Desarrollo de Aplicaciones WEB

Prueba Objetiva **Segundo Parcial**

1a.- Partiendo de las clases *JuegoAbstracto* y *Pantalla* (cuyo código se adjunta), desarrolla las siguientes clases:

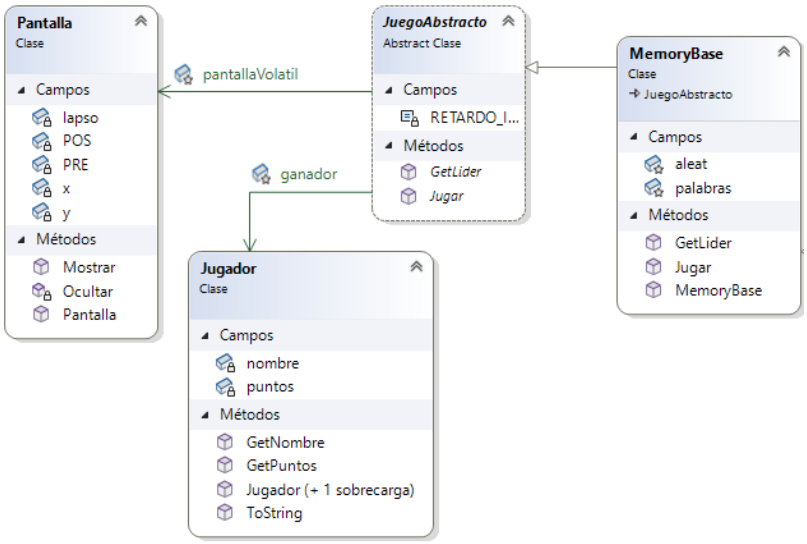
- Clase **Jugador**.

Dos constructores. + Uno recibe el nombre del jugador y los puntos que tiene. + Otro solo el nombre.	Si la puntuación es negativa, el nombre null, o es un string vacío, …. lanza Exception.
+ String **GetNombre**() + int **GetPuntos**()	Retornan el valor de los atributos.
+ String **ToString**()	Retorna la forma textual del jugador. Formato: <nombre> con <puntos> puntos Si tiene 0 puntos, solo se muestra el nombre: <nombre>

- Clase **MemoryBase** heredando de la clase abstracta. Deberá ser completamente funcional y sin errores de compilación ni de ejecución.

(Se adjunta un programa de prueba para ayudarte a que confirmes su funcionamiento mínimo correcto)

Constructor. + Recibe una array de String con las palabras para el juego.	Almacena en el objeto las palabras que recibe.
+ Jugador **GetLider**()	Retorna el objeto *ganador* heredado. Si es null lanza Exception explicando que aún no hay lider.
+ Boolean **Jugar**()	Elige una palabra al azar y la muestra temporalmente (usando la *pantallaVolatil* heredada). Pide por consola al usuario que introduzca la palabra que se mostró. Retorna si la acierta, o no.



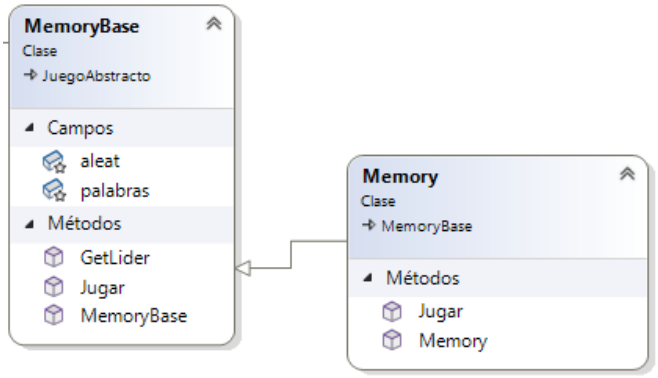
El atributo ***palabras*** de la clase ***MemoryBase*** que guarda las palabras…

obligatoriamente, debe ser ***privado***.

1b.- Partiendo de lo anterior, implementa:

- Clase **Memory** heredando de la clase *MemoryBase*

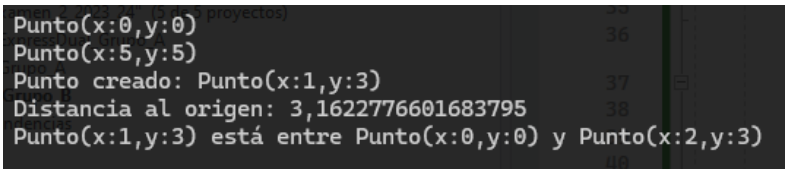
Constructor. + Recibe una array de String con las palabras para el juego.	Almacena en el atributo *palabras* que heredó, las palabras que recibe.
+ Boolean **Jugar**()	Hace uso de la función a la que sobrecarga para poner en marcha el juego. Mientras el usuario acierta el juego, vuelve a jugar. Se debe calcular las veces que el usuario acierta el juego. Cuando falle, si no hay ganador, o el usuario ha acertado más veces que el jugador que es hasta ahora el ganador, se le pregunta el nombre para registrar el nuevo objeto Jugador con la puntuación (número de aciertos) obtenida. Retorna si la se ha superado el récord (sustituyendo al ganador), o no.



