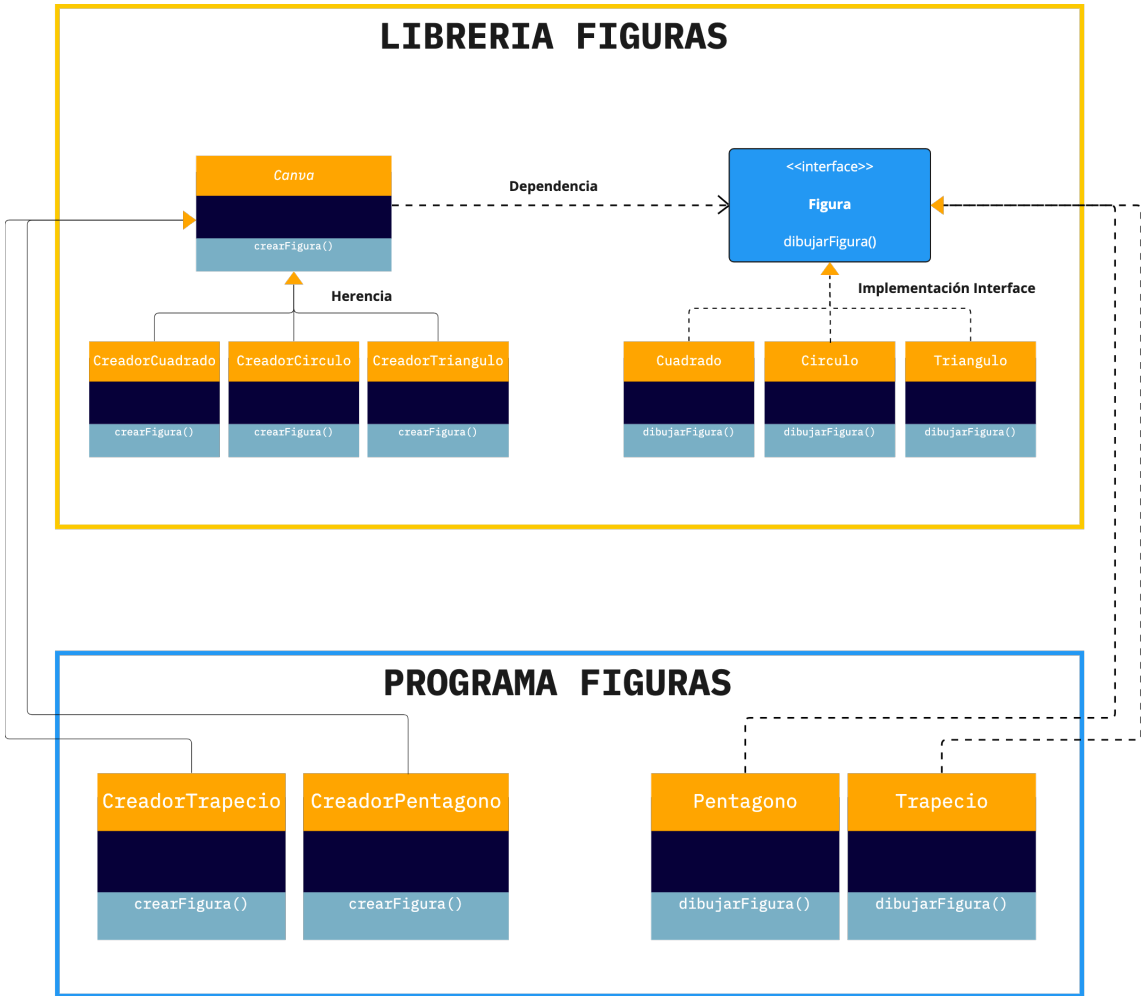
INSTITUTO SUPERIOR TECNOLÓGICO QUITO
Excelencia en Educación Superior

DESARROLLO DE SOFTWARE
LENGUAJE DE PROGRAMACIÓN 4
PAÚL ALVAREZ CORRAL

Objetivo: Caso práctico acerca del dibujo de figuras geométricas.

