

考試編碼:70-485

**考試名稱: Advanced Windows Store
App Development using C#**

版本: Demo

Topic 1, Scenario Margie's Travel

Background

You are developing a Windows Store media sharing app for the sales and marketing team at Margie's Travel. The app will allow team members to download documents and media about current and proposed products and services from the company's cloud-based media manager service. Team members will be able to add new content to the cloud service and to print and share content.

Business Requirements

Behavior:

Team members must be able to download product information data sheets, marketing materials, and product demonstration video clips from the company's server.

Team members must be able to select and upload multiple files that contain new and modified content as a batch.

Team members must be able to stream video clips to other devices in the vicinity of the team member's device. The app will not support the streaming of photographs.

The app must allow team members to pause, restart, or cancel uploads and downloads of files. The app must report both the progress and completion status of these operations. It must also return results about upload and download operations.

User Interface:

The app must include a photo viewer. When photos are added or deleted in the photo viewer window, they must animate in and out of the field of view. Remaining photos must move to fill the empty space created when photos are deleted. The photo viewer must support semantic zoom.

The app must display information on the lock screen of the device. The information must include text-based alerts and a value indicating the number of pending file downloads.

Technical Requirements

Behavior:

The company has an existing component named VideoProcessor. This component compresses video clips and performs other processing before the video clips are uploaded to the media manager service. The component was written with managed code. The VideoProcessor component will also be used by Windows Store apps developed in HTML5 and JavaScript. The apps must be able to call the overload of the ProcessVideo() method that accepts a string and a Boolean value as parameters.

When a team member selects a video clip to download, the app must download the file as a background task. After a download has started, the app should maintain the network connection to the server even when the app is suspended.

User Interface:

The app must include a custom photo viewer control. The control will be updated frequently and may be deployed separately from the rest of the app. The photo viewer control must support templates and styles.

User Interface:

The app must include a custom photo viewer control. The control will be updated frequently and may be deployed separately from the rest of the app. The photo viewer control must support templates and styles. The app must use a Grid control as the root layout control. The photo viewer must be placed in the second row of the grid.

The appearance of the app must change when the app is in snapped mode. The first row of the root layout grid must not change height. The second row must fill all available space.

Available video clips must be displayed in an extended ListView control class named Downloaded VideoList.

The template for the DownloadedVideoList is already defined. New video clips should be added to DownloadedVideoList when the DownloadVideoQ method completes.

New video clip items in the DownloadedVideoList should color change periodically to alert the team member.

Application Structure

Relevant portions of the app files are as follows. (Line numbers in the code segments are included for reference only and include a two-character prefix that denotes the specific file to which they belong.)

App.xaml.cs

```
AP01 cts= newCancellationTokenSource();
AP02 private List<DownloadOperation>MyPendingDownloads;
AP03 privateasyncTaskHandleMyPendingDownloads(DownloadOperationdownload,
boolstart)
AP04 {
AP05     MyPendingDownloads.Add(download);
AP06     Progress<DownloadOperation> progressCallback = new
Progress<DownloadOperation>(DownloadProgress);
AP07     if(start)
AP08     {
AP09         awaitdownload.StartAsync().AsTask(cts.Token, progressCallback);
AP10     }
AP11     else
AP12     {
AP13         awaitdownload.AttachAsync().AsTask(cts.Token, progressCallback);
AP14     }
AP15 }
AP16 privateasyncvoidUploadContent()
AP17 {
AP18     FileOpenPickerpicker = newFileOpenPicker();
AP19
AP20     List<BackgroundTransferContentPart> uploadGrp = new
List<BackgroundTransferContentPart>();
AP21     for(intfileNum = 0; fileNum < files.Count; fileNum ++)
AP22     {
AP23         BackgroundTransferContentPartuploadItem= new
BackgroundTransferContentPart("File"+ fileNum,
files[fileNum].Name);
AP24         uploadItem.SetFile(files[fileNum]);
AP25         uploadGrp.Add(uploadItem);
AP26     }
AP27     BackgroundUploaderuploader = newBackgroundUploader();
AP28
AP29     awaitHandleUploadAsync(upload, true);
AP30 }
```

