****

ICA40511 Certificate IV in Programming

Cluster: ICTPRG415 – Apply skills in object-oriented design

ICTPRG406 – Apply introductory object-oriented language skills

**Assignment 3: Documented C# Application Project**

**Description:** This is an object-oriented language skills and documentation assignment in which you must demonstrate how you would develop an object-oriented application in C# and document this application according to organisational standards and target language.

**Method:** Portfolio of evidence;

Observation of skills by the lecturer of you coding and documenting in C# is considered to be part of the assessment process.

**Due:**  Week 17 - Thursday 24 November 2016

**Which part of the units are being assessed?**

ICTPRG406: Element 1, 2, 3, 4, 5, 6

**Instructions:** Prepare the following:

* C# Windows Form Application showing:
  + Use arrays of objects
  + Inheritance must be shown to at least 2 levels
  + Apply internal documentation to code and activities
* Test Sheet showing:
  + - Tests performed
    - Results received
  + Full Application documentation

Submit your zipped documents and C# windows form application into the Blackboard assessments area. The work submitted must be your own individual effort. (Team work is not permitted.)

**Results and Appeals:**

Please refer to North Metropolitan TAFE website for information about the assessment process. <https://www.central.wa.edu.au/current_students/services/Pages/Policies-and-responsibilities.aspx>

**Assessment Marking Guide & Feedback:**

Please refer to companion document and Blackboard

**Requirements Scenario**

Please read the following:

The “Bingo Office Supplies” store is a “cash only” business that sells up to 20 products, the details of which it keeps in the file “Products.txt” (Get a copy from Blackboard).

This file has one line records consisting of

* + the product code (5 characters)
  + product name of up to 20 characters
  + the number in stock
  + the product cost price
  + the name of the file holding an image of the product. (at most 20 characters)

These fields are separated by commas.

When the company makes a sale of a product the following information needs to be recorded on a Sales Invoice. The company makes a maximum of 20 sales per day.

* + The Sales Invoice number (Start at any number and increment by 1)
  + The product name
  + The product selling price (mark-up of 60%)
  + The quantity purchased.
  + The value of the sale

**Part 1**

Create a **Product class** with the fields indicated above as the product attributes.

It must have a constructor method that sets all the attributes to parameter passed values and a null constructor.

It is to have the following additional methods.

* + Get and Set methods for all attributes except the picture file name field which is to have only a Get method

* + A “StockAvailable” method that returns false if the number in stock is below a parameter passed value and true if it is not.

* + A “MakeSale” method that will deduct a parameter passed number from the “Number in stock” data item.

Create a **Sales Invoice class** with the fields indicated above as the Sales Invoice attributes. It must have a constructor that sets each attribute to parameter passed values and a null constructor.

It is to have the following additional methods

* + Get and Set methods for all attributes
  + A method that calculates the total value of that sale (Selling Price \* Quantity Purchased)

**Part 2**

The “Bingo Office Supplies” store has decided to give a 5% discount where the value of the sale on the Sales Invoice is greater than $20 and also to capture details of those customers.

The company will capture the following Big Spender details

* + The customer name (at most 15 characters)
  + The customer phone number (at most 10 characters)

Create a **child class of Sales Invoice class called BigSpender** with the fields indicated above as the BigSpender attributes. It will inherit from Sales Invoice but also have a few additional attributes and methods. It must have a constructor that sets each attribute to parameter passed values and a null constructor.

It is to have the following additional methods

* + Get and Set methods for all attributes
  + An overloaded method that calculates the total value of the sale

(Selling Price \* Quantity Purchased \* Discount Percent)

**Part 3**

The “Bingo Office Supplies” store has also decided to start a SpotPrize incentive. Only BigSpender invoices are eligible. If a BigSpender invoice has purchased the product of the day e.g. AX012 - calculator (decided by the sales manager), then “BINGO!” They get 1of those products for half price.

The company will store the following spotprize details

* + The product of the day (5 character product code)

Create a **child class of BigSpender called SpotPrize** with the fields indicated above as the SpotPrize attributes. It will inherit from BigSpender but also have a few additional attributes and methods. It must have a constructor that sets each attribute to parameter passed values and a null constructor.

