# CS 371 Windows Application Development

# HW4: OO Programming/ Interfaces

* Background Material: Deitel Chapters 10-12 & study slides

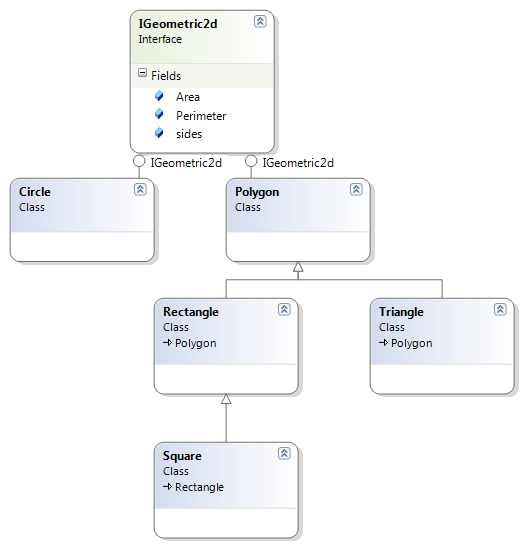
## NAME: DUE: Thurs, Jan 12 Received: .

\*\* Save project files under: **\\CS1\cs\_students\your username\CS371\HW4**

**GRADE:**

|  |  |  |
| --- | --- | --- |
| **CATEGORY** | **POINTS** |  |
| EX5\_1 IGeometric design |  | 5 |
| EX5\_2 Implementation of IGeometric interface |  | 10 |
| EX5\_3 Implementation subclasses |  | 10 |
| EX5\_4 Design and implement multiple Interfaces |  | 40 |
| EX5\_5 Introduce exception handling |  | 20 |
| EX5\_6 Implementing .NET standard Interfaces |  | 40 |
| **TOTAL** |  | 125 |

**EX5\_1: Design a hierarchy** for two-dimensional shapes. Your hierarchy should include a base interface called IGeometric2d, and at least four sub classes. Draw your hierarchy on a sheet of paper. Note that some objects are special cases of other objects. For example, Square is a special case of Rectangle. In these cases, one class should be a subclass (derived class) of the other, rather than directly from the base class of IGeometric2d. Make sure your hierarchy has at least two such subclasses.



**EX5\_2:** Implementyour **IGeometric2d interface** in C#. Two-dimensional geometric objects have area and perimeter. Add to your interface IGeometric2d with member properties double Area and double Perimeter.

**EX5\_3:** Implement your subclasses in C#, each inheriting from and implementing your IGeometric2d interface. Your shape objects should have ways to change the properties of the shape they represent (e.g., users should be able to set the height and width of a rectangle, while only being able to set the height of a square).

**EX5\_4:** Now, in groups of 2 work through section 12.7 in the textbook and implement the **IPayable** interface. You can start with the example code [\\CS1\CS\_ClassData\371\TextExamples\ch12\fig12\_11\_15\](file:///\\CS1\CS_ClassData\371\TextExamples\ch12\fig12_11_15\)

Add **BasePlusCommissionEmployee**, and **CommissionEmployee** to the inheritance hierarchy. In addition implement an **IContact** interface for the classes. The interface is defined:

**public interface IContact**

**{**

**string GetAddress();**

**}**

Your implementation of the method **GetAddress()** should return a string with the mailing address of the object it represents. **Modify your classes appropriately to support this.**

In **PayableInterfaceTest**, change the **IPayable[] payableObjects** array to **Object[] payableObjects.** Then in the **foreach** loop, cast the objects appropriate to an **IPayable** object to get the payment amount, and an **IContact** object to get the mailing address, and print these information out appropriately.

**EX5\_5:** Add fault-tolerance to your application by throwing exceptions at any possible error situation. Catch the exceptions that you throw, and either attempt to correct it and carry on with the application, or display a helpful error message and exit. **In particular, I would like you to define your own exception class, and use it.**

**EX5\_6:** Implement an **IComparable** Interface for the **BasePlusComissionEmployee**, **CommissionEmployee** and **SalaryEmployee**, such that these objects are always compared by their **Earnings**. Test this by sorting an array of employees using **Array.Sort().**

Finally, implement a class called **EmployeeContainer** with an **IEnumerable** interface. This class contains an array of different employee types (i.e. **BasePlusCommissionEmployee**, **CommissionEmployee**, and **SalaryEmployee**). Demonstrate that it works correctly by using a **foreach** loop to iterate over a collection of different employee types stored in the object.