MT264 Revision Paper Sample Solutions

1A

(a) Set aButton.Width To 40

(b) It will assign 10 to the Value property of the component aNumUpDn; this value will be displayed on the control.

(c)
```
aString As String
    Set aString To "Name"
    Set aLabel.Text To aString
```

(d) The Items property; use it when you want to have the user select one input value chosen from a predefined list. Example: in the Unit 3 “Character Finder” application, the user must select a target letter from a list (A-Z) presented by the GUI. Allowing the user to type into a text box would not restrict the user to a predefined list of allowable input values.

(Note: ComboBox originally meant a combination of a textbox and a listbox – you can still use it like that if you use the Text property rather than SelectedItem in code.)

1B

(a) Set aMonthCal.MaxSelectionCount To 1

(b) It will make the control aListBox become visible to the user of the GUI.

(c)
```
aColor As Color
    Set aColor To Color.Yellow
    Set aLabel.BackColor To aColor
```

(d) The Image property

2A

(a) isEngraved

(b) Yes, Boolean (return values: True or False)

(c) Integer

(d) No [the assumption is that the method implementations satisfy the postconditions - a rogue implementation *could* change the values but this is not what the question is about]. Both use the property value of Inscription but do not update anything.

(e)(i)
```
aTrophy As Trophy
      Set aTrophy To New Trophy
      Set aTrophy.Style To "Cup"
      Set aTrophy.Finish To "Silver"
      Set aTrophy.HeightInInches To 10
      Set aTrophy.Inscription To "The Winner, 2009"
```

(ii) 10 * number of non-space characters = 10 * 14 = 140 (as answer is in pence)

2B

(a) addDonation

(b) No (as there is no As *SomeType* at the end of the heading)

(c) Yes, one, the ‘donation’ parameter, of type Integer

(d) Yes – the addDonation method may change the value of the AmountCollected property. (Indeed it will, unless donation is 0.)

(e)(i)
```
aTin As CollectingTin
      Set aTin To New CollectingTin
      Set aTin.Charity To "Help the Aged"
      Set aTin.Collector To "Sue"
```

Adding: Set aTin.AmountCollected To 0 is not needed as the constructor will do this.

(ii) 100

MT264 Revision Paper Sample Solutions

3A

(a) The design code sets the value of the variable *total* to be the total of all the ages relating to key entries with name values less than "N".

(b) Note that the question asked for changes. The If statement should become

```
If (key >= "A" And key < "N") Then
  Set total To total + 1
End If
```

The selection of names has been changed to be in the range A-M inclusive (in fact, this could have been left as it was, but good exam technique is to spell things like this out). In the Set statement, *total* is now a count of the names selected, rather than the total of the values of the ages.

3B

(a) True

(b) 3

(c) 8

(d) The resulting value of *total* is 44, this being the total of all the Integer values in the list after the *remove* and *removeAt* statements have been executed, viz

1+2+3+5+6+8+9+10 = 44

Note that: aList.remove(4) removes 4

```
0   1   2   3   4   5   6   7   8   9  (index)
1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10
```

aList.removeAt(5) removes 7 which is now at index 5 (as 4 is already removed)

```
0   1   2   3   4   5   6   7   8  (index)
1 + 2 + 3 + 5 + 6 + 7 + 8 + 9 + 10
0   1   2   3   4   5   6   7  (index)
1 + 2 + 3 + 5 + 6 + 8 + 9 + 10  (total = 44 and count = 8)
```

4A

(a) Both constructors are declared Protected, which means that they can only be used inside the class implementation for creating an object of Photo. However, none of the methods or properties of Photo require an instance of Photo to be created, and client code cannot directly use a protected constructor. So there is no way to instantiate the class Photo, and it can therefore be considered to be abstract. [Note that constructors are not inherited, although a protected constructor can be used in a subclass constructor for carrying out the superclass initialisations on a newly created subclass instance with the code MyBase.New. Because a protected constructor can generally be used inside the class implementation to instantiate the class, VB has the keyword MustInherit to mark an abstract class.]

