

DOCUMENTACIÓN TÉCNICA DEL PROYECTO QUE CALCÚLA EL ÁREA DE UNA FIGURA GEOMÉTRICA A ELECCIÓN DEL USUARIO

- Se planteó la idea de realizar un programa que pueda calcular el área de un círculo, triángulo y de un rectángulo. Primero debemos analizar lo que se los está pidiendo para lograr encontrar los entes sobre los que vamos a trabajar, en este caso en particular nuestros objetos de trabajo son las figuras geométricas mencionadas. Ya que conocemos sobre quién vamos a trabajar necesitamos definir nuestros datos de entrada y saber que información daremos de salida posterior al proceso que realizará internamente el programa.
 - Datos necesarios para calcular el área del círculo:
 - Radio
 - Datos necesarios para calcular el área del triángulo:
 - Base
 - Altura
 - Datos necesarios para calcular el área del rectángulo:
 - Largo
 - Ancho
 - Información de salida requerida:
 - El área del (Figura Geométrica) es: (Área Calculada)

- Ya que se analizaron los datos requeridos por el programa y la información que se requiere como salida, vamos a diseñar las clases modelo que se utilizarán en este programa en el entendido de que una clase almacena la información que el programa necesita para poder funcionar.
 - Diseño de la clase modelo CL_CIRCULO que tiene como atributo privado “r” que representa al radio, tiene un método **constructor**, tiene sus métodos **set** y **get** y tiene un método funcional llamado **calcular_area_cir**, el cuál nos dice que sabe calcular el área de un círculo, más no nos dice cómo lo sabe hacer.

CL_CIRCULO

-r:REAL

```
CL_CIRCULO(radius:REAL)
set_r(x:REAL)
get_r:REAL
calcular_area_cir
```

- A continuación, se observa el pseudocódigo de la clase CL_CIRCULO, el cuál nos responde el ¿Cómo? sabe hacer lo que con el diseño de la clase se definió que sabía hacer.

```
CLASE CL_CIRCULO DERIVADA DE FIGURA_GEOMETRICA
/*
NOMBRE:RODRIGO DÍAZ SALGUERO
FECHA: 19/11/2024
*/
INICIO
SECCIÓN DE ATRIBUTOS
    r:REAL, PRIVADO
SECCIÓN DE MÉTODOS

    MÉTODO CL_CIRCULO(radius:REAL)
    INICIO
        r ← radius
    FIN MÉTODO CL_CIRCULO

    MÉTODO set_r(x:REAL)
    INICIO
        r←x
    FIN MÉTODO set_r

    MÉTODO get_r:REAL
    INICIO
        REGRESAR r
    FIN MÉTODO get_r

    MÉTODO calcular_area_cir
    INICIO
        area ←  $\pi * r^2$ 
    FIN MÉTODO calcular_area_cir

FIN CLASE CL_CIRCULO
```

- Diseño de la clase CL_TRIANGULO que tiene como atributos privados **base** y **altura**, tiene un método **constructor**, los métodos **setters** y **getters**, así como su método funcional **calcular_area_tri**, el cuál nos dice que sabe calcular el área de un triángulo.

CL_TRIANGULO

-base:REAL
-altura:REAL

CL_TRIANGULO(base:REAL, altura:REAL)
set_base(x:REAL)
set_altura(x:REAL)
get_base:REAL
get_altura:REAL
calcular_area_tri

- Pseudocódigo de la clase CL_TRIANGULO, el cual nos responde el ¿Cómo? es que sabe calcular el área de un triángulo por medio del método ***calcular_area_tri***.

```
CLASE CL_TRIANGULO DERIVADA DE FIGURA_GEOMETRICA
/*
NOMBRE:RODRIGO DÍAZ SALGUERO
FECHA: 19/11/2024
*/
INICIO
SECCIÓN DE ATRIBUTOS
    base:REAL, PRIVADO
    altura:REAL, PRIVADO
SECCIÓN DE MÉTODOS

    MÉTODO CL_TRIANGULO(base:REAL, altura:REAL)
    INICIO
        base ← base
        altura ← altura
    FIN MÉTODO CL_TRIANGULO

    MÉTODO set_base(x:REAL)
    INICIO
        base ← x
    FIN MÉTODO set_base

    MÉTODO set_altura(x:REAL)
    INICIO
        altura ← x
    FIN MÉTODO set_altura

    MÉTODO get_base:REAL
    INICIO
        REGRESAR base
    FIN MÉTODO get_base

    MÉTODO get_altura:REAL
    INICIO
        REGRESAR altura
    FIN MÉTODO get_altura

    MÉTODO calcular_area_tri
    INICIO
        area ← (base*altura)/2
    FIN MÉTODO calcular_area_tri

FIN CLASE CL_TRIANGULO
```

- Diseño de la clase CL_RETANGULO, la cual tiene como atributos privados ***largo*** y ***ancho*** de tipo real, tiene un método ***constructor***, tiene los métodos ***set*** y ***get*** de cada atributo, así como el método funcional ***calcular_area_rec***, el cual nos dice que sabe calcular el área del rectángulo, sin embargo, aún no sabemos cómo lo sabe hacer.

CL_RECTANGULO
-largo:REAL -ancho:REAL
CL_RECTANGULO(largo:REAL,ancho:REAL) set_largo(x:REAL) set_ancho(x:REAL) get_largo:REAL get_ancho:REAL calcular_area_rec

- El pseudocódigo de esta clase CL_RECTANGULO nos responde el ¿Cómo? sabe hacer lo que indican sus métodos.

```
CLASE CL_RECTANGULO DERIVADA DE FIGURA_GEOMETRICA
/*
NOMBRE:RODRIGO DÍAZ SALGUERO
FECHA: 19/11/2024
*/
INICIO
SECCIÓN DE ATRIBUTOS
    largo:REAL, PRIVADO
    ancho:REAL, PRIVADO
SECCIÓN DE MÉTODOS

    MÉTODO CL_RECTANGULO(largo:REAL,ancho:REAL)
    INICIO
        largo←largo
        ancho←ancho
    FIN MÉTODO CL_RECTANGULO

    MÉTODO set_largo(x:REAL)
    INICIO
        largo←x
    FIN MÉTODO set_largo

    MÉTODO set_ancho(x:REAL)
    INICIO
        ancho←x
    FIN MÉTODO set_ancho

    MÉTODO get_largo:REAL
    INICIO
        REGRESAR largo
    FIN MÉTODO get_largo

    MÉTODO get_ancho:REAL
    INICIO
        REGRESAR ancho
    FIN MÉTODO get_ancho

    MÉTODO calcular_area_rec
    INICIO
        area ← (largo * ancho)
    FIN MÉTODO calcular_area_rec

FIN CLASE CL_RECTANGULO
```

- A continuación vamos a poder observar el diseño y pseudocódigo de una *Super Clase*, observamos anteriormente en los pseudocódigos de las clases CL_CIRCULO, CL_TRIANGULO y CL_RECTANGULO que en el encabezado del documento posterior al nombre de la clase se escribió “DERIVADA DE FIGURA_GEOMETRICA”, esta instrucción indica *Herencia*, quiere decir que esas clases heredan de la clase FIGURA_GEOMETRICA, por lo tanto, las *Subclases* pueden hacer uso de los métodos y atributos de la clase FIGURA_GEOMETRICA, siempre y cuando tengan un correcto modificador de acceso.
 - Diseño de la Super Clase FIGURA_GEOMETRICA, la cuál tiene como atributo protegido **area**, dicho atributo es el que se usa para guardar el resultado del cálculo del área de cualquiera de esas figuras.