VideoProcessor.cs

```
IP01 publicclassVideoProcessor
IP02 {
IP03
IP04     publicVideoProcessor(intvideoID)
IP05     {
IP06         ...
IP07     }
IP08
IP09     publicVideoProcessor(stringvideoName)
IP10     {
IP11         ...
IP12     }
IP13
IP14
IP15     publicvoidProcessVideo(stringvideoName, stringvideoType)
IP16     {
IP17         ...
IP18     }
IP19
IP20     publicvoidProcessVideo(stringvideoName, boolcompressFile)
IP21     {
IP22         ...
IP23     }
IP24 }
```

MainPage.xaml

```
MP01 <Grid x:Name="LayoutRoot">
MP02     <Grid.RowDefinitions>
MP03         <RowDefinitionHeight="100"/>
MP04         <RowDefinitionHeight="200"/>
MP05     </Grid.RowDefinitions>
MP06     <VisualStateManager.VisualStateGroups>
MP07
MP08     </VisualStateManager.VisualStateGroups>
MP09 </Grid>
```

MainPage.xaml.cs

```
MC01 private PlayToManagerptMgr = PlayToManager.GetForCurrentView();
MC02
MC03 protectedoverridevoidOnNavigatedTo(NavigationEventArgs)
MC04 {
MC05
MC06
MC07 }
MC08 privatevoidSourceRequestHandler(PlayToManagersender,
    PlayToSourceRequestedEventArgs)
MC09 {
MC10
MC11     e.SourceRequest.SetSource(mediaElement.PlayToSource);
MC12 }
MC13 publicvoidStartNewVideoAnimation()
MC14 {
MC15     NewVideoStoryboard.Begin();
MC16 }
MC17 publicvoidDownloadVideo(stringvideoName)
MC18 {
MC19     ...
MC20     videoList.Items.Add(videoName);
MC21 }
```

QUESTION:1

- (Topic 1)

You need to meet the business requirements about downloading and uploading.

How should you configure the app? (To answer, select the appropriate options from each drop-down list in the answer area.)

Configure the Application UI settings in Package.appxmanifest

Lock screen notifications:

- Tile
- Badge and Tile Text
- Wide Logo Only

Logo files:

- Tile Image Only
- Badge Logo and Tile Image
- Badge Logo and Wide Logo

Configure the Declarations settings in Package.appxmanifest

Add a Background Task declaration and configure support for the following task types:

- Photo file stream
- Control channel
- User actions
- Device availability
- Playback status

Answer:

Configure the Application UI settings in Package.appxmanifest

Lock screen notifications:

- Tile
- Badge and Tile Text
- Wide Logo Only

Logo files:

- Tile Image Only
- Badge Logo and Tile Image
- Badge Logo and Wide Logo

Configure the Declarations settings in Package.appxmanifest

Add a Background Task declaration and configure support for the following task types:

- Photo file stream
- Control channel
- User actions
- Device availability
- Playback status

QUESTION:2

- (Topic 1)

You need to implement the business requirement to display video clips.

Which code segment should you use in the MainPage.xaml file?

- A.

```
<DownloadedVideoList x:Name="videoList">
  <DownloadedVideoList.Resources>
    <Storyboard x:Name="NewVideoStoryboard">
      <ColorAnimation Storyboard.TargetName="NewVideoBrush"
        Storyboard.TargetProperty="Color" From="Red" To="Green"
        Duration="0:0:8" RepeatBehavior="Forever"/>
    </Storyboard>
  </DownloadedVideoList.Resources>
  <DownloadedVideoList.Background>
    <SolidColorBrush x:Name="NewVideoBrush" Color="Red"/>
  </DownloadedVideoList.Background>
</DownloadedVideoList>
```
- B.

```
<DownloadedVideoList x:Name="videoList">
  <DownloadedVideoList.Resources>
    <Storyboard x:Name="NewVideoStoryboard">
      <ColorAnimation Storyboard.TargetName="NewVideoBrush"
        Storyboard.TargetProperty="Color" From="Red" To="Green"
        AutoReverse="true"/>
    </Storyboard>
  </DownloadedVideoList.Resources>
  <DownloadedVideoList.Background>
    <SolidColorBrush x:Name="NewVideoBrush" Color="Red"/>
  </DownloadedVideoList.Background>
</DownloadedVideoList>
```
- C.

```
<DownloadedVideoList x:Name="videoList">
  <DownloadedVideoList.Transitions>
    <TransitionCollection>
      <EntranceThemeTransition/>
    </TransitionCollection>
  </DownloadedVideoList.Transitions>
</DownloadedVideoList>
```
- D.

```
<DownloadedVideoList x:Name="videoList">
  <DownloadedVideoList.Transitions>
    <TransitionCollection>
      <AddDeleteThemeTransition/>
    </TransitionCollection>
  </DownloadedVideoList.Transitions>
</DownloadedVideoList>
```

- A. Option A
B. Option B
C. Option C
D. Option D

Answer: A

QUESTION:3

- (Topic 1)

You need to implement downloading of media files and other content.

