Semester 1 Examinations, 2015/2016

Exam Code	3BCT1
Exam	3rd University Examination in Computer Science and Information Technology
Module Code	CT5106
Module	Software Engineering 2
Paper No.	1
External Examiner **Internal Examiners**	Dr. John Power Prof. Gerard. Lyons Dr. Michael Madden Dr. Owen Molloy*
Instructions	You must answer **Question 1** (60 marks) **and any other 2 questions** (20 marks each).
Duration	2hrs
No. of Answer Books	1
Requirements	None
No. of Pages	5

1) ***[60 marks]*** *(you must answer this question)* Answers should be concise. Answer all parts. Each part is worth [**3]** marks.

1.1. Explain the difference between *functional* and *non-functional* requirements.

1.2. Give examples of the <<*uses*>> and <<*extends*>> stereotypes in UML.

1.3. If you were asked to draw a use case diagram for a hotel website, including an online booking system, list 6 different actors which you might identify.

1.4. Explain the purpose of the urlPatterns annotation as found in servlets, for example:

```
@WebServlet(name = "dataServlet", urlPatterns = {"/dataServlet"})
```

1.5. Explain the difference in scope between the *request*, *session* and *application* in JSP / Servlet applications.

1.6. Add the JPA annotations @Entity, @Table, @Column, @Id to the following Java bean class:

```
public class Product implements Serializable
{
private int id;
private String name;
}
```

1.7. What is the purpose of a Daily Scrum meeting in Agile, and what questions should scrum team members answer?

1.8. What information should a *Sprint Backlog* contain?

1.9. If you are using Planning Poker for estimation in an Agile project, what are the steps that the team follow in playing?

1.10. Describe briefly 3 methods of *requirements elicitation*.

1.11. Draw a simple class diagram to illustrate *Association* and *Generalisation* relationships.

1.12. Use examples to explain the difference between *composition* and *aggregation* relationships between classes.

1.13. Explain, using a simple diagram, how the Model View Controller architecture is implemented in a Java Enterprise application, such as the one you implemented in your group project.

1.14. Explain the meaning of the following terms used in JUnit:

a) *@Test*

b) *@BeforeClass*

c) *Assert*

1.15. What is the difference between *nice* mocking and *strict* mocking, when using a mocking framework such as EasyMock?

1.16. Assuming the following lines of code are executed in a servlet, and that the request is then dispatched to a JSP page, write the JSP code necessary to print out the users username and email address:

```
u1 = new User();
u1.username = “JohnS”;
u1.email = “john.s@bigmail.com”;
request.setAttribute (“user”, u1);
```

1.17. Explain the difference between the following 2 lines of JSP code, and what will happen when they are executed:

```
<%! int numVisits1 = 0; %>
<% int numVisits2 = 0; %>
```

1.18. Explain the meaning of the following terms in Ant build files:

a) *target*

b) *destfile*

c) *srcdir*

1.19. Explain the following Project Management terms (as used in Gannt charts):

a) *Critical Path*

b) *Lead Time*

c) *Task Dependencies*

1.20. Explain briefly how factors such as non-functional requirements, and environmental factors, are taken into account when using estimation techniques such as Function Point Analysis and Use Case Point Analysis.

2) ***[20 marks]*** Construct a Class Diagram for the order processing system described below.

- Each order consists of one or more order lines.
- Each order line corresponds to a particular product. Each order line has a product description, price, quantity and a field that indicates the status of this order line.
- A product can appear on many orders.
- The status of an order line can be either *filled* (sufficient quantities of the item are in stock), *on-order* (extra stock must be ordered to fulfill the order) or *cancelled*.
- The status of an order can be *pending* (initial state, while stock levels are being checked), *filled* (the order has been packaged ready for shipping), *shipped* (received by the customer but not paid), *closed* (the customer has received the goods and paid the amount due), or *cancelled*.
- A customer can place many orders, but each order belongs to only one customer.
- Each order has an order number, date and customer number.
- Customers can be either businesses or individuals.
- All customers have a name and address. Business customers also have a contact name, credit rating and available credit.
- Individual customers may have a credit card number on file.

3) ***[20 marks]*** You have been asked to develop an online shopping application, using Java Beans, Servlets, Java Server Pages and Session Beans as Facades between the entity classes and the persistence layer.

a) ***[10 marks]*** Draw a diagram to illustrate the architecture of the system, and explain the role of the different layers and how they interact with each other.

b) ***[10 marks]*** Explain, using a UML sequence diagram, the sequence of object interactions necessary to fulfil use cases such as adding a product item to a shopping cart. Annotate your diagram with notes as needed.

.

4) ***[20 marks]*** Assume you have a Java Bean class, called Product, which has the following properties (you can assume the getters and setters are also already written):

```
private String name;
private String description;
private double price;
private int ID;
```

Assume also that you are given initial code for a servlet, as show below:

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

Product p1 = new Product("Whiteboard", "Just a white board",
50.00, 101);
Product p2 = new Product("Stapler", "Just a staple stapler",
10.00, 102);
Product p3 = new Product("Chair", "Standard office chair",
40.00, 103);
Product p4 = new Product("Lamp", "Anglepoise!", 25.00, 104);
```

(a) ***[8 marks]*** Finish the java code for the servlet. The servlet code must be finished so that it adds the products to a List object. The List object must be added to the session as an attribute, and control forwarded to a JSP page.

(b) ***[12 marks]*** Write the JSP page code, where the products will be displayed as shown (you may ignore styling).

Product Catalogue

Code	Name	Description	Price
101	Whiteboard	Just a white board	50.0
102	Stapler	Just a staple stapler	10.0
103	Chair	Standard office chair	40.0
104	Lamp	Anglepoise!	25.0