Link GitHub: https://github.com/PaulAlvarezC/LibreryFactoryMethod.git

Diagrama:



Se tiene una lista de figuras geométricas descritas en un archivo JSON.

```json
[
  {
    "Figura": {
      "nombre": "Cuadrado",
      "tipo": "cuadrado",
      "cadena": "width=5;height=5;"
    }
  },
  {
    "Figura": {
      "nombre": "Circulo",
      "tipo": "circulo",
      "cadena": "radio=3;"
    }
  },
  {
    "Figura": {
      "nombre": "Triangulo",
      "tipo": "triangulo",
      "cadena": "a=3;b=5;c=2;"
    }
  },
  {
    "Figura": {
      "nombre": "Trapecio",
      "tipo": "trapecio",
      "cadena": "a=3;b=5;c=7;d=6;h=4;"
    }
  }
]
```

El programa debe leer el archivo y tomar la información de cada figura, para ser dibujada en el Canva.

```
private static string _path = @"C:\data.json";
//private static string _path = @"/Volumes/iGatoA

public static void Main(string[] args)
{
    //LEER JSON
    Console.WriteLine("LEER JSON -> FIGURAS");
    var figuras = GetFiguresJsonFromFile();
    Console.WriteLine(figuras);
```

```
public static string GetFiguresJsonFromFile()
{
    string figurasFromFile;
    using (var render = new StreamReader(_path))
    {
        figurasFromFile = render.ReadToEnd();
    }
    return figurasFromFile;
}
```

Se debe cumplir las siguientes condiciones:

1. La suma del área de todas las figuras en el archivo JSON debe ser menor al área disponible del canva.
2. Cada figura debe tener la capacidad de contener el texto que lleva asociado.

```
var fig = JsonConvert.DeserializeObject<List<Modelo>>(figuras);

if (
{
    class Newtonsoft.Json.JsonConvert
    Provides methods for converting between .NET types and JSON types.
    for (int i = 0; i < fig.Count; i++)
    {
        Console.WriteLine("Figura: " + i);
        if (fig[i].Figura.Nombre == "Cuadrado") {
            listaFiguras.Add(creatorA.crearFigura(fig[i].Figura.Nombre, fig[i].Figura.Tipo, fig[i].Figura.Cadena));
        } else if (fig[i].Figura.Nombre == "Circulo") {
            listaFiguras.Add(creatorB.crearFigura(fig[i].Figura.Nombre, fig[i].Figura.Tipo, fig[i].Figura.Cadena));
        } else if (fig[i].Figura.Nombre == "Triangulo"){
            listaFiguras.Add(creatorC.crearFigura(fig[i].Figura.Nombre, fig[i].Figura.Tipo, fig[i].Figura.Cadena));
        } else if (fig[i].Figura.Nombre == "Trapecio"){
            listaFiguras.Add(creatorD.crearFigura(fig[i].Figura.Nombre, fig[i].Figura.Tipo, fig[i].Figura.Cadena));
        }
    }
}

double acumuladorAreas = 0;
double canvaArea = 70;
```

Cada figura deberá calcular su área y perímetro de acuerdo a las características definidas en la cadena del archivo JSON.

```
"Figura": {
  "nombre": "Cuadrado",
  "tipo": "cuadrado",
  "cadena": "width=5;height=5;"
}
```

```
"Figura": {
  "nombre": "Circulo",
  "tipo": "circulo",
  "cadena": "radio=3;"
}
```

```
"Figura": {
  "nombre": "Triangulo",
  "tipo": "triangulo",
  "cadena": "a=3;b=5;c=2;"
}
```

Cada figura posee una formula distinta para calcular el área y el perímetro.

```
public string Texto { get; set; }
public string Tipo { get; set; }
public string Cadena { get; set; }

public void dibujarFigura()
{
    Console.WriteLine("Producto figura Cuadrado");
    Console.WriteLine("Texto  : " + Texto);
    Console.WriteLine("Tipo   : " + Tipo);
    Console.WriteLine("Cadena : " + Cadena);
}

public double calcularArea()
{
    var arreglo = Cadena.Split(";");
    var arr1 = arreglo[0];
    var number1 = (from t in arr1 where char.IsDigit(t) select t).ToArray();
    //Console.WriteLine(number1);
    var arr2 = arreglo[1];
    var number2 = (from t in arr2 where char.IsDigit(t) select t).ToArray();
    //Console.WriteLine(number2);

    var lado1 = Int32.Parse(number1);
    var lado2 = Int32.Parse(number2);
    var area = lado1 * lado2;
    return area;
}

public double calcularPerimetro()
{
    var arreglo = Cadena.Split(";");
    var arr1 = arreglo[0];
    var number1 = (from t in arr1 where char.IsDigit(t) select t).ToArray();
    //Console.WriteLine(number1);

    var lado = Int32.Parse(number1);
    var perimetro = lado * 4;
    return perimetro;
}
```

Se debe realizar el programa utilizando una librería que tenga la lógica correcta para dibujar tanto un cuadrado, un círculo y un triángulo.

Y si aumento una figura geométrica mas no debe fallar el funcionamiento.