**1. C# Windows Forms**

**i) What is C# Windows Forms?** The.NET framework offers the C# Windows Forms (WinForms) GUI class library, which is used to create desktop applications with sophisticated functionality. It provides a selection of pre-made controls (such labels, text boxes, and buttons) for building Windows application user interfaces.

**ii) How do you create a new Windows Forms application in Visual Studio using C#?**

To create a new Windows Forms application in Visual Studio using C#:

1. Open Visual Studio.
2. Select File -> New -> Project.
3. In the "Create a new project" window, select C# and Windows from the filters.
4. Choose Windows Forms App (.NET Framework) and click Next.
5. Provide a project name, location, and solution name.
6. Select the target .NET Framework version and click Create.
7. Use the designer to drag and drop controls from the Toolbox onto the form and configure them using the Properties window.
8. Write event-handling code by double-clicking on controls to generate event handlers.

**2. Adding Controls to a Windows Forms Application**

To add controls, such as buttons or text boxes, to a Windows Forms application:

1. Open the form in the designer by double-clicking on the form file in Solution Explorer.
2. Open the Toolbox by selecting View -> Toolbox.
3. Drag the desired control (e.g.,TextBox) from the Toolbox and drop it onto the form.
4. Set control properties using the Properties window.

**3. Form's Controls Collection**

**i) What is the purpose of the Form's Controls collection?**

The Form's Controls collection includes every control that was added to the form. It permits developers to programmatically manipulate these controls, allowing them to add, delete, and access controls during runtime.

**ii) How do you add, remove, and access controls within a Form dynamically?**

* To add a control:

Button myButton = new Button();

myButton.Text = "Mea ha";

myButton.Location = new Point(80, 20);

this.Controls.Add(myButton);

* To remove a control:

this.Controls.Remove(myButton);

* To access a control:

Button existingButton = (Button)this.Controls["buttonName"];

existingButton.Text = "New Text";

**4. Partial Class in C#**

**i) What is a partial class in C#?**

A class in C# can be divided into several files by using a partial class. This is beneficial for managing extensive classes, segregating automatically generated code, and enabling teamwork in development.

**ii) How does it allow a class to be defined in multiple files?**

A class can be split across multiple files by utilizing the partial keyword. The compiler merges the partial class parts into one class during compilation.

**5. Partial Methods in C#**

**i) How do you implement partial methods in C#?**

Partial methods are defined in a partial class by using the partial keyword. Method declarations without implementations are allowed, which can be provided in another section of the partial class.

* To declare a partial method:

public partial class MyClass

{

partial void MyPartialMethod();

}

* To implement a partial method:

public partial class MyClass

{

partial void MyPartialMethod()

{

}

}

**ii) What are the key characteristics of partial methods?**

* Declared within partial classes or structs.
* Implicitly private and cannot have access modifiers.
* Must return void; cannot have return types.
* Useful for extending auto-generated code.

**6. Declaring a Partial Class**

**i) How do you declare a partial class?**

Indicate that a class definition is divided among multiple files by using the partial keyword before the class keyword to declare a partial class.

**ii) Provide an example of a partial class declaration.**

Example:

* File: MyClassPart1.cs

public partial class MyClass

{

public void Method1()

{

Console.WriteLine("Method1");

}

}

* File: MyClassPart2.cs

csharp

Copy code

public partial class MyClass

{

public void Method2()

{

Console.WriteLine("Method2");

}

}

* Usage:

MyClass myClass = new MyClass();

myClass.Method1(); // Output: Method1

myClass.Method2(); // Output: Method2