Which code segment should you add to App.xaml.cs?

- A.

```
private async Task GetPendingDownloadsList()
{
    IReadOnlyList<DownloadOperation> downloads = await
        BackgroundDownloader.GetCurrentDownloadsAsync();
    if (downloads.Count > 0)
    {
        List<Task> myTasks = new List<Task>();
        for (int i=0; i < downloads.count; i++)
        {
            await HandleMyPendingDownloads(downloads[i], true);
        }
        await Task.WhenAll(myTasks);
    }
}
```
- B.

```
private async Task GetPendingDownloadsList()
{
    IReadOnlyList<DownloadOperation> downloads = await
        BackgroundDownloader.GetCurrentDownloadsAsync();
    if (downloads.Count > 0)
    {
        List<Task> myTasks = new List<Task>();
        foreach (DownloadOperation download in downloads)
        {
            myTasks.Add(HandleDownloadAsync(download, false));
        }
        await Task.WhenAll(myTasks);
    }
}
```
- C.

```
private GetPendingDownloadsList()
{
    IReadOnlyList<DownloadOperation> downloads = await
        BackgroundDownloader.GetCurrentDownloadsAsync();
    if (downloads.Count > 0)
    {
        List<Task> myTasks = new List<Task>();
        for (int i=0; i < downloads.count; i++)
        {
            await HandleMyPendingDownloads(downloads[i], true);
        }
        await Task.WhenAll(myTasks);
    }
}
```
- D.

```
private Task GetPendingDownloadsList()
{
    IReadOnlyList<DownloadOperation> downloads =
        BackgroundDownloader.CreateDownloadAsync();
    if (downloads.Count > 0)
    {
        List<Task> myTasks = new List<Task>();
        foreach (DownloadOperation download in downloads)
        {
            myTasks.Add(HandleDownloadAsync(download, false));
        }
        Task.WhenAll(myTasks);
    }
}
```

A. Option A

- B. Option B
- C. Option C
- D. Option D

Answer: B

QUESTION:4

- (Topic 1)

You need to ensure that the app uploads media and files to the media manager service.

What should you do? (Each correct answer presents part of the solution. Choose all that apply.)

- A. Insert the following line of code at line AP28.

```
IReadOnlyList<UploadOperation> upload =  
await BackgroundUploader.GetCurrentUploadsAsync();
```

- B. Insert the following line of code at line AP28.

```
UploadOperation upload = await uploader.CreateUpload(uri, uploadGrp);
```

- C. Insert the following line of code at line AP28.

```
UploadOperation upload = await uploader.CreateUploadAsync(uri, uploadGrp);
```

- D. Insert the following line of code at line AP19.

```
IReadOnlyList<StorageFile> files = await picker.PickMultipleFilesAsync  
();
```

- E. Insert the following line of code at line AP19.

```
IReadOnlyList<StorageFile> files = await picker.PickSingleFilesAsync  
();
```

- A. Option A
- B. Option B
- C. Option C
- D. Option D
- E. Option E

Answer: C,D

QUESTION:5

- (Topic 1)

You need to implement the photo viewer page to meet the business requirements.

How should you complete the code segment? (To answer, drag the appropriate [source or sources] to the correct location or locations in the answer area.)

