Universidad del Istmo de Guatemala
Facultad de Ingenieria
Ing. en Sistemas
Parcial #2
Prof. Ernesto Rodriguez - erodriguez@unis.edu.gt

Parcial #2

Fecha de entrega: 15 de Febrero, 2018 - 11:59pm

Instrucciones: Crear una solución llamada "Parcial2". Adentro de la solución crear dos projectos: "Lineas" y "LineasTests". A continuación se presentara una serie de ejercicios. Debe resolver dichos ejercicios y colocar su codigo en la ubicación correcta dentro de la solución. La entrega debe realizarse a traves de blackboard. La solución entera del parcial debe colocarse en un archivo zip.

Ejercicio #1 (10%)

Definir una interfaz llamada generica "ILinea" en el proyecto "Lineas". Esta interfaz debe definir los siguientes metodos:

1. "Longitud" de tipo `Longitud : void` $\rightarrow$ `double`
2. "PuntoMasCercano" de tipo `PuntoMasCercano : T` $\rightarrow$ `T`

Ejercicio #2 (18%)

Definir una clase generica llamada "LineaAbstracta" en el proyecto "Lineas". Esta clase debe implementar la interfaz "ILiena" de tal forma que sus metodos y propiedades queden asi:

1. Debe definir un metodo abstracto publico llamado "Distancia" de tipo `Distancia : T` $\otimes$ `T` $\rightarrow$ `double`
2. Debe definir una propiedad publica de solo lectura llamada "Puntos" de tipo `Puntos : T[ ]`
3. El metodo longitud se debe implementar sumando todas las distancias que hay entre los puntos de la linea mediante el metodo "Distancia"
4. El metodo "PuntoMasCercano" debe implementarse de tal forma que utiliza el metodo "Distancia" para encontrar cual es el punto más cercano de esta linea al punto que se le paso como parametro.

Ejercicio #3 (18%)

Utilizar la clase "Punto2D" que se ha adjuntado a este parcial para definir una clase llamada "Linea2D" en el proyecto "Lineas" la cual hereda de "LineaAbstracta" instanciando su parametro generico a "Punto2D". Esta clase debe implementar el metodo "Distancia" utilizando la distancia euclideana entre dos puntos. Como recordatorio, la distancia euclideana esta definida asi:

$$d_{euc}(\langle x_1, y_1\rangle, \langle x_2, y_2\rangle) := \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Esta clase tambien debe implementar la propiedad “Puntos”, inicializandola en el constructor mediante un arreglo que se le debe pasar como parametro.

Ejercicio #4 (18%)

Utilizar la clase “Punto3D” que se ha adjuntado a este parcial para definir una clase llamada “Linea3D” en el proyecto “Lineas”, la cula hereda de “LineaAbstracta” instanciando su parametro generico a “Punto3D”. Esta clase debe implementar el metodo “Distancia” utilizando la distancia euclideana entre dos puntos. Como recordatorio, la distancia euclideana en tres dimensiones se define asi:

$$d_{euc}(\langle x_1, y_1, z_1\rangle, \langle x_2, y_2, z_2\rangle) := \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$$

Esta clase tambien debe implementar la propiedad “Puntos”, inicializandola en el constructor mediante un arreglo que se le debe pasar como parametro.

Ejercicio #5 (18%)

En el proyecto “LineasTests”, crear una clase llamada “Linea2DTests”. Esta clase debe implementar un test para el metodo distancia y un test para el metodo punto más cercano. Puede utilizar la plantilla “TestPlantilla” adjunta para escribir dicho test.

Ejercicio #6 (18%)

En el proyecto “LineasTests”, crear una clase llamada “Linea3DTests”. Esta clase debe implementar un test para el metodo distancia y un test para el metodo punto más cercano. Puede utilizar la plantilla “TestPlantilla” adjunta para escribir dicho test.

Extra (10%)

En C# es posible asignar un valor a una variable, siempre y cuando el tipo del valor sea descendiente del tipo de la variable. A esto se le llama *polimorfismo*. El siguiente ejemplo muestra este comportamiento con las clases de este parcial.

```
public class Poli{
    public static void main(string[] args){
        LineaAbstracta linea = new Linea2D(new Punto2D[]{});
        linea = new Linea3D(new Punto3D[]{});
    }
}
```

Como puede ver, a la variable linea se le puede asignar tanto una “Linea2D” o “Linea3D”. Sin embargo, cuando la variable en cuestion es un arreglo, el polimorfismo puede conllevar a problemas cuando se utiliza como en el siguiente ejemplo:

```
public class PoliFail{
    public static void main(string[] args){
        string[] strs = new string[]{};
        object[] objs = strs;
    }
}
```

Escriba un ejemplo de C# donde utilizar el polimorfismo para arreglos como se muestra anteriormetne puede llevar a que el programa falle durante la ejecución, a pesar que los tipos estarian correctos.