****

**Reto de la semana 3**

**FORMADOR : JHON JAIRO OROZCO DÁVILA Esp.**

**Reto 3:**

**DIAGRAMA DE CLASES**

**IMPLEMENTACIÓN EN JAVA**

**Objetivo**

Utilizar los conocimientos adquiridos, durante las semanas uno, dos, tres, para solucionar problemas.

Implementar programas con expresiones lógicas para resolver un problema que involucre la toma de decisiones

Descomponer un problema en sub-problemas más pequeños y manejables para facilitar la implementación del programa

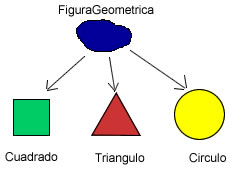
Implementar Diagrama de Clases, implementar programación orientada a objetos, instanciando y mostrando la información por consola.

**Descripción del Reto**

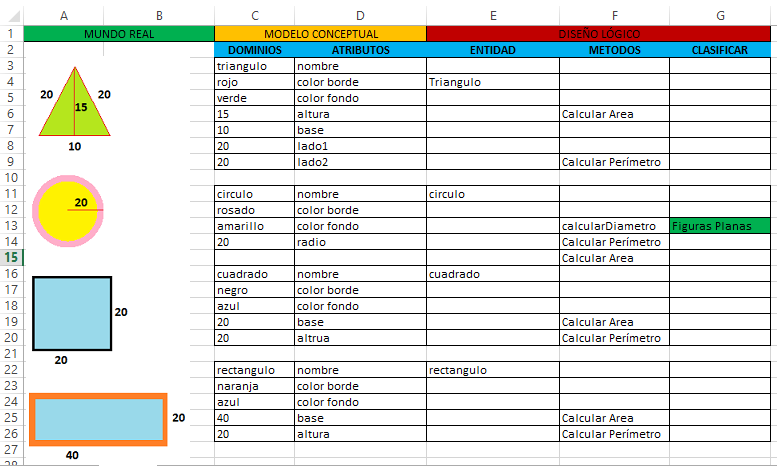
**FIGURAS PLANAS**

Teniendo como referencia la siguiente gráfica de figuras planas presentar:

* Análisis previo, Excel (Dominios – Atributos – Clases – Métodos - Asociación)
* Diagrama de Clases en StarUML basado en el análisis previo (Clases, Atributos, Métodos, Asociaciones)
* Asociación donde se requiera (Generalización – Agregación – Composición)
* Desarrollar programa en Java, aplicando la Programación Orientada a Objetos
* Controla por un menú la construcción de cada una de las figuras, mostrando por consola sus respectivos datos, tanto de la superclase como la subclase
* Mostrar para todas las figuras tanto el área como su perímetro.

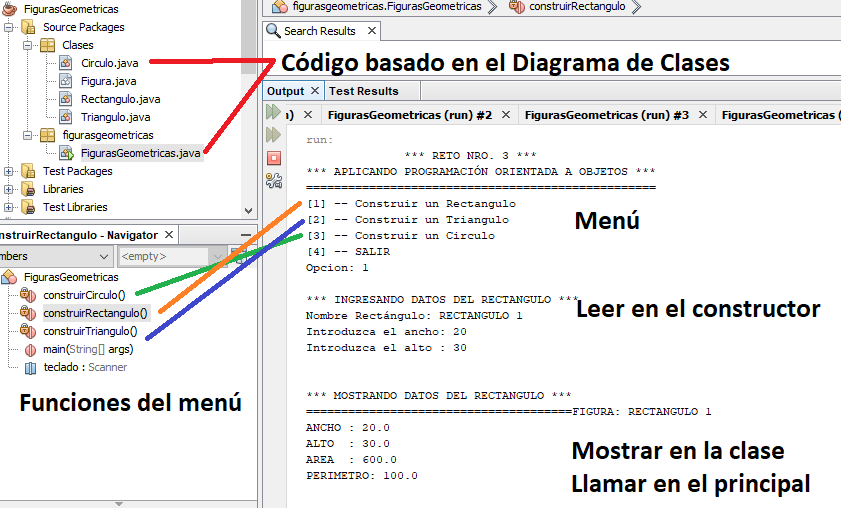


**ANÁLISIS PREVIO – BASADO EN LA FIGURA RESPECTIVA**



**CÓDIGO EN JAVA**

**PROGRAMACIÓN ORIENTADA A OBJETOS**



**AYUDA**:

Cuando se comparten las mismas operaciones como (calcularArea y calcularPerimetro), dependiendo de la figura cambia la fórmula, entonces;

1. En la superclase, solamente se crean los encabezados de las funciones:

/\*

\* Métodos abstractos, cada figura "hija"

\* sobreescribirá estos métodos para hacer

\* los cálculos de estos atributos a su manera

\*/

protected abstract void calcularArea();

protected abstract void calcularPerimetro();

1. En las subclases o clases hijas en el constructor:

* En el constructor se invocan las funciones calcularArea(); calcularPerimetro();
* Se construye el método con la formula respectiva

Por ejemplo para el rectángulo, el @Override indica que es un método que estaba pendiente de su estructura y se definio en la superclase

@Override

protected void calcularPerimetro() {

perimetro = (2\*alto) + (2\*ancho);

}

**Entregas:**

**Reto 3:** **Hasta el viernes 8 - Julio**

¡Mucha suerte!