(b) Example of overloading - the instance can be instantiated with or without a parameter for the subject (Unit 6 page 10). (Overloading means having two or more items with the same name in the same scope but with different parameters. Overriding is used to create a new version of a method - Unit 6 page 11.)

(c)

```
Constructor New
    MyBase.New
    Set Size To 0 ‘Note: other properties set by call to superclass constructor
  End Constructor
```

(d)

```
aDigitalPhoto As DigitalPhoto
Set aDigitalPhoto To New DigitalPhoto
```

```
      Set aDigitalPhoto.Size To 125600
      Set aDigitalPhoto.Subject To SubjectType.Portrait
      Set aDigitalPhoto.Album To "Family"
      Set aDigitalPhoto.Colour To True
```

4B

(a) Yes, but because it is declared protected, it can be invoked only in the class implementation or a subclass implementation.

(b) Yes – isTimeForVaccination is a Public method within the class definition of the subclass Cat, and can therefore be invoked from another class.

(c)
```
Constructor New
          MyBase.New
          Set VaccinationRecord To New Dictionary (Of String, String)
      End Constructor
```

(d)
```
moggy As Cat
     Set moggy To New Cat
     Set moggy.DateOfBirth To "01/01/2000"
     Set moggy.Colour To "Black"
     Set moggy.Name To "Moggy"
```

(e) A new version of isYoungster can be implemented in a subclass of Pet.

5A

(a)
```
 aFont As Font
      If aFontDialog.showDialog() = DialogResult.OK Then
         Set aFont To aFontDialog.Font
         'newFont = New Font(aFont, FontStyle.Bold)
         'or from the Handbook
         newFont = New Font(aFont.FontFamily, aFont.Size, FontStyle.Bold)
      End if
```

(b)
```
aReader As StreamReader
     aLine As String
     Try
        Set aReader To StreamReader(fileName)
        Set aLine To aReader.readLine()
     Catch ex As Exception
        MessageBox.Show("File could not be read")
     Finally
        If aReader IsNot Nothing Then
           aReader.close()
        End If
     End Try
```

5B

(a)(i) The loop uses a dialogue box to ask the user to enter their date of birth. If the user confirms this entry, the date is compared with today's date; if the dates match then the aString variable is set to a message to the effect that today is the user's birthday, otherwise the string is set to a message "not yet :-(" (the aString variable is not subsequently used in this code fragment). If the user cancels the dialog box then the loop is exited.

(ii) The loop control is the Boolean variable *finished*, which is False at the start of the loop, and becomes True when the user cancels the dialogue box, thus ending the loop.
(iii) The overall effect is to display repeatedly the dialogue box that asks a user to enter their birthday. The dialogue box will stop reappearing when the user cancels or closes with the X button the dialogue box.

(b)(i) The Finally part will be executed regardless of whether there was an exception or not. So placing the code in this part will make sure that resources are freed in all cases.
(ii) An exception would occurred if aReader could not be created. In that case it is not possible to close aReader, so the code aReader.close() should not be executed in this case.
(iii) The count counts the number of lines in names.txt that are equal to wanted.

6A

```
(a) aPlayer As SoundPlayer
    Set aPlayer To New SoundPlayer("music.wav")
    Try
       aPlayer.playLooping()
    Catch ex As Exception
       MessageBox.Show("Music file could not be played")
    End Try
(b) Version 1:
    aGraphics As Graphics
    aBrush As Brush
    Set aGraphics To canvasPanel.CreateGraphics()
    Set aBrush To New SolidBrush(Color.Red)
    aGraphics.fillRectangle(aBrush, 0, canvasPanel.Height-50, _
            canvasPanel.Width, 50)
    Set aBrush To New SolidBrush(Color.Black)
    aGraphics.drawString("Stop", newFont, aBrush, canvasPanel.Width/2, _
            canvasPanel.Height - 50/2)
```