FIGURA_GOMETRICA

#area:REAL

set_area(x:REAL)

get_area:REAL

- Pseudocódigo de la Super Clase FIGURA_GEOMETRICA

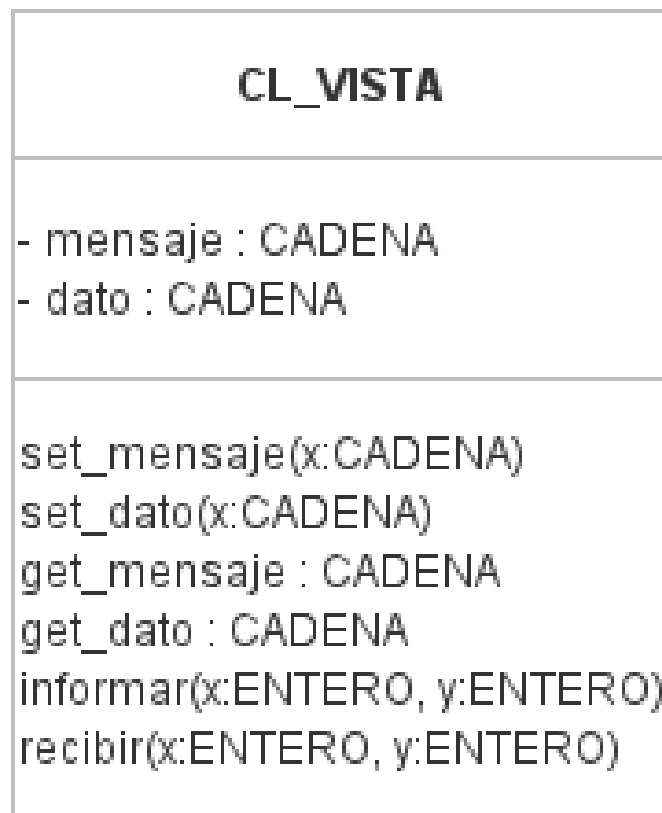
```
CLASE FIGURA_GEOMETRICA
/*
NOMBRE:RODRIGO DÍAZ SALGUERO
FECHA: 19/11/2024
*/
INICIO
SECCIÓN DE ATRIBUTOS
    area:REAL, PROTEGIDO
SECCIÓN DE MÉTODOS

    MÉTODO set_area(x:REAL)
    INICIO
        area ← x
    FIN MÉTODO set_area

    MÉTODO get_area:REAL
    INICIO
        REGRESAR area
    FIN MÉTODO get_area

FIN CLASE FIGURA_GEOMETRICA
```

- Ya que tenemos estas clases listas, vamos a diseñar una clase vista, vamos a diseñarla para que pueda trabajar con tecnología **Text User Interface** (TUI).
 - Diseño de la clase CL_VISTA, esta clase sigue contando con los atributos privados mensaje y dato, de igual forma los setters y getters, a diferencia de cuando se trabajaba con tecnología CLI, ahora los métodos informar y recibir van a recibir como parámetro dos números enteros representando la coordenada donde queremos informar o recibir.



- Pseudocódigo de la clase CL_VISTA

```
CLASE CL_VISTA
/*
Autor: Rodrigo Díaz Salguero
Fecha: 16/10/2024
Actualización: 19/11/2024
*/
INICIO
SECCIÓN DE VARIABLES
    mensaje : CADENA, PRIVADO
    dato : CADENA, PRIVADO
SECCIÓN DE MÉTODOS
    MÉTODO set_mensaje(x:CADENA)
        INICIO
            mensaje ← x
        FIN MÉTODO set_mensaje

    MÉTODO set_dato(x:CADENA)
        INICIO
            dato ← x
        FIN MÉTODO set_dato

    MÉTODO get_mensaje:CADENA
        INICIO
            REGRESAR mensaje
        FIN MÉTODO get_mensaje

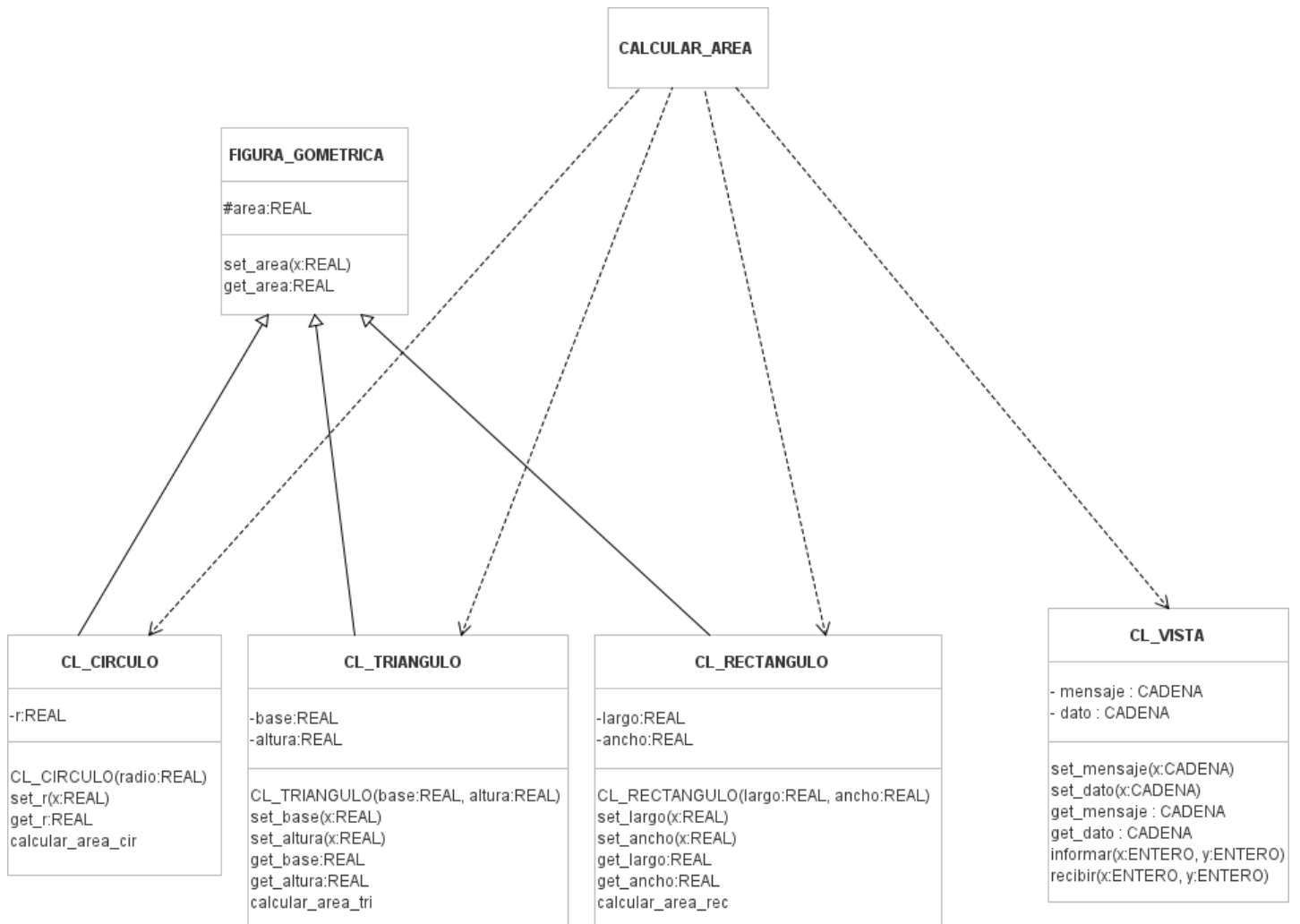
    MÉTODO get_dato:CADENA
        INICIO
            REGRESAR dato
        FIN MÉTODO get_dato

    MÉTODO informar(x:ENTERO, y:ENTERO)
        INICIO
            POSICIONAR(x,y)
            ESCRIBIR mensaje
        FIN MÉTODO informar

    MÉTODO recibir(x:ENTERO, y:ENTERO)
        INICIO
            POSICIONAR(x,y)
            LEER(dato)
        FIN MÉTODO recibir

FIN CLASE CL_VISTA
```

- Ya que tenemos correctamente bien diseñadas nuestras clases modelo y la clase vista podemos continuar con el proceso de desarrollo haciendo el diagrama de clases.



