GDAPS1 – Practice Exercise

Interfaces

# Objective

Practice implementing the interfaces given below.

# Details

Games usually require a lot of mathematical and geometric constructs, like points, circles, rectangles, etc. Certain functionality may require subsets of these constructs. For example: Circles and rectangles have an area, while points don’t.

We can use interfaces in C# to define the publicly accessible behaviors of these subsets. Each class will then implement the required public members, ensuring specific classes meet our needs.

Since interfaces also act as types, this allows our code elsewhere to work with objects of a *specific interface*, rather than just objects of a *specific class*. This is similar in concept to polymorphism, in that an object also counts as any of its interface types. However, a class can implement any number of interfaces.

## IPosition and IArea Interfaces

Create the following two interfaces in C#. You may copy/paste from this document. Remember to put each in its own file.

interface IPosition

{

// Properties

double X { get; set; } // X–axis coordinate

double Y { get; set; } // Y-axis coordinate

// Methods

// Distance to this coordinate from another coordinate

double DistanceTo(IPosition position);

// Moves the object to a new coordinate

// Example: MoveTo(500, 200); places the object at coordinate (500, 200)

void MoveTo(double x, double y);

// Increases or decreases the X and/or Y coordinate

// Example: MoveBy(-5, 6); would move 5 units negatively on X-axis

// and 6 units positively on Y-axis.

void MoveBy(double xOffset, double yOffset);

}

interface IArea

{

// Properties

double Area { get; }

double Perimeter { get; }

// Methods

// Is a coordinate within the area of this object?

bool ContainsPosition(IPosition position);

// Is this area larger than the area of another object?

bool IsLargerThan(IArea areaToCheck);

}

## Point and Circle Classes

Create two classes – one that represents a single point in space and another that represents a circle.

The Point class must implement the IPosition interface, and the Circle class must implement both IPosition and IArea. See below. Remember: separate files for each of these classes.

class Point : IPosition

class Circle : IPosition, IArea

To correctly implement the interfaces, you’ll need to include all required interface members (properties and methods). You will also need to add any fields and constructors that the Circle and Point classes will need.

You should also override ToString() to provide an easily printable version of the objects. Yes, there will probably be some duplication of code here. That’s ok for this exercise.

Even though Points and Circles might have some of the same members, *resist the urge to have Circle inherit from Point.* Polymorphism would cause a Circle to count as a Point, which we don’t want in this scenario. You will find, however, that Points and Circles both count as IPosition objects, and the compiler can be sure that they both have any public members defined in IPosition.

## Main Method

Test out your implementations by creating **two Points** and **two Circles**, each with different data. Do the following and report the results of each to the user:

* Print the initial data for all four objects
* Move point 2 and circle 2 objects using MoveTo() and/or MoveBy(), and print again
* Check the distance between both points
* Check the distance between each circle and each point
* Determine which circle has a larger area
* Determine if either circle contains either point

Notice that even though Circle and Point are distinct classes, they can both be used in certain contexts (since they both implement IPosition).

# Sample Run

**Please use these coordinates and radii for testing.** Decimal places may vary in your output.

Point: x 5 y 7

Point & Circle Info

Point 1: (5, 7)

Point 2: (10, 10)

Circle 1: (10, 10) with radius 3

Circle 2: (0, 0) with radius 5

Point: x 10 y 10

Circle: center x 10 center y 10, radius 3, area 28.27, perimeter 18.85

Circle: center x 0 center y 0, radius 5, area 78.54, perimeter 31.42

Moving point 2 to (2, 2)

Moving circle 2 by (-1, -1)

Point: x 5 y 7

Point: x 2 y 2

Circle: center x 10 center y 10, radius 3, area 28.27, perimeter 18.85

Circle: center x -1 center y -1, radius 5, area 78.54, perimeter 31.42

Distance between point 1 and point 2: 5.83095

Distance between point 1 and circle 1: 5.83095

Distance between point 1 and circle 2: 10

Distance between point 2 and circle 1: 11.3137

Distance between point 2 and circle 2: 4.24264

Circle 2’s area (78.54) is larger than circle 1’s area (28.27)

Does circle 1 contain point 1? No

Does circle 1 contain point 2? No

Does circle 2 contain point 1? No

Does circle 2 contain point 2? Yes

# Submission

All of your work must be commented and follow this course’s coding standards. **Read through the Coding Standards document (located in MyCourses) to check over your code before you complete your program. Make sure you follow the coding standards for all code you create.**

1) Submit: Submit your program to the appropriate Assignments dropbox in MyCourses.

2) Check-off: Show your working program to the instructor or TA. If you do not finish before class ends, complete the exercise for homework and show one of us in-class on the next class period. If your program works as expected, you will be “checked off” to earn credit for the exercise.