Version 2, making use of stock brushes and the event handler's parameter e:

```
   e.Graphics.fillRectangle(Brushes.Red, 0, canvasPanel.Height-50, _
            canvasPanel.Width, 50)
   e.Graphics.drawString("Stop", newFont, Brushes.Black, canvasPanel.Width/2, _
            canvasPanel.Height - 50/2)
```

6B

```
(a) Try
       My.Computer.Audio.play("music.wav", AudioPlayMode.BackgroundLoop)
    Catch ex As Exception
       MessageBox.Show("Music file could not be played")
    End Try
```

(b)(i) Line 1 draws a black line from coordinate 0,0 to coordinate 200,200 (i.e. a line from the top left-hand corner of the canvas at 45 degrees towards the bottom-right).
Line 2 inserts an image (‘anImage’) with its top left-hand corner at coordinate 200,200, that is, at the lower end of the line drawn above.
Line 3 inserts a line of text “My Picture”, in the font defined by aFont, in blue, at coordinate 100,100, that is, at the centre of the line drawn above (and to the left and above the image). [Giving a simple sketch might be the easiest way to answer this question.]
(ii) Set fGraphics To canvasPanel.createGraphics() (or: ... To e.Graphics)

Ref: Handbook page 46

7A

(a)
```
Method hoverUp()
     setMovement(0, -10)
 End Method
```
(b)
```
dragon As Dragonfly
Set dragon To New Dragonfly
dragon.hoverUp()
dragon.move()
dragon.move()
```
(c) If dragon.Position.Y >= 10 Then

7B

(a)
```
Method zoomRight()
   setMovement(15, 0)
End Method
```
(b)
```
If Not aPanel.ClientRectangle.contains(dragon.BoundingBox) Then
   MessageBox.Show("Game over")
End If
```
(c) If Not aPanel.ClientRectangle.intersectsWith(dragon.BoundingBox) Then

8A

(a)(i)
```
SELECT Name
  FROM Cat
  WHERE Colour = "Black" AND FavoriteFood = "Tuna";
```
(ii) The result table gives the family names of all those families at whose address a cat was fed on 2/1/09.

(b)(i) Visibility – it is not immediately apparent what the purpose of the form is, and how the user should make use of the controls in order to use the application.
(ii) Structure – the buttons are aligned consistently and neatly, each one opposite its related text box, with the text boxes being positioned consistently and neatly.
(iii) Memorability – related to the issue of (lack of) visibility (above), the form does not provide an interface that helps users to remember how to carry out tasks (especially for systems and operations that are used infrequently). The GUI could be improved by the addition of (at least) appropriate labels or tool tips that inform or remind the user of how to use the interface.
There are many other valid issues that could be addressed.

8B

(a)(i)
```
SELECT Shop.ShopName
       FROM Shop INNER JOIN Delivery
       ON Shop.ShopAddress = Delivery.ShopAddress
       WHERE Delivery.CustomerAddress = '1 The Crescent';
```

(ii) The result table gives the customer names of all those customers who have had a delivery on 1/4/09.

(b)(i) Accessibility changes, e.g. adding keyboard shortcuts and mnemonics to complement buttons and other controls that require a mouse.
Changes to tab order, focus, tool tips, fonts or colours (e.g. providing a new or more modern look to the interface).
(ii) Later improvements to the GUI might not be possible without also changing the model, for example in the case of modifications that involve the manner in which a user might select an option from a list or similar.
(iii) Alternative user interfaces can be created without changes being required to the underlying model.

