C# Program to Copy File Content

```
csharp
Copy code
using System;
using System.IO;

class FileCopy
{
    static void Main()
    {
        // Specify the source and destination file paths
        string sourceFile = @"C:\SourceFolder\sourcefile.txt";
        string destinationFile =
@"C:\DestinationFolder\destinationfile.txt";

        try
        {
            // Check if the source file exists
            if (File.Exists(sourceFile))
            {
                // Copy the source file to the destination file
                File.Copy(sourceFile, destinationFile, true);
                // "true" overwrites the file if it exists

                Console.WriteLine("File copied successfully.");
            }
            else
            {
                Console.WriteLine("Source file does not exist.");
            }
        }
        catch (Exception ex)
        {
            // Handle any exceptions that occur during the copy operation
            Console.WriteLine("An error occurred: " + ex.Message);
        }
    }
}
```

C# Program Using BinaryReader and BinaryWriter

```
csharp
Copy code
using System;
using System.IO;

class Program
{
    static void Main()
```

```
    {
        // Specify the file path
        string filePath = @"C:\Temp\binarydata.dat";

        // Writing to the binary file
        WriteToBinaryFile(filePath);

        // Reading from the binary file
        ReadFromBinaryFile(filePath);
    }

    // Method to write data to a binary file
    static void WriteToBinaryFile(string filePath)
    {
        try
        {
            using (BinaryWriter writer = new
BinaryWriter(File.Open(filePath, FileMode.Create)))
            {
                // Example data to write
                int[] numbers = { 1, 2, 3, 4, 5 };

                foreach (int number in numbers)
                {
                    writer.Write(number); // Write each integer to the file
                }

                Console.WriteLine("Data written to binary file
successfully.");
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine("Error writing to file: " + ex.Message);
        }
    }

    // Method to read data from a binary file
    static void ReadFromBinaryFile(string filePath)
    {
        try
        {
            using (BinaryReader reader = new
BinaryReader(File.Open(filePath, FileMode.Open)))
            {
                Console.WriteLine("Data read from binary file:");

                // Read until the end of the file
                while (reader.BaseStream.Position !=
reader.BaseStream.Length)
                {
                    int number = reader.ReadInt32(); // Read each integer
from the file
                    Console.WriteLine(number);
                }
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine("Error reading from file: " + ex.Message);
        }
```

```
    }
}
```

Example of Client-Server Communication using Sockets in C#

Server Code

```
csharp
Copy code
using System;
using System.Net;
using System.Net.Sockets;
using System.Text;

class Server
{
    static void Main()
    {
        // Create a TCP/IP socket
        TcpListener serverSocket = new TcpListener(IPAddress.Any, 8080);
        serverSocket.Start();
        Console.WriteLine("Server started. Waiting for a connection...");

        // Accept an incoming connection
        using (TcpClient clientSocket = serverSocket.AcceptTcpClient())
        {
            Console.WriteLine("Client connected.");

            // Get the stream to read/write data
            NetworkStream stream = clientSocket.GetStream();
            byte[] buffer = new byte[1024];

            // Read data from the client
            int bytesRead = stream.Read(buffer, 0, buffer.Length);
            string message = Encoding.ASCII.GetString(buffer, 0,
bytesRead);
            Console.WriteLine("Received from client: " + message);

            // Send a response back to the client
            string response = "Hello from the server!";
            byte[] responseBytes = Encoding.ASCII.GetBytes(response);
            stream.Write(responseBytes, 0, responseBytes.Length);
            Console.WriteLine("Response sent to client.");
        }

        // Stop the server
        serverSocket.Stop();
        Console.WriteLine("Server stopped.");
    }
}
```

Client Code

```
csharp
Copy code
using System;
using System.Net.Sockets;
using System.Text;
```

```
class Client
{
    static void Main()
    {
        // Create a TCP/IP socket
        using (TcpClient clientSocket = new TcpClient("127.0.0.1", 8080))
        {
            Console.WriteLine("Connected to server.");

            // Get the stream to read/write data
            NetworkStream stream = clientSocket.GetStream();

            // Send a message to the server
            string message = "Hello from the client!";
            byte[] messageBytes = Encoding.ASCII.GetBytes(message);
            stream.Write(messageBytes, 0, messageBytes.Length);
            Console.WriteLine("Message sent to server: " + message);

            // Read the response from the server
            byte[] buffer = new byte[1024];
            int bytesRead = stream.Read(buffer, 0, buffer.Length);
            string response = Encoding.ASCII.GetString(buffer, 0,
bytesRead);
            Console.WriteLine("Received from server: " + response);
        }
    }
}
```

Here's a complete C# example that demonstrates how to store employee details in a SQL Server database and display the details in a `GridView` using Windows Forms. This example assumes you have a SQL Server database set up and that you can connect to it.

