# Description: _LITfinalLOGO

# SUMMER EXAMINATIONS 2012

**Friday, 11th May 2012, 14.30 p.m. – 16.30 p.m.**

**KSDEM\_8\_Y2**

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

**Year:** One

**Subject:** Object Oriented & GUI Programming

**Time Allowed:** 2 Hours

**Instructions: 1.** You **MUST** answer **Question ONE (1)**

Answer **ANY OTHER TWO** **(2)** questions.

**2.** Marks for **Question ONE (1)** 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:**

### Appendix A for Q.4

**Internal Examiners: External Examiners:**

Tom Costello Mr. Paul Powell

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

1. A Windows API Program will contain a window function with a prototype

**LRESULT CALLBACK WndProc(HWND hWnd,**

**UINT message, WPARAM wParam, LPARAM lParam);**

Write a skeleton definition of the function and use it to describe the role of the function.

1. “The .NET framework uses a Just In Time (JIT) compiler”. Discuss this statement.
2. With reference to the deployment of Windows Applications, describe the process called ‘ClickOnce deployment’.
3. In a Windows Forms Application, Visual Studio will add a class derived from the **class System.Windows.Forms.Form** to the project. To that class it will add a method

**private void InitializeComponent( )**

Describe the role of this function.

1. Describe the role of the highlighted code in the following code fragment from a Windows Forms program in C#

public partial class formContainer : Form

{

public formContainer()

{

**Application.ApplicationExit**

**+= new EventHandler(this.OnApplicationExit);**

InitializeComponent();

}

**private void OnApplicationExit(object sender, EventArgs e)**

**{**

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

**FileStream studentFile = new FileStream("Students.bin",**

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

**bformatter.Serialize(studentFile, studentList);**

**studentFile.Close();**

**}**

}

1. “WPF applications use a markup plus code-behind model” . Describe what is meant by this statement and discuss the advantages of using such a model.
2. With respect to WPF controls, write a brief note on each of the following
   1. Dependency Properties
   2. Attached Properties
3. Describe in detail the role of the following XAML code

**<Window.Triggers>**

**<EventTrigger RoutedEvent="ButtonBase.Click"**

**SourceName="stopButton">**

**<PauseStoryboard BeginStoryboardName**

**="Spin\_BeginStoryboard"/>**

**</EventTrigger>**

**</Window.Triggers>**

**<Grid>**

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

**<Button Height="23” Name="toggleButton" Content="Toggle"**

**Click="toggleButton\_Click" />**

**</Grid>**

**(Total 50 Marks)**

**Q. 2**

1. Write suitable definitions in C++ for each of the classes in the class hierarchy shown below, given the following conditions:

* An account has an id and a balance. The class should be an abstract base class.
* A deposit account should have an interest rate.
* A current account should have an overdraft limit.
* Default constructors should be included in each of the classes
* 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 **class CAccount**
  + an id, a balance and an interest rate for the **class CDepositAccount**
  + an id, a balance and a credit limit for the **class CCurrentAccount**
* Methods should be included to support lodgements to and withdrawals from accounts.
* Methods should be included to allow access to the class data members. However, the balance in any type of account should not be capable of being changed except through the lodgement and withdrawal methods.
* The class hierarchy should support runtime polymorphism.

**(3 x 6 marks)**

Note: You need not write an implementation file for the classes

1. Write definitions for the non-default constructors in the classes
   1. **class CAccount**

**(3 marks)**

* 1. **class CDepositAccount**

**(4 marks)**

##### class CAccount

**(Total 25 Marks)**

##### class CCurrentAccount

##### class CDepositAccount

**Q.3**

1. Describe the role of interfaces in C# programming

**(5 marks)**

1. By commenting on the creation of copies of objects of the two classes defined below, distinguish between the terms shallow copying and deep copying.

**public class Student**

**{**

**private string name;**

**private int mark;**

**public Student(string name, int mark)**

**{**

**this.name = name;**

**this.mark = mark;**

**}**

**}**

**public class Group**

**{**

**private ArrayList list;**

**public Group()**

**{**

**list = new ArrayList();**

**list.Add( new Student(“Fred”, 44);**

**list.Add( new Student(“Ann”, 55);**

**list.Add( new Student(“Jim”, 66);**

**}**

**}**

**(8 marks)**

1. The copying of objects in a standard way may be achieved by a class implementing the **ICloneable** interface. Rewrite both classes so that they implement this interface.

**(12 marks)**

**(Total 25 Marks)**

# Q. 4

# Describe the roles of the keywords try, throw, catch and finally that are used in the C# language to support the implementation of structured exception handling.

**(12 marks)**

1. **Appendix A** contains the definition for a **class** **Sample** and the method **Main** for the **class Program** which instantiates objects of the **class Sample.**

The appendix also contains the definition for a **class SampleException** and an object of this class should be thrown if a sample code which is not six-digits long is used.

Re-write the method **Main** with the inclusion of appropriate structured exception handling code.

**(13 marks)**

**(Total 25 Marks)**

**Appendix A**

**public class Sample**

**{**

**private int code;**

**private double weight;**

**public Sample(int code, double weight)**

**{**

**this.code = code;**

**this.weight = weight;**

**}**

**public int Code**

**{**

**get{ return code;}**

**}**

**public double Weight**

**{**

**get{ return weight; }**

**set{ weight = value; }**

**}**

**public void Display()**

**{**

**Console.WriteLine("{0} {1}", code, weight);**

**}**

**}**

**class Program**

**{**

**static ArrayList list;**

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

**{**

**list = new ArrayList();**

**list.Add(new Sample(001234, 3.45));**

**list.Add(new Sample(001235, 3.45));**

**list.Add(new Sample(001236, 3.45));**

**list.Add(new Sample(001237, 3.45));**

**list.Add(new Sample(001238, 3.45));**

**Appendix A contd.**

**Console.Write("Enter sample code : ");**

**string codeStr = Console.ReadLine();**

**int code = int.Parse(codeStr);**

**bool found = false;**

**foreach ( Sample s in list )**

**{**

**if (s.Code == code)**

**{**

**s.Display();**

**found = true;**

**break;**

**}**

**}**

**if (!found)**

**{**

**Console.WriteLine("Could not find sample with this code");**

**}**

**}**

**}**

**public class SampleException:System.Exception**

**{**

**string codeStr;**

**public SampleException(string codeStr)**

**{**

**this.codeStr = codeStr;**

**}**

**public void Message()**

**{**

**Console.WriteLine("Invalid code {0}, is only {1} digits long",**

**codeStr, codeStr.Length);**

**Console.WriteLine("All codes should be 6 digits long");**

**}**

**}**