Q9
(a)
```
Method addPet(addName As String, spec As String, descr As String, col As String)
   aPet as Pet
   Set aPet To New Pet
   Set aPet.Name To addName
   Set aPet.Species To spec
   Set aPet.Description To descr
   Set aPet.Colour To col
   fPets.add(aPet)
End Method
```
(b)
```
Method speciesCount(thisSpecies As String) As Integer
   count As Integer
   Set count To 0
   For Each aPet As Pet In fPets
      If aPet.Species = thisSpecies Then
         Set count To count + 1
      End If
    End For
    Return count
End Method
```
(c)
(1) declares thisPet as a [local] object of class Pet
(2) declares index as a [local] integer
(3) initialises index to 0 [the first index for Pets]
(4) sets thisPet to the first pet in Pets
(5) sets up a loop that continues until we find the pet with the right name or we get to the end of the collection (Note: last index is Pets.Count – 1, if the name is that of the last one, the return statement will pick it up.)
(6) moves onto the next index position
(7) sets thisPet to the pet at the new position in Pets

(8) marks the end of the loop
(9) Returns True or False depending on whether the name has been found.
(d)
You need to check that the list is not empty (as the precondition of isPet) and that the pet's name is in the collection (using isPet); i.e. check that the count of the Pets property is greater than 0, and if it is, invoke the isPet method and only proceed if it returns True.
[Note that the precondition on isPet was chosen in order to make this question more challenging - a better design would not use such a precondition, but would return False if the list is empty.]

Q10
(a)

```
Method doesItContain(word As String) As Boolean
   found As Boolean
   Set found To False
   index As Integer
   Set index to 0
   While (index <= Text.Length – word.Length And Not found)
      If Text.substring(index, word.Length) = word Then
         Set found To True
      Else
         Set index To index + 1
      End If
   End While
   Return found
End Method
```

(b) Examples: The best three are probably the ones in bold.
"Sir" - it should not be in the string (you should include a test like this)
"sheep" - something that appears in the string
"?" - boundary value (last character) and also smallest string
"B" or "Baa" – appears at the very beginning of the string
"Baa baa, black sheep, have you any wool?" – boundary value (whole string)
(c)

```
Method everyOther() As String
    returnString As String
    Set returnString To ""
    'Note that the If is not actually required as the loop will be skipped
    'for an empty string.
    'Exam technique: if you decide to leave it out, put in a comment
    'to show you have thought about it.
    If Not Text = "" Then
       For index As Integer From 0 To Text.Length-1 Step 2
          Set returnString To returnString + Text.Chars(index).toString()
       End For
    End If
    Return returnString
End Method
```

(d) Examples: The best three are probably the ones in bold.
“” – the empty string (essential to include this)
“z” – a single character
“xy” – two characters
“abc%+” – multiple characters to be selected from a string of odd length
“f?ghijk” – as above, but a string of even length

(e) editedText returns a copy of Text that has had any occurrence of a double space just after a '.', '!' or '?' changed to a single space.

Q11
(a) A good solution is the use of a menu strip with a File menu that has a Save and an Exit menu item, together with two data grid views, one for the shop details and one for the delivery details.

(b) The shop data grid view should have all its editing permissions switched off. The delivery data grid view should have all its editing permissions switched on.

(c) Extract from the main property table for the Shop Delivery application

Components			
DataSet	shopDeliveryDataSet		
BindingSource	shopBindingSource	DataSource	shopDeliveryDataSet
		DataMember	Shop
ShopTableAdapter	shopTableAdapter		
BindingSource	fkDeliveryShop-BindingSource	DataSource	shopBindingSource
		DataMember	FKDeliveryShop
DeliveryTableAdapter	deliveryTableAdapter		

(d) Extract from the main property table for the Shop Delivery application

Controls			
DataGridView	shopDGView	DataSource	shopBindingSource
DataGridView	deliveryDGView	DataSource	fkDeliveryShop-BindingSource

(e) The OnClick event handler of the Save menu item.

```
'Save updates to the database.
fkDeliveryShopBindingSource.endEdit()
Try
   deliveryTableAdapter.update(shopDeliveryDataSet.Delivery)
Catch ex As Exception
   MessageBox.Show("A problem has occurred when updating the database")
End Try
```

*** End of sample answers ***