- La primer vista o interfaz que verá el usuario será el menú en que va a seleccionar si desea calcular el área del círculo, triángulo o cuadrado, de igual forma podrá seleccionar la opción salir.

```

*****
*
*
*          CALCULAR ÁREAS DE FIGURAS GEOMÉTRICAS
*
*
*****
*
*          MENÚ
*
*          1.- ÁREA DEL CIRCULO
*
*          2.- ÁREA DEL TRIÁNGULO
*
*          3.- ÁREA DEL RECTÁNGULO
*
*
*
*          INGRESE EL NÚMERO DE LA OPERACIÓN A REALIZAR:
*
*
*****

```

- Si se selecciona el área del círculo se podrá observar la siguiente interfaz

```
*****
*
*          CALCULAR ÁREAS DE FIGURAS GEOMÉTRICAS          *
*
*****
*
*                  CIRCULO                                *
*
*
*          INGRESE EL VALOR DEL RADIO:                    *
*
*
*
*          EL AREA DEL CIRCULO ES:                         *
*
*
*
*
*
*****
```

- Si se selecciona el área del triángulo se podrá observar la siguiente interfaz

```
*****
*
*          CALCULAR ÁREAS DE FIGURAS GEOMÉTRICAS          *
*
*****
*
*                  TRIÁNGULO                               *
*
*
*          INGRESE EL VALOR DE LA BASE:                   *
*
*          INGRESE EL VALOR DE LA ALTURA:                *
*
*
*          EL AREA DEL TRIÁNGULO ES:                      *
*
*
*
*
*
*****
```

- ```

*
*
* CALCULAR ÁREAS DE FIGURAS GEOMÉTRICAS *
*

*
*
* RECTÁNGULO *
*
*
* INGRESE EL VALOR DEL LARGO: *
*
* INGRESE EL VALOR DEL ANCHO: *
*
*
* EL AREA DEL RECTÁNGULO ES: *
*
*
*

```

- Ya que sabemos las posiciones dónde se ejecutará el método informar y el método recibir de la clase CL\_VISTA y teniendo listo nuestro diagrama de clases, podemos hacer el diagrama de secuencias, que nos representa la interacción entre los objetos que componen nuestro programa. Junto con la entrega de éste documento se adjunta el documento en pdf dónde podrá consultar el diagrama de secuencias correspondiente al proyecto ya que las imágenes que genera VIOLET al ser tan grande el diagrama de secuencias, no se logra apreciar absolutamente nada.

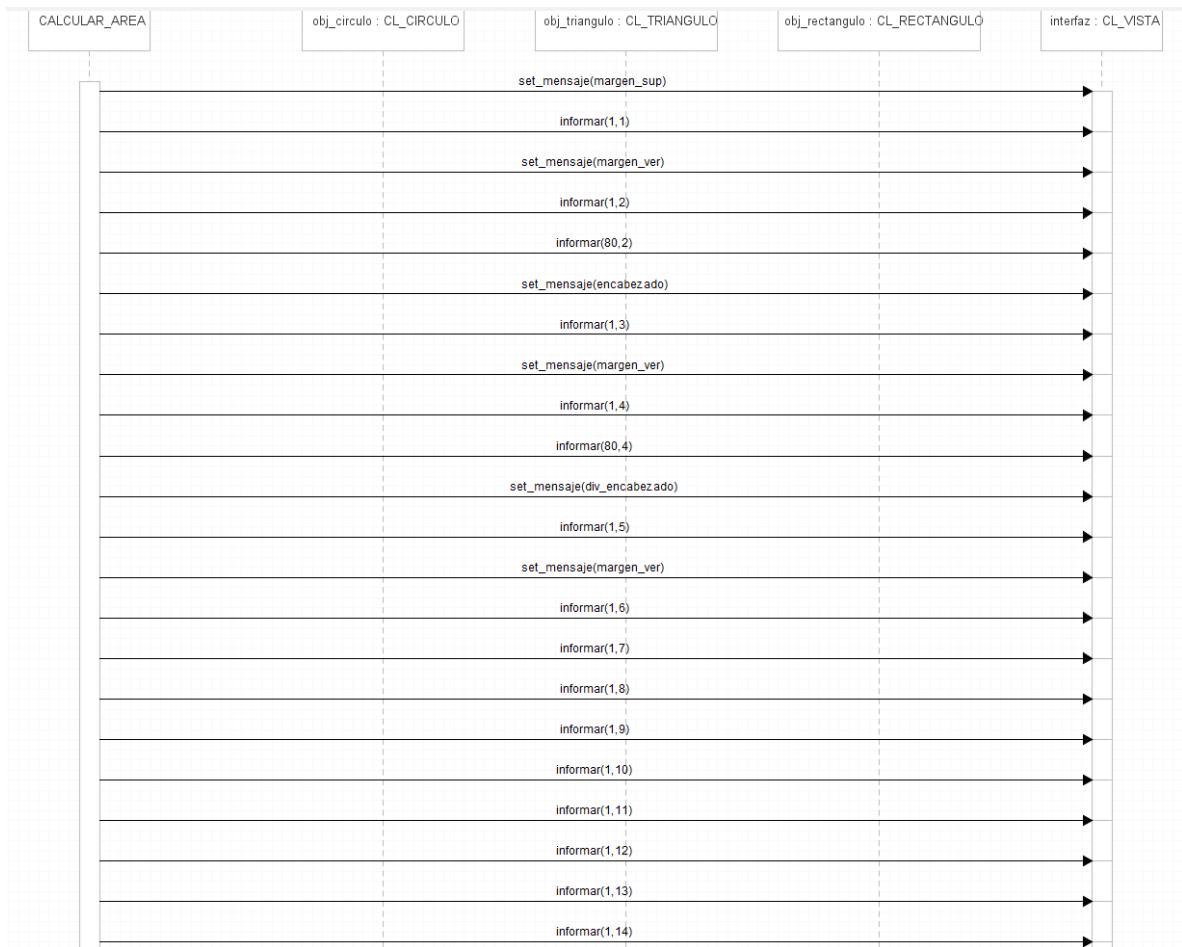
De igual manera, aquí se encuentra un acceso directo para abrir el documento del diagrama.