<pre><RepositionThemeTransition/> <AddDeleteThemeTransition/> <ReorderThemeTransition/> <EntranceThemeTransition/> <ViewBox/> <GridView/> <FlipView/> <WrapGrid/></pre>	<p>Answer Area</p> <pre><Button Content="Add New Photo" Click="btnAdd_Click"/> <Button Content="Remove Selected Photo" Click="btnDelete_Click" <ItemsControl Grid.Row="1" x:Name="rectangleItems"> <ItemsControl.ItemContainerTransitions> <TransitionCollection> [] </TransitionCollection> </ItemsControl.ItemContainerTransitions> <ItemsControl.ItemsPanel> <ItemsPanelTemplate> [] </ItemsPanelTemplate> </ItemsControl.ItemsPanel> </ItemsControl></pre>
---	---

Answer:

<pre><RepositionThemeTransition/> <AddDeleteThemeTransition/> <ReorderThemeTransition/> <EntranceThemeTransition/> <ViewBox/> <GridView/> <FlipView/> <WrapGrid/></pre>	<p>Answer Area</p> <pre><Button Content="Add New Photo" Click="btnAdd_Click"/> <Button Content="Remove Selected Photo" Click="btnDelete_Click" <ItemsControl Grid.Row="1" x:Name="rectangleItems"> <ItemsControl.ItemContainerTransitions> <TransitionCollection> <AddDeleteThemeTransition/> </TransitionCollection> </ItemsControl.ItemContainerTransitions> <ItemsControl.ItemsPanel> <ItemsPanelTemplate> <WrapGrid/> </ItemsPanelTemplate> </ItemsControl.ItemsPanel> </ItemsControl></pre>
---	---

QUESTION:6

- (Topic 1)

You need to implement a custom control to display thumbnail images of video clips.

Which code segment should you use?

- A.

```
public sealed class DownloadedVideoList: FlipView
{
    public DownloadedVideoList()
    {
        this.DefaultStyleKey = typeof(ListView);
    }
}
```
- B.

```
public sealed class DownloadedVideoList: FlipView
{
    public DownloadedVideoList()
    {
        this.DefaultStyleKey = typeof(DownloadedVideoList);
    }
}
```
- C.

```
public sealed class DownloadedVideoList: ListView
{
    public DownloadedVideoList()
    {
        this.DefaultStyleKey = typeof(DownloadedVideoList);
    }
}
```
- D.

```
public sealed class DownloadedVideoList: ListView
{
    public DownloadedVideoList()
    {
        this.DefaultStyleKey = typeof(ListView);
    }
}
```