2.- Modifica la implementación de la clase Punto que se facilita para incluir en ella los siguientes métodos:

+ `CrearPunto(Punto a, Punto b)`	Función **estática** que retorna un nuevo objeto Punto colocado al azar entre los punto a y b que se le pasan.
+ `Distancia(Punto p)`	Retornan un double con la distancia entre el propio punto y p.
+ `EstaDentro(Punto a, Punto b)`	Retornan Boolean indicando si el propio punto está entre los puntos a y b que se le pasan.

Este sería un pantallazo de una ejecución del programa que se adjunta y que usa dichos métodos (por si lo quieres probar):



Desarrollo de Aplicaciones WEB

Prueba Objetiva **Segundo Parcial**

```
public class Pantalla
{
    private int x, y;
    private int lapso;
    private String PRE = "-----> ";
    private String POS = " <-----";

    public Pantalla(int x, int y, int lapso)
    {
        this.x = x;
        this.y = y;
        this.lapso = lapso;
    }
    public void Mostrar(Object obj)
    {
        Console.Clear();
        Console.SetCursorPosition(x, y);
        Console.Write(PRE+obj+POS);
        Thread.Sleep(lapso);
        lapso = lapso*2/3;
        Ocultar(obj);
    }
    private void Ocultar(Object obj)
    {
        int caracteres = (PRE+obj+POS).Length;
        Console.SetCursorPosition(x, y);
        for (int i = 0; i < caracteres; i++)
            Console.Write(' ');
    }
}
```

```
public abstract class JuegoAbstracto {

    private const int RETARDO_INICIAL = 1000;
    protected Jugador? ganador;
    protected Pantalla pantallaVolatil =
                new Pantalla(5,5,RETARDO_INICIAL);

    public abstract Boolean Jugar();
    public abstract Jugador GetLider();
}
```

```
public class Punto {
    private int x, y;
    private static Random alea = new Random();
    public Punto(int x, int y) {
        this.x = x;
        this.y = y;
    }
    public int GetX() { return x; }
    public int GetY() { return y; }
    public override string ToString() {
        return $"Punto(x:{x},y:{y})";
    }
}
```

```
private static void PruebaPregunta1a()
{
  String[] palabras =
    { "caja", "percha", "hacha", "sachillo", "guarapo" };
  JuegoAbstracto j = new MemoryBase(palabras);
  if (j.Jugar())
    Console.WriteLine("Enhorabuena. ¡ACERTASTE!");
  else
    Console.WriteLine("Respuesta errónea. ¡Repita y Suerte!");
}

private static void PruebaPregunta1b()
{
  char opcion;
  Boolean recordSuperado;
  String[] palabras = { "rojo", "blanco", "verde", "azul" };
  JuegoAbstracto j = new Memory(palabras);
  do
  {
    recordSuperado = j.Jugar();
    if (recordSuperado)
      Console.WriteLine(
                    "\n  >>>  Nuevo Lider: " +
                    j.GetLider() + "  <<<\n");
    Console.WriteLine("Quiere volver a jugar (S/n)?");
    opcion = Char.ToLower(Console.ReadKey().KeyChar);
  } while (opcion != 'n');
  Console.WriteLine(
                    "\n\nJuego Terminado\n\nGANADOR: >>> " +
                    j.GetLider() + " <<<\n");
}
```

```
private static void PruebaPregunta2()
{
  Punto p00 = new Punto(0, 0);
  Punto p55 = new Punto(5, 5);
  Punto p23 = new Punto(2, 3);
  Console.WriteLine(p00);
  Console.WriteLine(p55);

  //CrearPunto
  Punto p = Punto.CrearPunto(p00, p55);
  Console.WriteLine($"Punto creado: {p}");

  //Distancia
  Console.WriteLine("Distancia al origen: "+
                    p.Distancia(new Punto(0, 0)));

  //EstaDentro
  if (p.EstaDentro(p00, p23))
    Console.WriteLine($"{p} está entre {p00} y {p23}");
  else
    Console.WriteLine($"{p} NO está entre {p00} y {p23}");
}
```