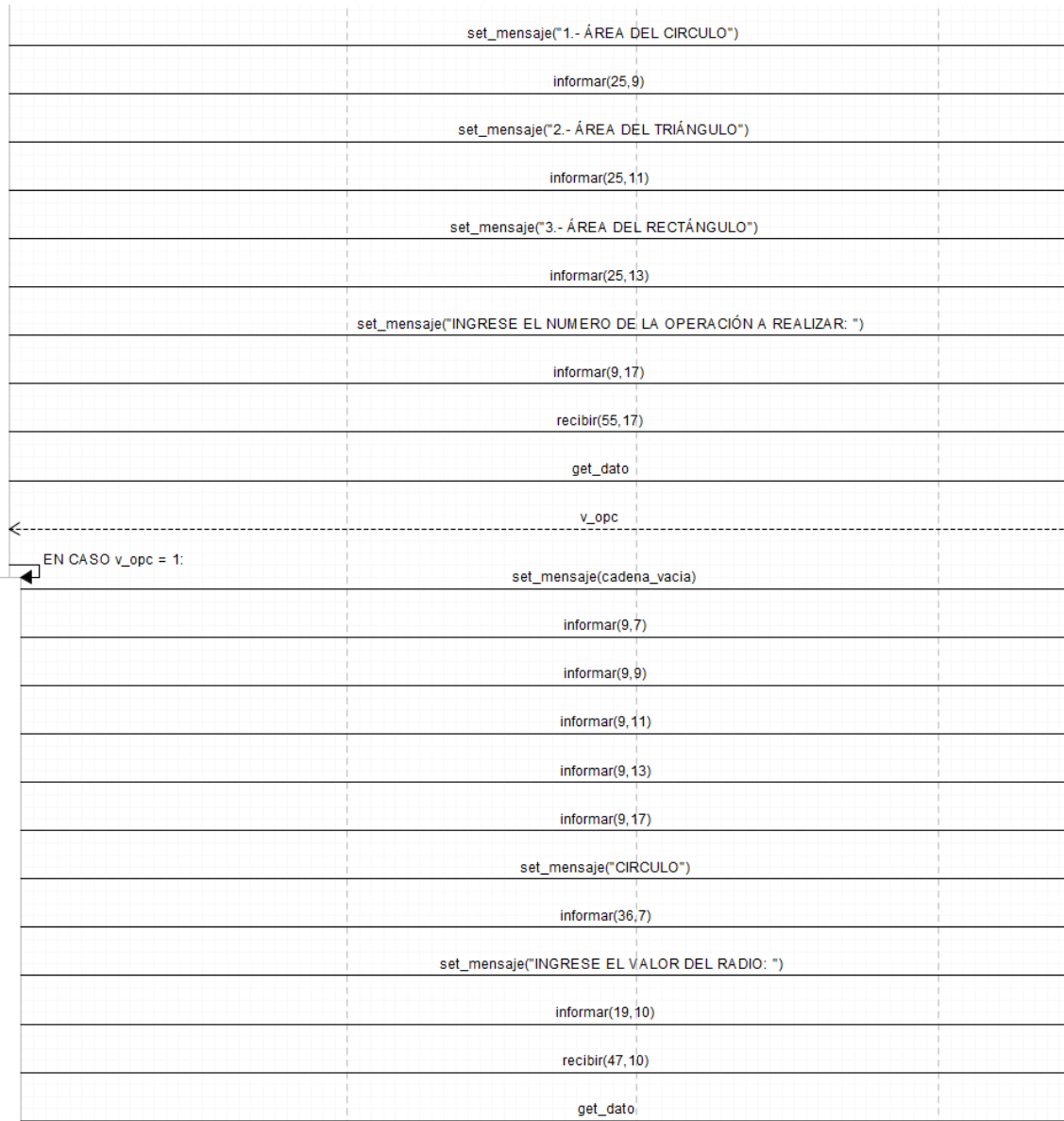
DIAGRAMA\_SECUENCIAS.pdf

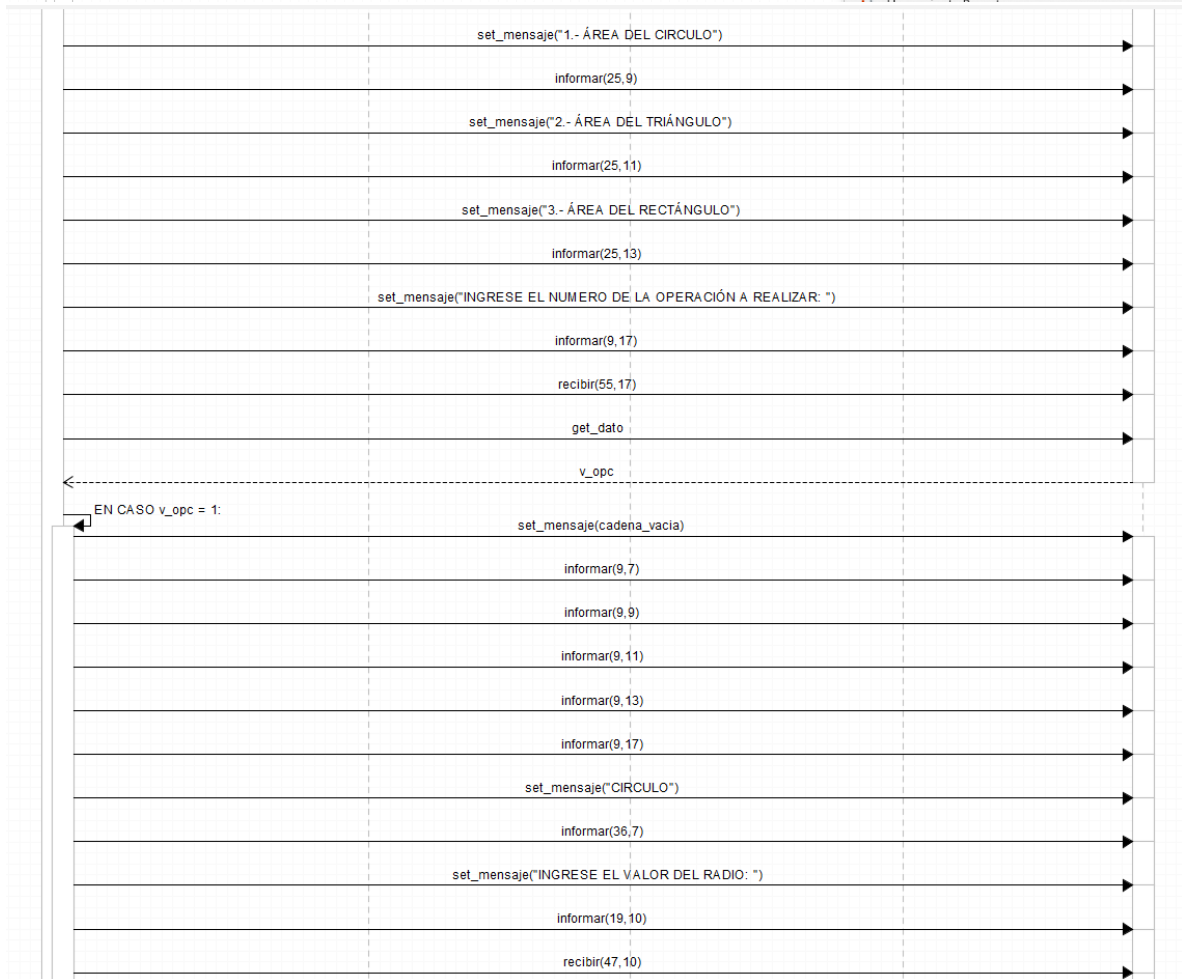
Capturas de pantalla del diagrama de secuencias:

