ADPROC, Advanced Programming Concepts (U21266)

Coursework

Hand out date: 21.X.2014 **Hand in date: 5.XII.2014 (Demonstration: by week 11)**

This is an assessed piece of individual 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 11, in your lab session) **you have to submit a disk** (with your student number on it) with your source code and Java NetBeans project files.
The coursework you submit should be your own 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 which makes an extensive range of boxes for packaging a wide range of goods. Due to the wide range of requirements of their customers, the variety of boxes which they have to produce is very extensive. The boxes are all rectangular and have the following characteristics:

Fig. 1. Simple cardboard boxes.

- They are all made of card;
- The card 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.

The types of boxes, produced by the company, are shown in Table 1.

Table 1. Types of cardboard boxes available.

Type	Grade of card	Colour print			Reinforced bottom	Reinforced 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

The costs of $1m^2$ of card are given in Table 2.

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

Grade of card	1	2	3	4	5
Cost per m^2 [in £]	0.50	0.59	0.70	0.92	1.35

Table 3. Additional costs.

1 colour	12% extra
2 colours Text	15% extra
Reinforced bottom	12% extra
Reinforced corners	8% extra
Sealable tops	6% extra

There are some additional costs depending on whether the box has printing and if there is any reinforcing. These are shown in Table 3 and the percentage increase **is based on the basic cost**.
All boxes may have sealable 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 card;
- 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 box requested 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 and quoted.

Customers should not be asked for the type of the box they order, since this is only used within the company to calculate the cost: **your application must determine the type of the ordered boxes** (using Table 1).

If a customer is placing more than one order (say one order for 5 boxes (of type I) and another order of 10 boxes (of type II)), then he/she should receive a quote with the total cost of the orders.
Your user interface should be a GUI (graphical user interface) using AWT/Swing. If no GUI is used, you will lose the marks allocated for this part of your coursework.

Your Task

- Write an application, which will allow the customer to enter the details of his/her order and will calculate the cost of the order. Your application should verify that *FlexBox* can supply the type of box requested (the customer should not be asked to specify the box type).
- Use OO design approach (abstraction, inheritance and polymorphism) and create a class hierarchy that describes the types of boxes *FlexBox* sells. Use an abstract class if necessary.
- 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.
- 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 disk** (with your student number on it) with your source code and Java NetBeans project files of your software no later than week11 (**4.XII.2014**), during your lab session.
On **5.XII.2014** you should hand in a **report** that includes the following:

- ✓ A UML use case diagram of placing an order, UML class hierarchy diagram of your application design, and also one UML class diagram 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 and screen shots to evidence the testing and evaluation;
- ✓ The code you have written as an Appendix (the same code that you submitted on the disc during your demonstration);
- ✓ Some sample input and output to demonstrate your application is working;
- ✓ This document.

The assessment criteria and marks distribution 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 for 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	