4.5 Interfaces 

This section will guide you to:

* Create a Windows Console project in Visual Studio to demonstrate the use of Interfaces in Classes
* Create an Interface IPerson which will store details for all people-entities
* Create classes for students, teachers, and office staff which will implement the IPerson interface

**Development Environment**

* Visual Studio 2019 Community Version
* Windows 10

This guide has nine subsections, namely:

* + 1. Creating a Windows Console project in Visual Studio to demonstrate the use of Interfaces in classes
    2. Creating an IPerson interface for people-based classes
    3. Creating a Student class to store student data and implement IPerson
    4. Creating a Teacher class to store teacher data and implement IPerson
    5. Creating an OfficeStaff class to store office staff data and implement IPerson
    6. Adding a method, runApp(), in Program class to create objects and populate them
    7. Building the project
    8. Publishing and running the project
    9. Pushing the code to your GitHub repositories

**Step 4.5.1:** Creating a Windows Console project in Visual Studio to demonstrate the use of interfaces in classes

* Open Visual Studio.
* From the top menu, select **File->New->Project.**
* In the **Create A New Project** screen, select **Console app (.NET Core)** from the list of available project types and click on **Next.**
* Enter **Project Name** as **Phase1Section5.12** and click on **Create.**
* This will create the files for a Windows Console project.

**Step 4.5.2:** Creating an IPerson interface for people-based classes

* In the **Solution Explorer** window, right click **Phase1Section5.12** and choose **Add->Class.**
* Enter Class name as IPerson.cs and click **Add.**
* Enter the following code:

**using** System;

**using** System.Collections.Generic;

**using** System.Text;

**namespace** Phase1Section5.\_12

{

**public** **interface** IPerson

{

**string** getInfo();

**string** getTypeOfPerson();

}

}

**Step 4.5.3:** Creating a Student class to store student data and implement IPerson

* In the **Solution Explorer** window, right click **Phase1Section5.12** and choose **Add->Class.**
* Enter Class name as Student.cs and click **Add.**
* Enter the following code:

**using** System;

**using** System.Collections.Generic;

**using** System.Text;

**namespace** Phase1Section5.\_12

{

**public** **class** Student : IPerson

{

**private** **string** name;

**private** **string** address;

**private** DateTime dateOfJoining;

**public** **string** Name

{

**get** { **return** name; }

**set** { name = value; }

}

**public** **string** Address

{

**get** { **return** address; }

**set** { address = value; }

}

**public** DateTime DateOfJoining

{

**get** { **return** dateOfJoining; }

**set** { dateOfJoining = value; }

}

**public** **string** getInfo()

{

**string** retVal = "Name: " + name + ", Address=" + address + ", Joining Date =" + dateOfJoining.ToShortDateString();

**return** retVal;

}

**public** **string** getTypeOfPerson()

{

**return** "student";

}

}

}

**Step 4.5.4:** Creating a Teacher class to store teacher data and implement IPerson

* In the **Solution Explorer** window, right click **Phase1Section5.12** and choose **Add->Class.**
* Enter Class name as Teacher.cs and click **Add.**
* Enter the following code:

**using** System;

**using** System.Collections.Generic;

**using** System.Text;

**namespace** Phase1Section5.\_12

{

**public** **class** Teacher: IPerson

{

**private** **string** name;

**private** **string** address;

**private** DateTime dateOfJoining;

**public** **string** Name

{

**get** { **return** name; }

**set** { name = value; }

}

**public** **string** Address

{

**get** { **return** address; }

**set** { address = value; }

}

**public** DateTime DateOfJoining

{

**get** { **return** dateOfJoining; }

**set** { dateOfJoining = value; }

}

**public** **string** getInfo()

{

**string** retVal = "Name: " + name + ", Address=" + address + ", Joining Date =" + dateOfJoining.ToShortDateString();

**return** retVal;

}

**public** **string** getTypeOfPerson()

{

**return** "teacher";

}

}

}

**Step 4.5.5:** Creating an OfficeStaff class to store office staff data and implement IPerson

* In the **Solution Explorer** window, right click **Phase1Section5.12** and choose **Add->Class.**
* Enter Class name as OfficeStaff.cs and click **Add.**
* Enter the following code:

**using** System;

**using** System.Collections.Generic;

**using** System.Text;

**namespace** Phase1Section5.\_12

{

**public** **class** OfficeStaff : IPerson

{

**private** **string** name;

**private** **string** address;

**private** DateTime dateOfJoining;

**public** **string** Name

{

**get** { **return** name; }

**set** { name = value; }

}

**public** **string** Address

{

**get** { **return** address; }

**set** { address = value; }

}

**public** DateTime DateOfJoining

{

**get** { **return** dateOfJoining; }

**set** { dateOfJoining = value; }

}

**public** **string** getInfo()

{

**string** retVal = "Name: " + name + ", Address=" + address + ", Joining Date =" + dateOfJoining.ToShortDateString();

**return** retVal;

}

**public** **string** getTypeOfPerson()

{

**return** "OfficeStaff";

}

}

}

**Step 4.5.6:** Adding a method, runApp(), in Program class to create objects and populate them

* Select **Program.cs** as the current Code tab.
* Enter the following code:

**using** System;

**namespace** Phase1Section5.\_12

{

**class** Program

{

**static** **void** Main(**string**[] args)

{

runApp();

}

**public** **static** **void** runApp()

{

Teacher teacher = **new** Teacher();

teacher.Name = "Mr.Teacher";

teacher.Address = "some address";

teacher.DateOfJoining = DateTime.Today;

Student student= **new** Student();

student.Name = "Student name";

student.Address = "some address";

student.DateOfJoining = DateTime.Today;

OfficeStaff staff = **new** OfficeStaff();

staff.Name = "Mr.Staff";

staff.Address = "some address";

staff.DateOfJoining = DateTime.Today;

Console.WriteLine(teacher.getInfo() + "\n" + teacher.getTypeOfPerson());

Console.WriteLine(student.getInfo() + "\n" + student.getTypeOfPerson());

Console.WriteLine(staff.getInfo() + "\n" + staff.getTypeOfPerson());

}

}

}

**Step 4.5.7:** Building the project

* From the top menu, choose **Build->Build Solution.**
* If any compile errors are shown, fix them as required

**Step 4.5.8:** Publishing and running the project

* From the top menu, select **Debug->Start Without Debugging.**
* This will execute the program in a console window

**Step 4.5.9:** Pushing the code to your GitHub repositories

Open your command prompt and navigate to the folder where you have created your files.

cd <folder path>

Initialize your repository using the following command:

git init

Add all the files to your git repository using the following command:

git add .

Commit the changes using the following command:

git commit -m “Changes have been committed.”

Push the files to the folder you created initially using the following command:

git push -u origin master