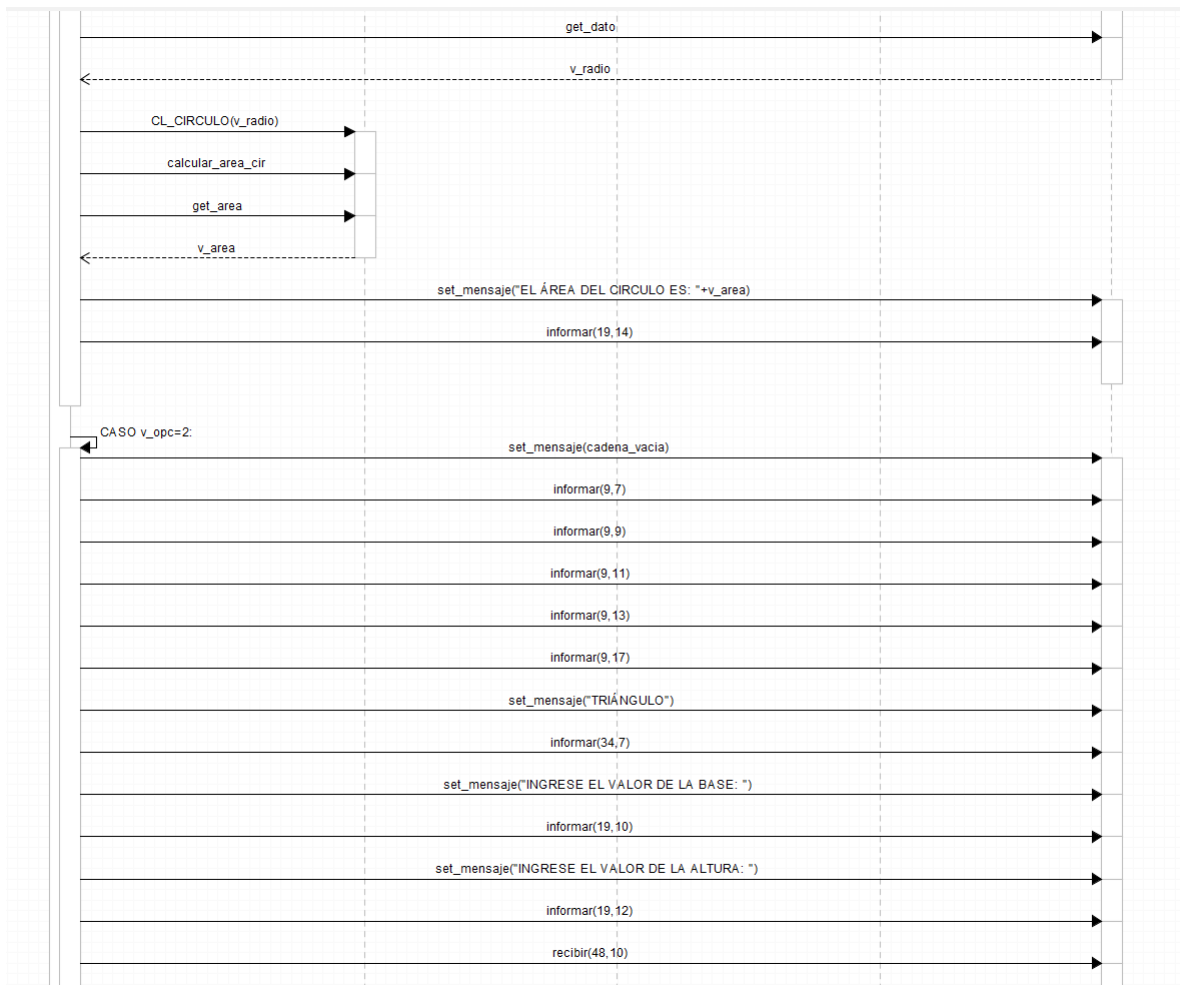


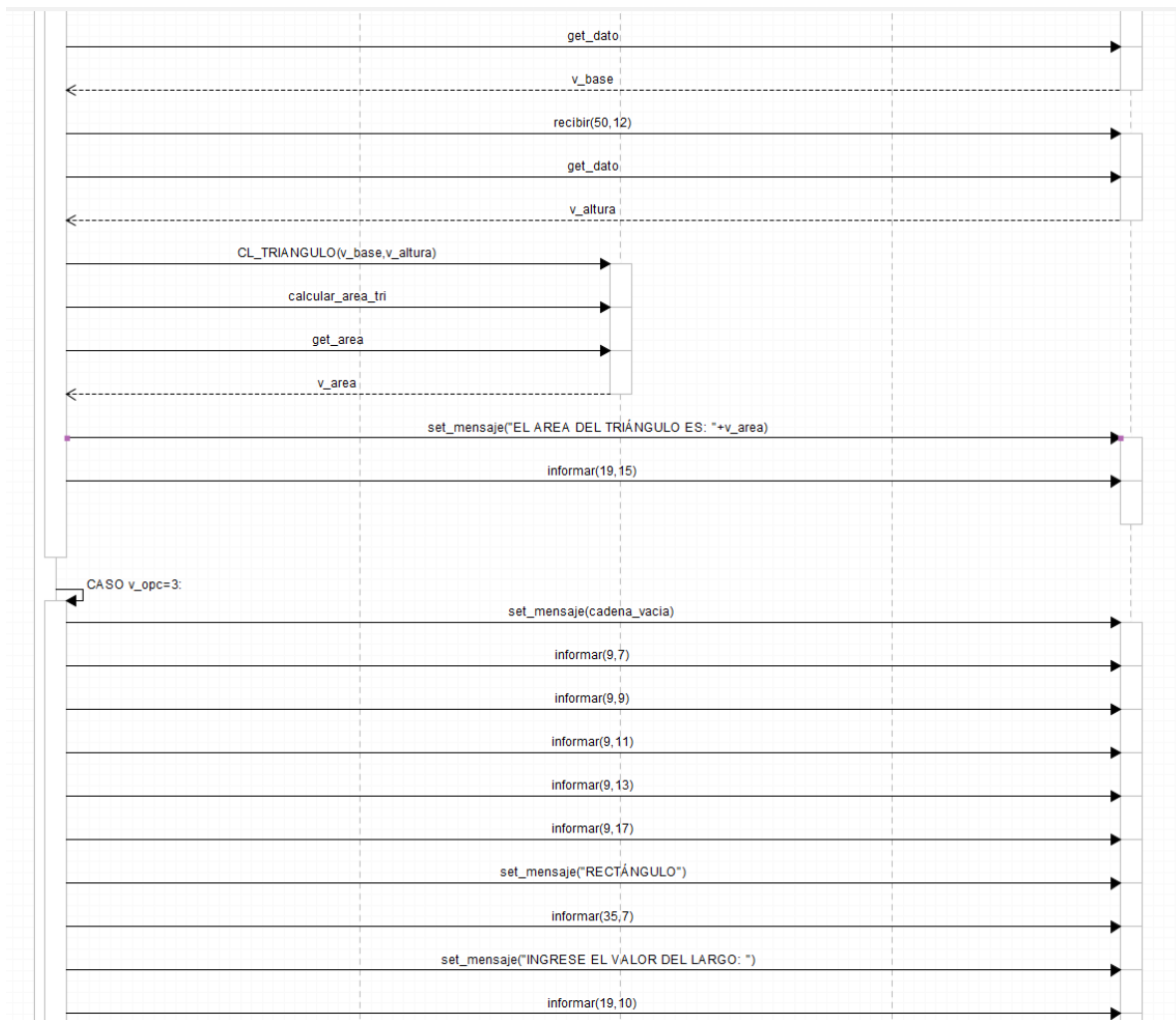


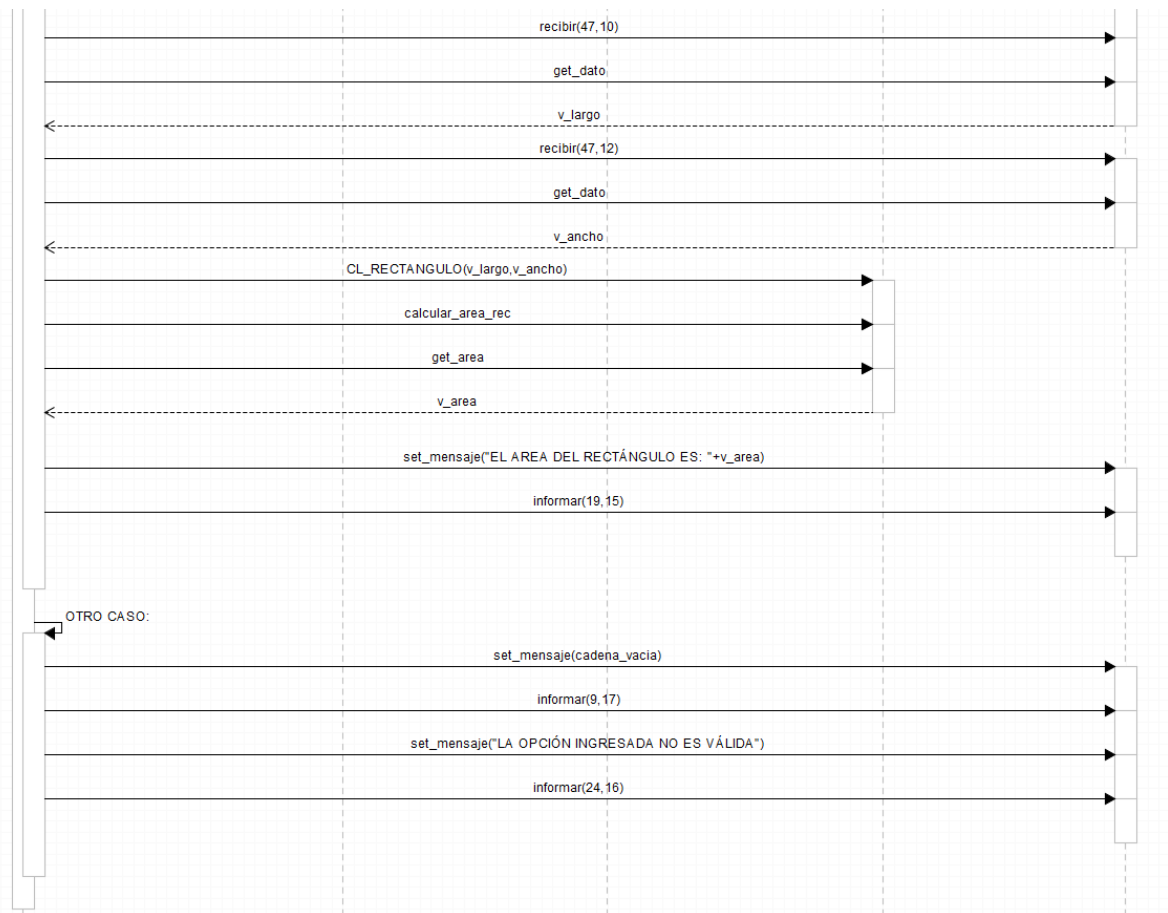
|  |                         |  |
|--|-------------------------|--|
|  | informar(1,15)          |  |
|  | informar(1,16)          |  |
|  | informar(1,17)          |  |
|  | informar(1,18)          |  |
|  | informar(1,19)          |  |
|  | informar(80,6)          |  |
|  | informar(80,7)          |  |
|  | informar(80,8)          |  |
|  | informar(80,9)          |  |
|  | informar(80,10)         |  |
|  | informar(80,11)         |  |
|  | informar(80,12)         |  |
|  | informar(80,13)         |  |
|  | informar(80,14)         |  |
|  | informar(80,15)         |  |
|  | informar(80,16)         |  |
|  | informar(80,17)         |  |
|  | informar(80,18)         |  |
|  | informar(80,19)         |  |
|  | set_mensaje(margen_inf) |  |
|  | informar(1,20)          |  |
|  | set_mensaje("MENÚ")     |  |
|  | informar(39,7)          |  |











