Hand out date: 24.X.2018 Hand in date: 7.XII.2018 (Demonstration: by week starting 3rd Dec)

This is an assessed piece of group coursework, it is therefore essential to be completed and handed-in on time. If you are unclear about any aspect of the assignment, including the assessment criteria, please raise this at the first opportunity. The usual regulations apply to a late submission of work. The submitted application must be in Java (using Java NetBeans IDE) to be marked. During the demonstration (by week 12 – week starting 3rd December, in your lab session) **you have to submit a memory stick** with your source code and Java NetBeans project files with **your group number** on it.

The coursework you submit should be your group work. If your coursework includes other people's ideas and material, they must be properly referenced or acknowledged. Failing to do so intentionally or unintentionally constitutes plagiarism. The University treats plagiarism as a serious offence.

ORDER SYSTEM FOR A BOX-SELLING COMPANY

The Chinese invented cardboard in the 1600s and the English created the first commercial cardboard box in 1817. "*FlexBox*" is a company producing variety of boxes for packaging wide range of goods. Due to the wide range of requirements of their customers, the variety of boxes the *FlexBox* has to produce is very extensive.

The boxes are all rectangular and have the following characteristics:

- They are all made of cardboard;
- The cardboard has a specified grade;
- The boxes may have no printing, or 1, or 2 colour printing;
- Some boxes may have reinforced bottoms;
- Some boxes may have reinforced corners;
- All boxes may have sealable tops.

Fig. 1. Simple cardboard boxes.

The types of boxes, produced by the company, are shown in Table1 and the costs of 1m^2 of cardboard are given in Table2.

Table 1. Types of cardboard boxes available.

ype	Grade of cardboard	Colour print			einforced bottom	einforced corners
		0	1	2		
I	1 – 3	YES	NO	NO	NO	NO
II	2 – 4	NO	YES	NO	NO	NO
III	2 – 5	NO	NO	YES	NO	NO
IV	2 – 5	NO	NO	YES	YES	NO
V	3 – 5	NO	NO	YES	YES	YES

Table 2. Basic cost of 1 square metre of cardboard.

Cardboard Grade	1	2	3	4	5
Cost per m^2 [in £]	0.55	0.65	0.82	0.98	1.5

Table 3. Additional costs.

1 colour	12% extra
2 colours	15% extra
Reinforced bottom	13% extra
Reinforced corners	12% extra
Sealable tops	10% extra

There are some additional costs depending on whether the box has printing and if there is any box reinforcement.
These are shown in Table 3 and the percentage increases **are all applied using the basic cost**.
All boxes may have scalable tops.

When a customer asks *FlexBox* to quote a price for an order, they specify the following:

- The size of the box (width, length, and height);
- The grade of the cardboard;
- Whether they want any colour printing (no colour, or 1, or 2 colour printing);
- Whether they want any bottom and/or corner reinforcement;
- Whether the box has a sealable top;
- The quantity of boxes.

From this information, the order system should determine if the type of the requested box can be supplied by *FlexBox*, if it cannot, it should display an appropriate message and reject the order. If the ordered box/boxes correspond to any of the types given in Table 1, and can be supplied by *FlexBox*, the cost of the order must be calculated (using Table 2 and Table 3) and quoted.

The customer should be able to place several orders in one session, in which case the total cost should be prompted.

Customers should not be asked for the type of box they want (since this is only used within the company to calculate the cost). **It is your application that must determine** (using Table 1) **the box type, based on the ordered box attributes.**

Customers should be able to get a quote for as many pipes (of different types) as they like (within the capacity of *FlexBox*) in the same order. In such cases, the total cost of the order should be calculated and displayed.
Your user interface should be a GUI (graphical user interface) using AWT/Swing. If no GUI is developed, you will lose the marks allocated for this part of your coursework.

Your Task

- Write an application, which will allow the customers to enter the details of their order and subsequently prompts the cost of the order. Your application should verify that *FlexBox* can supply the type of the requested boxes (the customer should not be asked to specify the box type).
- Use OO design approach (abstraction, inheritance, aggregation, and polymorphism) and create a class hierarchy that describes the types of boxes *FlexBox* sells. Use an abstract class as well.
- Give UML use case diagram, UML class hierarchy diagram, one class and one instance diagrams.
- Use proper level of abstraction, encapsulation and accessibility for the class attributes and methods. Application with no levels of abstraction will fail the coursework!
- Devise suitable test plan and data, which you can use to test the performance of your ordering system.

Assessment Criteria

- You should give **a demonstration and submit a memory stick** (with **your group number on it**) with your source code and Java NetBeans project files of your software no later than week12 (week starting **3.XII.2018**), during your lab session.

On **7.XII.2018** your group should submit electronically (**by 6pm**) to Moodle a **.pdf** file with your **report. The file name should be your group name** (e.g., ***GrC-2.pdf***, or ***GrA-3.pdf***, or ***GrD-5.pdf***, etc.). **Only one file per group** should be submitted (decide in advance who is going to submit it). Your report **should be no more than 6 pages (excluding the appendices) and should include** the following:

- ✓ **A UML** use case diagram of your order system, UML class hierarchy diagram of your OO application design, and also one UML class diagram (one class of your choice), and one instance diagram;
- ✓ **A brief** description of the application including any assumptions you have made and any limitations in your implementation of the application;
- ✓ **A test** schedule (no more than one page) and screen shots to evidence the testing and evaluation;
- ✓ **The source** code that you have written as an Appendix (the same code that you used in your demonstration);
- ✓ **Some sample** input and output (screenshots) to demonstrate your application is working;
- ✓ **A Group contribution form** with your individual contributions;
- ✓ **This document**.

The assessment criteria and allocated marks are given in Table 4.

Table 4. Assessment criteria and marks distribution.

Topic/Criteria	Comments	Marks available	Marks awarded
Class hierarchy descriptions (UML)	How suitable is the design and the adopted hierarchy of the application? Use of abstract class?	10 (Report)	
UML class and instance diagrams	Are the UML use case, class and instance diagrams relevant to the application?	10 (Report)	
Code and functionality	How complete is the implementation? Does it perform as specified? Does it use an OO design approach? Use of abstract class? Are the class attributes and methods at the appropriate hierarchy level? Is the verification and validation of input data adequate? Is the exception handling properly done? Are the style, indentation and comments appropriate? Is the layout clear?	45 (Demo(20), Report(25))	
Using AWT/Swing	Is the layout clear? How well designed is the interface? How appropriate is the use of components? How appropriate is the use of attributes? Is it working, or just an attempt?	15 (Demo)	
Testing	How thorough is planning and testing? Does it cover most/few possible errors?	10 (Report)	
Supporting documentation and comments.	Is the text clearly written and well presented? Are the assumptions, limitations, problems and features of the application well documented?	10 (Report)	
OVERALL MARK		100	