- A. Option A
- B. Option B
- C. Option C
- D. Option D

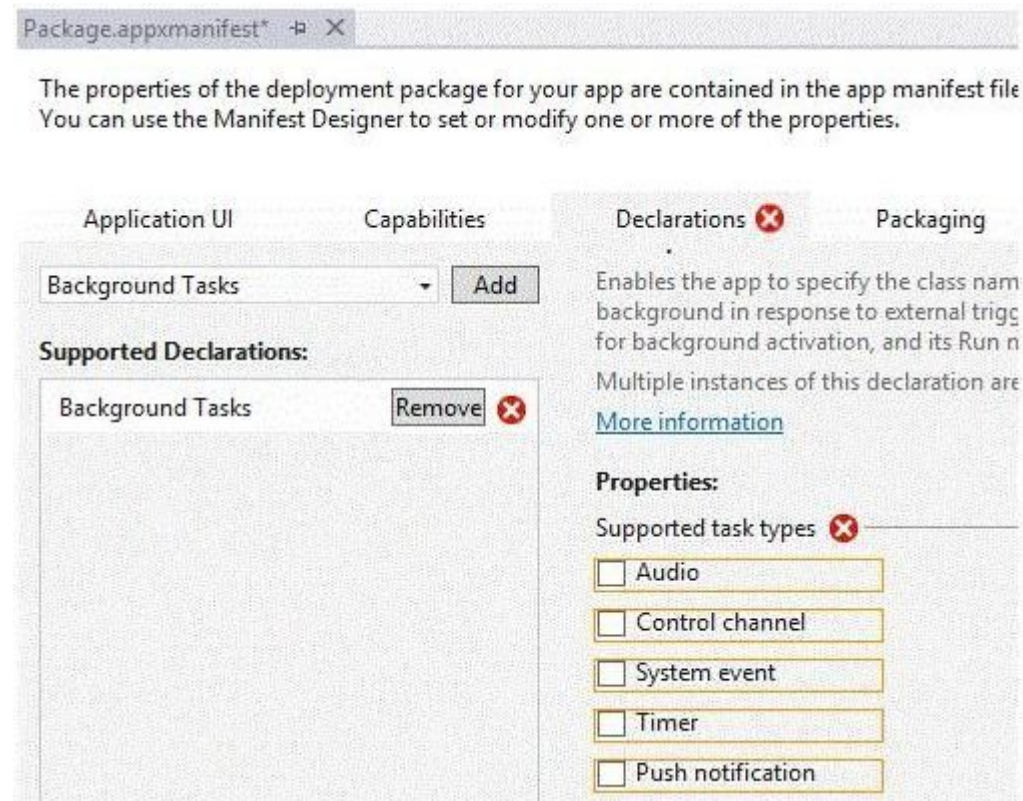
Answer: C

QUESTION:7

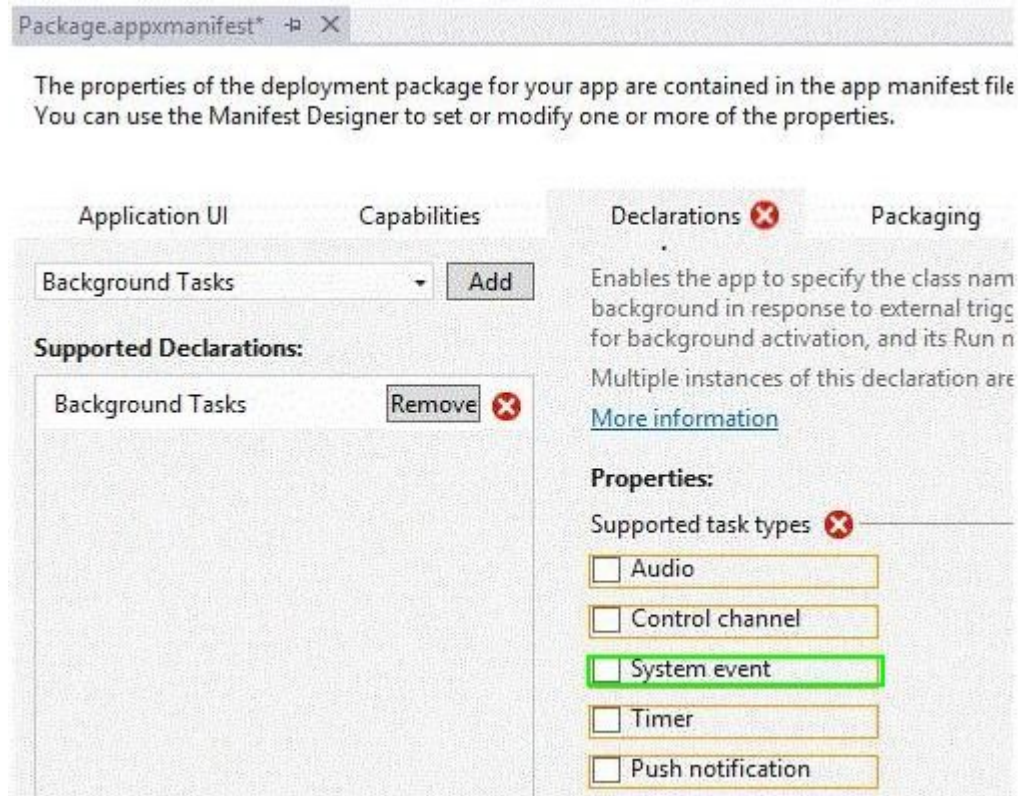
- (Topic 1)

You need to configure the app manifest to support the file download requirements.

Which task type property should you specify? (To answer, select the appropriate property in the answer area.)



Answer:



QUESTION:8
- (Topic 1)

You need to implement the requirements for the playback of media.

What should you do? (Each correct answer presents part of the solution. Choose all that apply.)

- A. Add the following line of code at line MC02.

```
private void ShowPlayTo()
{
Windows.Media.PlayTo.PlayToManager.ShowPlayToUI(); }

```
- B. Add the following line of code at line MC06.

```
ptMgr.DefaultSourceSelection = false;

```
- C. Add the following line of code at line MC10.

```
ptMgr.PlayRequested += SourceRequestHandler;

```
- D. Add the following line of code at line MC05.

```
ptMgr.SourceRequested += SourceRequestHandler;

```

Answer: B,D

QUESTION:9

- (Topic 1)

You need to implement the requirements for the behavior of the main page. Which code segment should you insert at line MP07?