- Pseudocódigo de la clase de control

```

PROGRAMA CALCULAR_AREA
/*
NOMBRE: RODRIGO DÍAZ SALGUERO
FECHA: 19/11/2024
*/
INICIO
 SECCIÓN DE VARIABLES
 margen_sup : CADENA
 margen_ver : CADENA
 encabezado : CADENA
 div_encabezado : CADENA
 margen_inf : CADENA
 cadena_vacia : CADENA
 v_opc : ENTERO
 v_radio : REAL
 v_base : REAL
 v_altura : REAL
 v_largo : REAL
 v_ancho : REAL
 v_area : REAL
 MÉTODO PRINCIPAL
 INICIO
 //INICIALIZO LAS VARIABLES CON LAS CADENAS QUE SE SETEAN A
LA CLASE VISTA
 margen_sup <--
"=====
"
 margen_ver <-- "||"
 encabezado <-- "||"
ÁREAS DE FIGURAS GEOMÉTRICAS ||"
 div_encabezado <--
"=====
"
 margen_inf <--
"=====
"
 cadena_vacia <-- "

 //INSTANCIA DE CL_VISTA
 interfaz : NUEVO OBJETO DE CL_VISTA
 interfaz.set_mensaje(margen_sup)
 interfaz.informar(1,1)
 interfaz.set_mensaje(margen_ver)
 interfaz.informar(1,2)
 interfaz.informar(80,2)
 interfaz.set_mensaje(encabezado)
 interfaz.informar(1,3)
 interfaz.set_mensaje(margen_ver)
 interfaz.informar(1,4)
 interfaz.informar(80,4)
 interfaz.set_mensaje(div_encabezado)
 interfaz.informar(1,5)

```



```

interfaz.set_mensaje(margen_ver)
interfaz.informar(1,6)
interfaz.informar(1,7)
interfaz.informar(1,8)
interfaz.informar(1,9)
interfaz.informar(1,10)
interfaz.informar(1,11)
interfaz.informar(1,12)
interfaz.informar(1,13)
interfaz.informar(1,14)
interfaz.informar(1,15)
interfaz.informar(1,16)
interfaz.informar(1,17)
interfaz.informar(1,18)
interfaz.informar(1,19)
interfaz.informar(80,6)
interfaz.informar(80,7)
interfaz.informar(80,8)
interfaz.informar(80,9)
interfaz.informar(80,10)
interfaz.informar(80,11)
interfaz.informar(80,12)
interfaz.informar(80,13)
interfaz.informar(80,14)
interfaz.informar(80,15)
interfaz.informar(80,16)
interfaz.informar(80,17)
interfaz.informar(80,18)
interfaz.informar(80,19)
interfaz.set_mensaje(margen_inf)
interfaz.informar(1,20)
interfaz.set_mensaje("MENÚ")
interfaz.informar(39,7)
interfaz.set_mensaje("1.- ÁREA DEL CIRCULO")
interfaz.informar(25,9)
interfaz.set_mensaje("2.- ÁREA DEL TRIÁNGULO")
interfaz.informar(25,11)
interfaz.set_mensaje("3.- ÁREA DEL RECTÁNGULO")
interfaz.informar(25,13)
interfaz.set_mensaje("INGRESE EL NUMERO DE LA OPERACIÓN A
REALIZAR: ")
interfaz.informar(9,17)
interfaz.recibir(55,17)
v_opc <-- CONVERTIR_A_ENTERO(interfaz.get_dato)

//SELECCIÓN MÚLTIPLE SEGÚN LO QUE SELECCIONE EL USUARIO
EN CASO v_opc SEA
1:
INICIO
 interfaz.set_mensaje(cadena_vacia)
 interfaz.informar(9,7)
 interfaz.informar(9,9)
 interfaz.informar(9,11)
 interfaz.informar(9,13)

```

```

interfaz.informar(9,17)
interfaz.set_mensaje("CIRCULO")
interfaz.informar(36,7)
interfaz.set_mensaje("INGRESE EL VALOR DEL
RADIO: ")

interfaz.informar(19,10)
interfaz.recibir(47,10)
v_radio <-- CONVERTIR_A_REAL(interfaz.get_dato)
obj_circulo : NUEVO OBJETO DE
CL_CIRCULO(v_radio)

obj_circulo.calcular_area_cir
v_area <-- obj_circulo.get_area
interfaz.set_mensaje("EL ÁREA DEL CIRCULO ES:
"+v_area)

interfaz.informar(19,14)
FIN

2:
INICIO
interfaz.set_mensaje(cadena_vacia)
interfaz.informar(9,7)
interfaz.informar(9,9)
interfaz.informar(9,11)
interfaz.informar(9,13)
interfaz.informar(9,17)
interfaz.set_mensaje("TRIÁNGULO")
interfaz.informar(34,7)
interfaz.set_mensaje("INGRESE EL VALOR DE LA
BASE: ")

interfaz.informar(19,10)
interfaz.set_mensaje("INGRESE EL VALOR DE LA
ALTURA: ")

interfaz.informar(19,12)
interfaz.recibir(48,10)
v_base <-- CONVERTIR_A_REAL(interfaz.get_dato)
interfaz.recibir(50,12)
v_altura <--
CONVERTIR_A_REAL(interfaz.get_dato)
obj_triangulo : NUEVO OBJETO DE
CL_TRIANGULO(v_base,v_altura)

obj_triangulo.calcular_area_tri
v_area <-- obj_triangulo.get_area
interfaz.set_mensaje("EL AREA DEL TRIÁNGULO ES:
"+v_area)

interfaz.informar(19,15)
FIN

3:
INICIO
interfaz.set_mensaje(cadena_vacia)
interfaz.informar(9,7)
interfaz.informar(9,9)
interfaz.informar(9,11)
interfaz.informar(9,13)