It is to have the following additional methods

* + Get and Set methods for all attributes
  + An overloaded method that calculates the total value of the sale

(Selling Price \* Quantity Purchased \* Discount Percent – Selling Price of Product of the day \* 0.5)

**Part 4**

Create a C# Windows Form Application for the company to match the specifications given below (you design an appropriate form). The application is to be started at the beginning of each day the store is open and is to be used by a sales person to assist them in making sales and in recording all separate sale details. Application actions are as below:

* + On loading the form, the application is to copy the contents of the current products file (Products.txt) into an array of product objects. The product names only are also to be copied into a list box on the form.

* + When a user selects (clicks on) a product name within the form’s list box, its retail price (60% mark-up on cost price) and picture are to be displayed on the form below the list box.

* + When a “Sell Product” button is clicked, an invoice number is created and displayed. The user will be prompted to enter the number of the currently selected product they are to sell (via an input box). When there are not sufficient items in stock to match the required number (check using the “StockAvailable” method), a message indicating this is to pop up. This message should also show the current number of that item in stock. If there are sufficient numbers of that item in stock, then the value of the sale is to be displayed in a text box or label and “Confirm Sale” and “Cancel Sale” buttons are to appear.

* + When the “Confirm Sale” button is clicked the “MakeSale” method must be called to reduce, for that product, the number in stock by the sale quantity. Depending on the circumstances of the sale, you will instantiate either a **new Sales Invoice object**, **BigSpender object or SpotPrize object** using the parameter constructor.This will be stored as an array of Invoice objects. The “Confirm sale” and Cancel Sale” buttons are to then to disappear from the form and the sale value text box or label cleared.

* + When a “Cancel Sale” button is clicked no action is to occur apart from the “Confirm sale” and Cancel Sale” buttons disappearing from view and the clearing of the sale value text box or label.
  + When a “Close of Business” button is clicked a final **Sales Invoice object** is instantiated using the null constructor. Use the Get and Set methods to set the
    - The Sales Invoice number to 999
    - The product name = “Total Sales for: dd/mm/yyyy” where dd/mm/yyyy is the current date.
    - The product selling price = 0
    - The quantity purchased = Total number of products sold for the day
    - The value of the sale = Total value of sales for the day

* + The following is also to occur at close of business:
    - 1. All product details are to be written back into a file with the same name and location as the starting product file and with the same format (with fields separated by commas).
      2. A new sales text file is to be created that contains
         1. detail lines, one for each sale which consist of the invoice number, product name, retail price, quantity sold and sale value, customer name, customer phone number and product of the day separated by commas.
         2. On the final line of this file the details of the COB object are to be written (separated by commas).

Note : To place a picture in a picture box given a string that holds its file name use the statement

PictureBoxName.Image = Image.FromFile(FileNameString)

For example PictureBox1.Image = Image.FromFile("Pokemon.jpg")

**Part 5**

To document this project sufficiently, the following processes, guidelines and standards must be followed:

1. Requirements Documentation - describe the process for developing your small-size application. Include the following:

* Samples of Form(GUI) layouts
* Variables and Data Types required
* Processing required for each control / procedure / method
* Processes for calculations

1. C# naming standards must be followed:

|  |  |  |
| --- | --- | --- |
| **Controls, Items and Data Types:** | **Prefixed By:** | **Example** |
| Text Boxes | txt | txtName, txt\_Title |
| Labels | lbl | lblName, lbl\_Title |
| Buttons | btn | btnDisplay, btn\_Calculate |
| List Boxes | lst | lstProducts |
| Radio Buttons | rdo | rdoGender |
| Integers | int | intAge, int\_Age |
| Double | dbl | dbl\_Salary |
| Decimal | dec | dec\_Average |
| Float | flt | flt\_Value |
| Boolean | bln | bln\_Found |
| String | str | str\_Address |

1. Comments must be inserted at the top of your application stating:

* Name of Programmer
* Date
* Description of what the Application does

1. Comments must be included at the start of every procedure / method / function stating the main purpose.
2. Include comments anywhere else in your code where you feel an explanation adds value to whoever may need to maintain or change your code in the future.
3. Create a table of test data similar to the one below:

Include sufficient test data to confirm that the code meets the design specifications.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Variable** | **Input Value** | **Process** | **Expected Output** | **Actual Output** |
| dbl\_SellPrice | dbl\_CostPrice = $0.80  intMarkUp = 20 | dbl\_CostPrice \* (intMarkUp + 100) / 100 | $0.96 | $96.00 |