- A. `<VisualStateGroup x:Name="ApplicationViewStates">
 <VisualState x:Name="Snapped">
 <Storyboard>
 <ObjectAnimationUsingKeyFrames Storyboard.TargetName="LayoutRoot"
 Storyboard.TargetProperty="(Grid.RowDefinitions)[1].Height">
 <DiscreteObjectKeyFrame KeyTime="0" Value="Auto"/>
 </ObjectAnimationUsingKeyFrames>
 </Storyboard>
 </VisualState>
</VisualStateGroup>`
- B. `<VisualStateGroup x:Name="ApplicationViewStates">
 <VisualState x:Name="Filled">
 <Storyboard>
 <ObjectAnimationUsingKeyFrames Storyboard.TargetProperty="LayoutRoot">
 <DiscreteObjectKeyFrame KeyTime="0" Value="*" />
 </ObjectAnimationUsingKeyFrames>
 </Storyboard>
 </VisualState>
</VisualStateGroup>`
- C. `<VisualStateGroup x:Name="ApplicationViewStates">
 <VisualState x:Name="FullScreenLandscape">
 <Storyboard>
 <ObjectAnimationUsingKeyFrames Storyboard.TargetProperty="LayoutRoot">
 <DiscreteObjectKeyFrame KeyTime="0" Value="Auto" />
 </ObjectAnimationUsingKeyFrames>
 </Storyboard>
 </VisualState>
</VisualStateGroup>`
- D. `<VisualStateGroup x:Name="ApplicationViewStates">
 <VisualState x:Name="FullScreenPortrait">
 <Storyboard>
 <ObjectAnimationUsingKeyFrames Storyboard.TargetName="LayoutRoot"
 Storyboard.TargetProperty="(Grid.RowDefinitions).Height">
 <DiscreteObjectKeyFrame KeyTime="0" Value="*" />
 </ObjectAnimationUsingKeyFrames>
 </Storyboard>
 </VisualState>
</VisualStateGroup>`

- A. Option A
B. Option B
C. Option C
D. Option D

Answer: A

QUESTION:10

- (Topic 1)

You need to meet the business requirements about downloading and uploading.

How should you configure the app? (To answer, select the appropriate options from each drop-down list in the answer area.)

Configure the Application UI settings in Package.appxmanifest

Lock screen notifications:

Badge
Badge and Tile Text
Badge Logo Only

Logo files:

Badge Logo Only
Wide Logo Only
Badge Logo and Wide Logo

Configure the Declarations settings in Package.appxmanifest

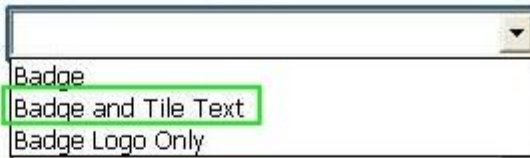
Add a Background Task declaration and configure support for the following task types:

Audio
Control channel
System event
Timer
Push Notification

Answer:

Configure the Application UI settings in Package.appxmanifest

Lock screen notifications:



A dropdown menu with a downward arrow on the right. The menu is open, showing three options: 'Badge', 'Badge and Tile Text', and 'Badge Logo Only'. The option 'Badge and Tile Text' is highlighted with a green box.

Logo files:



A dropdown menu with a downward arrow on the right. The menu is open, showing three options: 'Badge Logo Only', 'Wide Logo Only', and 'Badge Logo and Wide Logo'. The option 'Badge Logo and Wide Logo' is highlighted with a green box.

Configure the Declarations settings in Package.appxmanifest

Add a Background Task declaration and configure support for the following task types:



A dropdown menu with a downward arrow on the right. The menu is open, showing five options: 'Audio', 'Control channel', 'System event', 'Timer', and 'Push Notification'. The option 'Control channel' is highlighted with a green box.

QUESTION:11

- (Topic 1)

You need to ensure that the VideoProcessor component can be used by the Windows Store app.

What should you do? (Each correct answer presents part of the solution. Choose all that apply.)

- A. Add the following attribute to line IP19.
[Windows.Foundation.Metadata.DefaultOverload()]
- B. Replace line IP01 with the following line of code.
static class VideoProcessor
- C. Replace line IP09 with the following line of code.
public VideoProcessor(string videoName, int ID)
- D. Add the following attribute to line IP14.
[Windows.Foundation.Metadata.DefaultOverload()]
- E. Replace line IP01 with the following line of code.
public sealed class VideoProcessor

Answer: A,C,E