```

```

 interfaz.informar(9,17)
 interfaz.set_mensaje("RECTÁNGULO")
 interfaz.informar(35,7)
 interfaz.set_mensaje("INGRESE EL VALOR DEL
LARGO: ")

 interfaz.informar(19,10)
 interfaz.set_mensaje("INGRESE EL VALOR DEL
ANCHO: ")

 interfaz.informar(19,12)
 interfaz.recibir(47,10)
 v_largo <-- CONVERTIR_A_REAL(interfaz.get_dato)
 interfaz.recibir(47,12)
 v_altura <--
CONVERTIR_A_REAL(interfaz.get_dato)
 obj_rectangulo : NUEVO OBJETO DE
CL_RECTANGULO(v_largo,v_alto)
 obj_rectangulo.calcular_area_rec
 v_area <-- obj_rectangulo.get_area
 interfaz.set_mensaje("EL AREA DEL RECTÁNGULO ES
ES: "+v_area)
 interfaz.informar(19,15)
 FIN

 OTRO CASO:
 INICIO
 interfaz.set_mensaje(cadena_vacia)
 interfaz.informar(9,17)
 interfaz.set_mensaje("LA OPCIÓN INGRESADA NO ES
VÁLIDA")
 interfaz.informar(24,16)
 FIN
 FIN MÉTODO PRINCIPAL
FIN PROGRAMA CALCULAR_AREA

```

- Implementación en PASCAL posterior a que ya tenemos hechos nuestros pseudocódigos.

- Implementación de la clase CL\_CIRCULO

```
unit U_CL_Circulo;

{$mode objfpc}{$H+}

interface

uses
 U_Figura_Geometrica;

type
 { Clase CL_Circulo derivada de Figura_Geometrica }
 CL_CIRCULO = class(FIGURA_GEOMETRICA)
 private
 r: Real; // Atributo privado
 public
 // Métodos
 constructor Create(radius: Real);
 procedure set_r(x: Real);
 function get_r: Real;
 procedure calcular_area_cir;
 end;

implementation
```

```

{ Implementación de los métodos }

constructor CL_CIRCULO.Create(radio: Real);
begin
 r := radio;
end;

procedure CL_CIRCULO.set_r(x: Real);
begin
 r := x;
end;

function CL_CIRCULO.get_r: Real;
begin
 Result := r;
end;

procedure CL_CIRCULO.calcular_area_cir;
begin
 area := Pi * r * r;
end;

end.

```

#### ○ Implementación de CL\_TRIANGULO

```

unit U_CL_Triangulo;

{$mode objfpc}{$H+}

interface

uses
 U_Figura_Geometrica;

type
 { Clase CL_Triangulo derivada de Figura_Geometrica }
 CL_TRIANGULO = class(FIGURA_GEOMETRICA)
 private
 base: Real; // Atributo privado
 altura: Real; // Atributo privado
 public
 // Métodos
 constructor Create(p_base: Real; p_altura: Real);
 procedure set_base(x: Real);
 procedure set_altura(x: Real);
 function get_base: Real;
 function get_altura: Real;
 procedure calcular_area_tri;
 end;

implementation

{ Implementación de los métodos }

```

```

constructor CL_TRIANGULO.Create(p_base: Real; p_altura: Real);
begin
 base := p_base;
 altura := p_altura;
end;

procedure CL_TRIANGULO.set_base(x: Real);
begin
 base := x;
end;

procedure CL_TRIANGULO.set_altura(x: Real);
begin
 altura := x;
end;

function CL_TRIANGULO.get_base: Real;
begin
 Result := base;
end;

function CL_TRIANGULO.get_altura: Real;
begin
 Result := altura;
end;

procedure CL_TRIANGULO.calcular_area_tri;
begin
 area := (base * altura) / 2;
end;

end.

```

### ○ Implementación de CL\_RECTANGULO

```

unit U_CL_Rectangulo;

{$mode objfpc}{$H+}

interface

uses
 U_Figura_Geometrica;

type
 { Clase CL_Rectangulo derivada de Figura_Geometrica }
 CL_RECTANGULO = class(FIGURA_GEOMETRICA)
 private
 largo: Real; // Atributo privado
 ancho: Real; // Atributo privado
 end;

```

```

public
 // Métodos
 constructor Create(p_largo: Real; p_ancho: Real);
 procedure set_largo(x: Real);
 procedure set_ancho(x: Real);
 function get_largo: Real;
 function get_ancho: Real;
 procedure calcular_area_rec;
end;

implementation

{ Implementación de los métodos }

constructor CL_RECTANGULO.Create(p_largo: Real; p_ancho: Real);
begin
 largo := p_largo;
 ancho := p_ancho;
end;

procedure CL_RECTANGULO.set_largo(x: Real);
begin
 largo := x;
end;

procedure CL_RECTANGULO.set_ancho(x: Real);
begin
 ancho := x;
end;

function CL_RECTANGULO.get_largo: Real;
begin
 Result := largo;
end;

function CL_RECTANGULO.get_ancho: Real;
begin
 Result := ancho;
end;

procedure CL_RECTANGULO.calcular_area_rec;
begin
 area := largo * ancho;
end;

end.

```

### ○ Implementación de FIGURA\_GEOMETRICA

```

unit U_Figura_Geometrica;

```

```

{$mode objfpc}{$H+}

interface

type
 { Clase Figura_Geometrica }
 FIGURA_GEOMETRICA = class
 protected
 area: Real; // Atributo protegido
 public
 // Métodos
 procedure set_area(x: Real);
 function get_area: Real;
 end;

implementation

{ Implementación de los métodos }

procedure FIGURA_GEOMETRICA.set_area(x: Real);
begin
 area := x;
end;

function FIGURA_GEOMETRICA.get_area: Real;
begin
 Result := area;
end;

end.

```

### ○ Implementación de CL\_VISTA

```

unit U_CL_VISTA;

{$mode objfpc}{$H+}

interface

uses
 crt; // Necesario para usar GotoXY

type
 CL_VISTA = class
 private
 mensaje: string;
 dato: string;
 public
 procedure set_mensaje(x: string);
 procedure set_dato(x: string);
 function get_mensaje: string;

```



```

 function get_dato: string;
 procedure informar(x: integer; y: integer);
 procedure recibir(x: integer; y: integer);
 end;

implementation
{
procedure POSICIONAR(x: integer; y: integer);
begin
 GotoXY(x, y); // Posiciona el cursor en la consola en las
 coordenadas (x, y)
end;}

procedure CL_VISTA.set_mensaje(x: string);
begin
 mensaje := x;
end;

procedure CL_VISTA.set_dato(x: string);
begin
 dato := x;
end;

function CL_VISTA.get_mensaje: string;
begin
 Result := mensaje;
end;

function CL_VISTA.get_dato: string;
begin
 Result := dato;
end;

procedure CL_VISTA.informar(x: integer; y: integer);
begin
 GotoXY(x, y);
 WriteLn(mensaje);
end;

procedure CL_VISTA.recibir(x: integer; y: integer);
begin
 GotoXY(x, y);
 ReadLn(dato);
end;

end.

