

eLearning

anytime,



anywhere
education

EliteCertify

LEADING THE WAY TO SUCCESS

70-316 DEVELOPING WINDOWS-BASED APPLICATIONS

WITH MICROSOFT VISUAL C# .NET DEMO



© Copyright 1997-2003, EliteCertify.com, All Rights Reserved.

You develop a Windows-based application that enables to enter product sales. You add a subroutine named `EliteCertify`.

You discover that `EliteCertify` sometimes raises an `IOException` during execution. To address this problem you create two additional subroutines named `LogError` and `CleanUp`. These subroutines are governed by the following rules:

- `LogError` must be called only when `EliteCertify` raises an exception.
- `CleanUp` must be called whenever `EliteCertify` is complete.

You must ensure that your application adheres to these rules. Which code segment should you use?

- A.

```
try {
    EliteCertify();
    LogError();
}
catch (Exception e) {
    CleanUp(e);
}
```
- B.

```
try {
    EliteCertify();
}
catch (Exception e) {
    LogError(e);
    CleanUp();
}
```
- C.

```
try {
    EliteCertify();
}
catch (Exception e) {
    LogError(e);
}
finally {
    CleanUp();
}
```
- D.

```
try {
    EliteCertify();
}
catch (Exception e) {
    CleanUp(e);
}
finally {
    LogError();
}
```

Answer: C

Explanation: We must use a **try...catch...finally** construct. First we run the EliteCertify() code in the **try** block. Then we use the LogError() subroutine in the **catch** statement since all exceptions are handled here. Lastly we put the CleanUp() subroutine in the **finally** statement since this code will be executed regardless of whether an exception is thrown or not.

Reference: 70-306/70-316 Training kit, Page 237.

Incorrect Answers

- A:** LogError should not run each time, only when an exception occurs. It should be placed in the **catch** block, not in the **try** block.
- B:** CleanUp should not run only when an exception occurs. It should run when no exception occurs as well. It should be put in the **finally** block not in the **catch** block.
- D:** CleanUp must be put in the finally block, and LogError in the catch block. Not the opposite way around.

You use Visual Studio .NET to create a Windows-based application. The application includes a form named TestKForm, which displays statistical data in graph format. You use a custom graphing control that does not support resizing.

You must ensure that users cannot resize, minimize, or maximize TestKForm. Which three actions should you take? (Each answer presents part of the solution. Choose three)

- A. Set TestKForm.MinimizeBox to **False**.
- B. Set TestKForm.MaximizeBox to **False**.
- C. Set TestKForm.ControlBox to **False**.
- D. Set TestKForm.ImeMode to **Disabled**.
- E. Set TestKForm.WindowState to **Maximized**.
- F. Set TestKForm.FormBorderStyle to one of the Fixed Styles.
- G. Set TestKForm.GridSize to the appropriate size.

Answer: A, B, F

Explanation: We disable the Minimize and Maximize buttons with the TestKForm.MinimizeBox and the TestKForm.MaximizeBox properties. Furthermore we should use a fixed FormBorderStyle to prevent the users from manually resizing the form.

Reference:

Visual Basic and Visual C# Concepts, Changing the Borders of Windows Forms
 .NET Framework Class Library, Form.MinimizeBox Property [C#]
 .NET Framework Class Library, Form.MaximizeBox Property [C#]

You develop an application that includes a Contact Class. The contact class is defined by the following code:

```
public class Contact{
    private string name;
    public event EventHandler ContactSaved;

    public string Name {
        get {return name;}
        set {name = value;}
    }

    public void Save () {
        // Insert Save code.
        // Now raise the event.
        OnSave();
    }

    public virtual void OnSave() {
        // Raise the event:
        if (ContactSaved != null) {
            ContactSaved(this, null);
        }
    }
}
```

You create a form named EliteCertifyForm. This form must include code to handle the ContactSaved event raised by the Contact object. The Contact object will be initialized by a procedure named CreateContact.

Which code segment should you use?

- A.

```
private void HandleContactSaved() {
    // Insert event handling code.
}
```
- ```
private void CreateContact() {
 Contact oContact = new Contact();
 oContact.ContactSaved +=
 new EventHandler(HandleContactSaved);
 oContact.Name = "EliteCertify";
 oContact.Save();
}
```
- B. 

```
private void HandleContactSaved(
 object sender, EventArgs e) {
 // Insert event handling code.
}
```

```

private void CreateContact() {
 Contact oContact = new Contact();
 oContact.Name = "EliteCertify";
 oContact.Save();
}
C. private void HandleContactSaved(
 object sender, EventArgs e) {
 // Insert event handling code.
}

private void CreateContact() {
 Contact oContact = new Contact();
 oContact.ContactSaved +=
 new EventHandler (HandleContactSaved);
 oContact.Name = "EliteCertify";
 oContact.Save();
}
D. private void HandleContactSaved(Object sender, EventArgs e) {
 // Insert event-handling code.
}

private void CreateContact() {
 Contact oContact = new Contact();
 new EventHandler(HandleContactSaved);
 oContact.Name = "EliteCertify";
 oContact.Save();
}

```

**Answer: C**

**Explanation:** The delegate is correctly declared with appropriate parameters:

```
private void HandleContactSaved(object sender, EventArgs e)
```

The association between the delegate and the event is correctly created with the += operator:

```
oContact.ContactSaved += new EventHandler (HandleContactSaved)
```

**Note:** An event handler is a method that is called through a delegate when an event is raised, and you must create associations between events and event handlers to achieve your desired results. In C# the += operator is used to associate a delegate with an event..

**Reference:** 70-306/70-316 Training kit, Implementing Event Handlers, Pages 143-144

**Incorrect Answers**

**A:** The declaration of the delegate do not contain any parameters.

```
private void HandleContactSaved()
```

**B:** There is no association made between the delegate and the event.