# Description: _LITfinalLOGO

# SUMMER EXAMINATIONS 2013

**Friday, 10th May 2013, 14.30 p.m. – 16.30 p.m.**

**KSDEM\_8\_Y2**

**Course:** Bachelor of Science (Hons) in Software Development

**Year:** Two

**Subject:** Object Oriented & GUI Programming

**Time Allowed:** 2 Hours

**Instructions: 1.** You **MUST** answer **Q1**

Answer **ANY OTHER TWO** questions.

**2.** Marks for **Q1** are **50 marks**.

All other questions are **25 marks**.

**3.** Start each question on a new page.

**4.** Write the question number at the top of each page.

**5.** Circle the numbers of the questions you answer at the front of your answer book.

**Additional Attachments Exam Materials to accompany this paper:**

### None

**Internal Examiners: External Examiners:**

Tom Costello Mr. Brian Gillespie

**Q.1**  Answer any FIVE of the following **(10 marks per part)**

1. The WinMain function of a Windows API Program will contain the following code

**while ( GetMessage(&msg, NULL, 0, 0) ) {**

**TranslateMessage(&msg);**

**DispatchMessage(&msg);**

**}**

Describe the role of this code during application execution.

1. In unmanaged C++ programming, there should be a matching call to the operator **delete** for each call to the operator **new**. In C# .NET programming this condition does not apply. Explain why this is so.
2. Describe the role of the Windows Form Designer in Visual Studio.
3. In a Windows Forms Application describe the roles of the **IsMdiContainer** and **MdiParent** properties of the **class Form**.
4. Describe the role of the following code fragment from a C# .NET program

**Lecturer lecturer = new Lecturer("Tom", 1234);**

**FileStream lecturerFile = new FileStream("LecturerFile.bin",**

**FileMode.Append, FileAccess.Write);**

**BinaryFormatter bformatter = new BinaryFormatter();**

**bformatter.Serialize(lecturerFile, lecturer); ;**

**lecturerFile.Close();**

1. Describe in detail the role of the following two lines of code from the definition of the class Timer

**public delegate void ElapsedEventHandler(object source, ElapsedEventArgs e);**

**public event ElapsedEventHandler Elapsed;**

1. With respect to WPF controls, write a brief note on Routed Events
2. The following extract from a XAML file contains examples of each of Object Element Syntax, Attribute Syntax, Property Element Syntax and Content Syntax.

Write out two examples of each type, explaining its role.

**<Grid>**

**<Ellipse Margin="50,50,0,0" Name="ellipse5" Stroke="Black"**

**Height="150" Width="150">**

**<Ellipse.Effect>**

**<BlurEffect Radius="10" />**

**</Ellipse.Effect>**

**<Ellipse.Fill>**

**<RadialGradientBrush>**

**<GradientStop Color="#FF000000" Offset="1"/>**

**<GradientStop Color="#FFFFFFFF" Offset="0.306"/>**

**</RadialGradientBrush>**

**</Ellipse.Fill>**

**</Ellipse>**

**<Button Height="23" Name="goButton" Width="75"> Go </Button>**

**<Button Height="23" Name="stopButton" Width="75"> Stop </Button>**

**</Grid>**

**(Total 50 Marks)**

**Q. 2**

Write suitable definitions in C# for the classes **Account** and **CurrentAccount** in the class hierarchy for various bank account types shown below, given the following conditions:

* The account class should be an abstract base class **(1 mark)**
* The current account class should be a concrete class **(1 mark)**
* The account class should have fields for an id and a balance and the current account class should have a field for an overdraft limit **(3 marks)**
* Default constructors should be included in each of the classes **(2 marks)**
* An additional constructor should be included in each of the classes that may be passed parameters as follows:
  + an id and a balance for the account class
  + an id, a balance and a credit limit for the current account class

**(4 marks)**

* A method should be included to support lodgements to all types of accounts, assuming that an account will just have its’ balance topped up by the amount of the lodgement **(3 marks)**
* Methods should be included to support withdrawals from accounts. In the case of current accounts, withdrawals can be made up to the credit limit value **(5 marks)**
* Properties should be included to allow access to the class fields. However, the balance in any type of account should not be capable of being changed except through the lodgement and withdrawal methods.

**(6 marks)**

##### class Account

##### class CurrentAccount

##### class DepositAccount

##### class SavingsAccount

**(Total 25 Marks)**

**Q.3**

1. Distinguish between the terms ‘reference equality’ and ‘value equality’ for objects of a class. Illustrate your answer with some sample code

**(8 marks)**

1. Some of the recommendations for implementing value equality for a class are
   1. Override the **virtual Object.Equals(obj Object)** method. This should just call the type-specific **Equals** method

**(4 marks)**

* 1. Implement the **IEquatable<T>** interface by providing a type-specific **Equals** method

**(10 marks)**

* 1. Override the **Object.GetHashCode( )** so that two objects with the same fields produce the same hash code

**(3 marks)**

For the **class GridPosition** partially defined below, redefine the class implementing the above recommendations.

**public class GridPosition**

**{**

**private int row;**

**private int col;**

**public GridPosition() {this.row = 0;this.col = 0;}**

**public GridPosition(int row, int col)**

**{this.row = row;this.col = col;}**

**public int Row**

**{**

**get{ return row; }**

**}**

**public int Col**

**{**

**get{ return col; }**

**}**

**}**

**(Total 25 Marks)**

**Q. 4** The **class GridPosition** shown in **Q.3** has the method **Move** added to the

class as defined below. The method will throw an exception if a move were to

cause either the row or col value to fall out of the range 0 to 99.

**public void Move(int numRows, int numCols)**

**{**

**this.row += numRows;**

**if ( this.row < 0 || this.row > 99 )**

**throw( new GPException("row", this.row-numRows,numRows));**

**this.col += numCols;**

**if ( this.col < 0 || this.col > 99 )**

**throw( new GPException("col", this.col-numCols,numCols));**

**}**

1. Write a suitable definition for the **class GPException**.
   1. The class should have three fields as indicated by the call to the constructor **(4 marks)**
   2. A suitable constructor

**(3 marks)**

* 1. A method **Message** that will display suitable details of why the exception has occurred.

**(5 marks)**

1. The sample program fragment below instantiates an object of the **class GridPosition** and applies the **Move** method to the object. Re-write the code including suitable Structured Exception Handling code.

Note: A **FormatException** object may be thrown by the call to **int.Parse( )**

**static void Main(string[] args)**

**{**

**GridPosition g = new GridPosition(49,49);**

**Console.Write("Enter number of rows to move :");**

**int numRows = int.Parse(Console.ReadLine());**

**g.Move(numRows, 0);**

**Console.Write("Enter number of cols to move :");**

**int numCols = int.Parse(Console.ReadLine());**

**g.Move(0, numCols);**

**Console.WriteLine("New position is {0}, {1}", g.Row, g.Col);**

**}**

**(13 marks) (Total 25 Marks)**