```

- Implementación de la clase de control en PASCAL

```
program Calcular_Area;
{$mode objfpc}{$H+}

uses
 U_CL_VISTA, U_CL_CIRCULO, U_CL_TRIANGULO, U_CL_RECTANGULO, SysUtils,
 crt;

var
 margen_sup, margen_ver, encabezado, div_encabezado, margen_inf,
 cadena_vacia: string;
 v_opc: Integer;
 v_radio, v_base, v_altura, v_largo, v_ancho, v_area: Real;
 interfaz: CL_VISTA;
 obj_circulo: CL_CIRCULO;
 obj_triangulo: CL_TRIANGULO;
 obj_rectangulo: CL_RECTANGULO;

begin
 // INICIALIZO LAS VARIABLES CON LAS CADENAS QUE SE SETEAN A LA CLASE
 VISTA
 margen_sup :=
 '*****';
 margen_ver := '*';
 encabezado := '*
 CALCULAR AREAS DE FIGURAS
GEOMETRICAS *';
```

```

 div_encabezado :=
'*****';
 margen_inf :=
'*****';
 cadena_vacia := '
';

// INSTANCIA DE CL_VISTA
interfaz := CL_VISTA.Create;
interfaz.set_mensaje(margen_sup);
interfaz.informar(1,1);
interfaz.set_mensaje(margen_ver);
interfaz.informar(1,2);
interfaz.informar(80,2);
interfaz.set_mensaje(encabezado);
interfaz.informar(1,3);
interfaz.set_mensaje(margen_ver);
interfaz.informar(1,4);
interfaz.informar(80,4);
interfaz.set_mensaje(div_encabezado);
interfaz.informar(1,5);
interfaz.set_mensaje(margen_ver);
interfaz.informar(1,6);
interfaz.informar(1,7);
interfaz.informar(1,8);
interfaz.informar(1,9);
interfaz.informar(1,10);
interfaz.informar(1,11);
interfaz.informar(1,12);
interfaz.informar(1,13);
interfaz.informar(1,14);
interfaz.informar(1,15);
interfaz.informar(1,16);
interfaz.informar(1,17);
interfaz.informar(1,18);
interfaz.informar(1,19);
interfaz.informar(80,6);
interfaz.informar(80,7);
interfaz.informar(80,8);
interfaz.informar(80,9);
interfaz.informar(80,10);
interfaz.informar(80,11);
interfaz.informar(80,12);
interfaz.informar(80,13);
interfaz.informar(80,14);
interfaz.informar(80,15);
interfaz.informar(80,16);
interfaz.informar(80,17);
interfaz.informar(80,18);
interfaz.informar(80,19);
interfaz.set_mensaje(margen_inf);
interfaz.informar(1,20);

```

```

interfaz.set_mensaje('MENU');
interfaz.informar(39,7);
interfaz.set_mensaje('1.- AREA DEL CIRCULO');
interfaz.informar(25,9);
interfaz.set_mensaje('2.- AREA DEL TRIANGULO');
interfaz.informar(25,11);
interfaz.set_mensaje('3.- AREA DEL RECTANGULO');
interfaz.informar(25,13);
interfaz.set_mensaje('INGRESE EL NUMERO DE LA OPERACION A REALIZAR:
');
interfaz.informar(9,17);
interfaz.recibir(55,17);
v_opc := StrToInt(interfaz.get_dato);

// SELECCIÓN MÚLTIPLE SEGÚN LO QUE SELECCIONE EL USUARIO
case v_opc of
 1:
 begin
 interfaz.set_mensaje(cadena_vacia);
 interfaz.informar(9,7);
 interfaz.informar(9,9);
 interfaz.informar(9,11);
 interfaz.informar(9,13);
 interfaz.informar(9,17);
 interfaz.set_mensaje('CIRCULO');
 interfaz.informar(36,7);
 interfaz.set_mensaje('INGRESE EL VALOR DEL RADIO: ');
 interfaz.informar(19,10);
 interfaz.recibir(47,10);
 v_radio := StrToFloat(interfaz.get_dato);
 obj_circulo := CL_CIRCULO.Create(v_radio);
 obj_circulo.calcular_area_cir;
 v_area := obj_circulo.get_area;
 interfaz.set_mensaje('EL ÁREA DEL CIRCULO ES: ' +
FloatToStr(v_area));
 interfaz.informar(19,14);
 end;
 2:
 begin
 interfaz.set_mensaje(cadena_vacia);
 interfaz.informar(9,7);
 interfaz.informar(9,9);
 interfaz.informar(9,11);
 interfaz.informar(9,13);
 interfaz.informar(9,17);
 interfaz.set_mensaje('TRIANGULO');
 interfaz.informar(34,7);
 interfaz.set_mensaje('INGRESE EL VALOR DE LA BASE: ');
 interfaz.informar(19,10);
 interfaz.set_mensaje('INGRESE EL VALOR DE LA ALTURA: ');
 interfaz.informar(19,12);
 interfaz.recibir(48,10);
 v_base := StrToFloat(interfaz.get_dato);
 interfaz.recibir(50,12);
 end;
end;

```

```

 v_altura := StrToFloat(interfaz.get_dato);
 obj_triangulo := CL_TRIANGULO.Create(v_base, v_altura);
 obj_triangulo.calcular_area_tri;
 v_area := obj_triangulo.get_area;
 interfaz.set_mensaje('EL AREA DEL TRIANGULO ES: ' +
FloatToStr(v_area));
 interfaz.informar(19,15);
 end;
3:
 begin
 interfaz.set_mensaje(cadena_vacia);
 interfaz.informar(9,7);
 interfaz.informar(9,9);
 interfaz.informar(9,11);
 interfaz.informar(9,13);
 interfaz.informar(9,17);
 interfaz.set_mensaje('RECTANGULO');
 interfaz.informar(35,7);
 interfaz.set_mensaje('INGRESE EL VALOR DEL LARGO: ');
 interfaz.informar(19,10);
 interfaz.set_mensaje('INGRESE EL VALOR DEL ANCHO: ');
 interfaz.informar(19,12);
 interfaz.recibir(48,10);
 v_largo := StrToFloat(interfaz.get_dato);
 interfaz.recibir(50,12);
 v_ancho := StrToFloat(interfaz.get_dato);
 obj_rectangulo := CL_RECTANGULO.Create(v_largo, v_ancho);
 obj_rectangulo.calcular_area_rec;
 v_area := obj_rectangulo.get_area;
 interfaz.set_mensaje('EL AREA DEL RECTANGULO ES: ' +
FloatToStr(v_area));
 interfaz.informar(19,16);
 end;
 else
 interfaz.set_mensaje('OPCION INVALIDA');
 interfaz.informar(39,19);
 end;

 interfaz.Free;
end.

```

- Compilación exitosa del programa

```
C:\Users\rodri\OneDrive\Documentos\ParadigmasDeProgramación\PROYECTOS\
FIGURAS GEOMETRICAS\PASCAL>fpc Calcular_Area.pas
Free Pascal Compiler version 3.2.2 [2021/05/15] for i386
Copyright (c) 1993-2021 by Florian Klaempfl and others
Target OS: Win32 for i386
Compiling Calcular_Area.pas
Compiling U_CL_Circulo.pas
Compiling U_Figura_Geometrica.pas
Compiling U_CL_Triangulo.pas
Compiling U_CL_Rectangulo.pas
Linking Calcular_Area.exe
360 lines compiled, 0.3 sec, 81536 bytes code, 5332 bytes data

C:\Users\rodri\OneDrive\Documentos\ParadigmasDeProgramación\PROYECTOS\
FIGURAS GEOMETRICAS\PASCAL>
```

- Ejecución del programa

```
Símbolo del sistema - Calcula X + v

*:\Users\rodri\OneDrive\Documentos\ParadigmasDeProgramación\PROYECTOS\FIGURAS G=0METRICAS\PASCAL>Calcular_Area.exe
* CALCULAR AREAS DE FIGURAS GEOMETRICAS *
*

*
* MENU *
*
* 1.- AREA DEL CIRCULO *
*
* 2.- AREA DEL TRIANGULO *
*
* 3.- AREA DEL RECTANGULO *
*
*
* INGRESE EL NUMERO DE LA OPERACION A REALIZAR:
*

```

```

*: \Users\rodri\OneDrive\Documentos\ParadigmasDeProgramación\PROYECTOS\FIGURAS G*OMETRICAS\PASCAL>Calcular_Area.exe
* CALCULAR AREAS DE FIGURAS GEOMETRICAS
*

*
* CIRCULO
*
*
* INGRESE EL VALOR DEL RADIO: 8
*
*
* EL AREA DEL CIRCULO ES: 201.061929829747
*
C:\Users\rodri\OneDrive\Documentos\ParadigmasDeProgramación\PROYECTOS\FIGURAS GEOMETRICAS\PASCAL>
*
*
*

```