users.dtd -------

<!ELEMENT rentals ( rental\*) >

<!ELEMENT rental (customer,vehicle, startdate, enddate,id) >

<!ELEMENT customer (firstname, surname, address, phone?, email?)>

<!ELEMENT vehicle (registration, vehicletype, brand, model, seats?)>

<!ELEMENT startdate (#PCDATA ) >

<!ELEMENT enddate ( #PCDATA ) >

<!ELEMENT id (#PCDATA ) >

<!ELEMENT firstname ( #PCDATA ) >

<!ELEMENT surname (#PCDATA ) >

<!ELEMENT address ( #PCDATA ) >

<!ELEMENT phone (#PCDATA ) >

<!ELEMENT email ( #PCDATA ) >

<!ELEMENT registration (#PCDATA ) >

<!ELEMENT vehicletype ( #PCDATA ) >

<!ELEMENT brand ( #PCDATA ) >

<!ELEMENT model ( #PCDATA ) >

<!ELEMENT seats ( #PCDATA ) >

### [End of Solution for Question 5]

## [END OF SOLUTIONS]