Higher Diploma in Science in Computing

January 2016 EXAMINATIONS

Module Code:	**B8IT117**
Module Description:	**Object-Oriented Programming**
Examiner:	**Dr. Shazia A Afzal**
Internal Moderator:	**Damien Kettle**
External Examiner:	**Dr Paul Stynes**

Date: *Monday,25th January2016*
Time: *10:00 – 12:00*

INSTRUCTIONS TO CANDIDATES:

Time allowed is 2 hours.

QUESTION 1 IS COMPULSORY (30 marks);
Answer any 2 other questions (35 marks each).

<u>Question 1 – Compulsory – 30 Marks</u>

a) Explain the following by providing appropriate examples for each of them.

   i. Constructor
   ii. Interface
   iii. Overriding
   iv. Operator Overloading

(4 *5 = 20 marks)

b) Explain the concept of encapsulation in object-oriented programming. List and briefly explain the access modifiers provided by C#.

(10 marks)
(Total: 30 marks)

<u>Question 2 - 35 Marks</u>

Create an inheritance hierarchy to represent various types of purchases made by manufacturing organizations, which might be used by a manufacturing company for keeping purchase information details.

1. Create class Purchase as the base class of the hierarchy, then include classes CashPurchase and CreditPurchase that derive from Purchase. Base class Purchase should include data members representing
   - The name of supplier
   - Factory code of the factory for which the materials are procured
   - Item code
   - Item description of the material purchased
   - The quantity purchased in units
   - Cost per unit
   - The constructor of the Purchase class should initialize these data members
   - The Purchase class should provide a public method CalculateCost that returns a double value indicating the cost associated with each purchase. Purchase's CalculateCost() method should determine the cost by multiplying the quantity by the cost per unit
   - A ToString() method to return the details of Purchase objects

Note: The properties should validate the quantity purchased and the cost per unit, to ensure that the values are greater than or equal to 0.0; if not, throw an exception.

(11 marks)

2. Derived class CashPurchase should inherit the functionality of base class Purchase, and also include
   - A data member that represents a discount that is generally provided by the suppliers for cash purchase
   - A Discount property should check that the discount value is never less than zero, otherwise, it should throw an exception
   - Constructor of the CashPurchase class should receive a value to initialize the data members
   - The CashPurchase class should redefine member function CalculateCost so that it computes the total cost by subtracting the discount from the quantity-based cost calculated by base class Purchase's CalculateCost function (discount is calculated by multiplying it to the net cost returned by the CalculateCost() method of base class and then is subtracted from the net cost)
   - ToString() method

(8 marks)

3. The derived class CreditPurchase should inherit directly from class Purchase and should contain
   - An additional data member representing an additional charge per unit charged for credit purchase
   - A CreditCharge property should check that the charges value is never less than zero, otherwise, it should throw an exception
   - The CreditPurchase class should redefine member function CalculateCost so that it adds the additional charge per unit to the standard cost per unit before calculating the material cost. (Cost returned by the CalculateCost() method of base class is added to the QuantityPurchased multiplied by the CreditCharge)
   - ToString() method

(8 marks)

4. Write a test class that creates objects of each type of Purchase and show the details.

Sample Output is given below:

```
Supplier Name: Emma Brown
Factory Code: F001
Item Code: 1111
Quantity: 8.5
Cost Per Unit: 5.00 euros.
Total Cost: 42.50 euros.
______________________________________________________
```

```
Supplier Name: Lisa Kelly
Factory Code: F001
Item Code:  9802
Quantity: 45
Cost Per Unit: 100.00 euros.
Total Cost: 3600.00 euros.
Discount: 0.20
___________________________________________________
Supplier Name: Pat Smith
Factory Code: F003
Item Code: 2301
Quantity: 20
Cost Per Unit: 43.00 euros.
Total Cost: 860.00 euros.
Credit Charged: 0.00
___________________________________________________
```

(8 marks)
(Total: 35 marks)

Question 3 - 35 Marks

You are developing an application for geometric shapes. To develop this application, write C# code for the following:

1. An interface named IColourable with
   - A method
     - string HowToColor();

(5 marks)

2. An abstract class Shape that must implement the IColourable and IComparable interfaces. The Shape class is used to model all geometric shapes such as squares, circles, triangles, etc. These shapes can be drawn in certain colours. The Shape class has the following properties and methods:
   - Colour (Property)
   - DateCreated (Property)
   - A constructor to initialise the properties
   - GetArea() (abstract method)
   - GetPerimeter() (abstract method)
   - ToString() (method)
   - Methods that are defined in the interfaces

(12 marks)

3. A class Square inherited from the Shape class with the following members:
   - Name (read only property) that returns “Square”
   - Side (property)
   - A parameterised constructor
   - Implementation of HowToColour that returns “Colour all four sides”
   - Implementations of abstract methods

(12 marks)

Note: Formula for Area of a Square: Side * Side and Formula for Perimeter of a Square is: 4* Side.

4. A test class with a list of three square objects. Sort the list by area of the squares. Display the details of each square object.

   A sample output after sorting is shown below:

```
*********After Sorting*************
Name: Square
Colour: Green
Date Created: 12/11/2013
Side: 6
Area: 36
Perimeter: 24
________________________
Name: Square
Colour: Red
Date Created: 10/12/2014
Side: 7
Area: 49
Perimeter: 28
________________________
Name: Square
Colour: Black
Date Created: 09/12/2012
Side: 8
Area: 64
Perimeter: 32
____________________
```

(6 marks)
(Total: 35 marks)

<u>Question 4 - 35 Marks</u>

You are required to develop a console application "CatalogueOfCourses" for a local college in Dublin. To develop this application, write the following C# classes:

1. A Course class with the following properties:
    - CourseCode
    - CourseTitle
    - StartDate
    - MaxNumber (represents maximum no of students to be registered on the course)
    - A parameterised constructor to initialise all the properties
    - And a method ToString() to return all details of a course

    (8 marks)
2. A Courses class that implements ICollection<Course> interface and holds a property CourseList<Course> to store a number of courses.
    - Courses class should use a method to Sort the Courses by Code and then by Title (Write a class with a compare method to provide this function)
    - A method to find a course by Title. So the method takes the title as an argument from the user and finds the course

    (15 marks)
3. A test class with a property Catalogue and a menu for the following functions to see the courses in the Catalogue:
    - Add a Course
    - Find a Course (by entering CourseTitle)
        - Appropriate message should be displayed on the screen
    - Show all courses (sort the before displaying)
    - Quit (to quit the application)

    (12 marks)

(Total: 35 marks)

<u>END OF EXAMINATION</u>