Step 1: Set Up the Database

1. **Create a Database**: Create a new database called `EmployeeDB`.
2. **Create a Table**: Run the following SQL command to create an `Employees` table:

```
sql
Copy code
CREATE TABLE Employees (
    Id INT PRIMARY KEY IDENTITY(1,1),
    Name NVARCHAR(100),
    Position NVARCHAR(100),
    Salary DECIMAL(18, 2)
);
```

Step 2: Create a Windows Forms Application

1. **Open Visual Studio** and create a new **Windows Forms App** (.NET Framework).
2. **Add Controls**: Drag and drop the following controls onto your form (`Form1`):
    - 3 `TextBox` controls for entering `Name`, `Position`, and `Salary`.
    - 1 `Button` control for saving the employee details (label it "Add Employee").

- 1 `Button` control for loading employee details (label it "Load Employees").
- 1 `DataGridView` control for displaying employee details.

Step 3: Add Required NuGet Package

Ensure you have the `System.Data.SqlClient` library. You can install it via NuGet Package Manager if needed.

Step 4: Write the Code

Below is the complete code for your `Form1.cs`:

```
csharp
Copy code
using System;
using System.Data;
using System.Data.SqlClient;
using System.Windows.Forms;

namespace EmployeeManagement
{
    public partial class Form1 : Form
    {
        // Connection string to connect to SQL Server
        private string connectionString =
@"Server=YOUR_SERVER_NAME;Database=EmployeeDB;Trusted_Connection=True;";

        public Form1()
        {
            InitializeComponent();
        }

        // Method to add employee details to the database
        private void btnAddEmployee_Click(object sender, EventArgs e)
        {
            string name = txtName.Text;
            string position = txtPosition.Text;
            decimal salary;
            if (!decimal.TryParse(txtSalary.Text, out salary))
            {
                MessageBox.Show("Please enter a valid salary.");
                return;
            }

            using (SqlConnection connection = new
SqlConnection(connectionString))
            {
                string query = "INSERT INTO Employees (Name, Position,
Salary) VALUES (@Name, @Position, @Salary)";
                using (SqlCommand command = new SqlCommand(query,
connection))
                {
                    command.Parameters.AddWithValue("@Name", name);
                    command.Parameters.AddWithValue("@Position", position);
                    command.Parameters.AddWithValue("@Salary", salary);

                    connection.Open();
                    command.ExecuteNonQuery();
```

```
                    MessageBox.Show("Employee added successfully.");
                }
            }

            // Clear the text boxes after adding
            txtName.Clear();
            txtPosition.Clear();
            txtSalary.Clear();
        }

        // Method to load employee details into the DataGridView
        private void btnLoadEmployees_Click(object sender, EventArgs e)
        {
            using (SqlConnection connection = new
SqlConnection(connectionString))
            {
                string query = "SELECT * FROM Employees";
                SqlDataAdapter adapter = new SqlDataAdapter(query,
connection);
                DataTable dataTable = new DataTable();

                connection.Open();
                adapter.Fill(dataTable);
                dataGridView1.DataSource = dataTable;
            }
        }
    }
}
```

```
//what is a deligate:
//it is the representative of class
//A delegate in C# is a type that represents references to methods with a
specific parameter list and return type.
//Delegates are used to pass methods as arguments to other methods, allowing
for flexible and reusable code.
using System;

namespace Deligate_Generics
{
    internal class Program
    {
        // Define a delegate that can reference any method that takes two
integers and returns an integer
        public delegate int SumDeligate(int x, int y);

        static void Main(string[] args)
        {
            // Instantiate the delegate, pointing it to the Add method
            SumDeligate sd = Add;

            // Use the delegate to call the method
            int result = sd(5, 3);
```

```
            // Print the result
            Console.WriteLine("The sum is: " + result);

            // You can also change the delegate to point to another method
            sd = Subtract;
            result = sd(5, 3);
            Console.WriteLine("The difference is: " + result);
        }

        // A method that matches the delegate signature
        public static int Add(int a, int b)
        {
            return a + b;
        }

        // Another method that matches the delegate signature
        public static int Subtract(int a, int b)
        {
            return a - b;
        }
    }
}
```

ANONYMUS

```
using System;

internal class Program
{
    public delegate int SumDelegate(int x, int y);

    public static void Main(string[] args)
    {
        SumDelegate sd1 = delegate (int x, int y) { return x + y; };
        Console.WriteLine(sd1(20, 30));
    }
}
```