SESIÓN 6: Clases y Objetos II. Herencia, Clases Abstractas e Interfaces.

Objetivos

- Profundizar y repasar el diseño de clases a través de la clase *Complejo*.
- Reescribir la clase *Fraccion* que implemente la interface *Comparable.*
- Desarrollar clases derivadas o *subclases* a partir de una clase base o *superclase*.
- Saber invocar al constructor de la *superclase* usando la palabra reservada *super*.
- Saber sobrescribir métodos de objeto en las *subclases.*

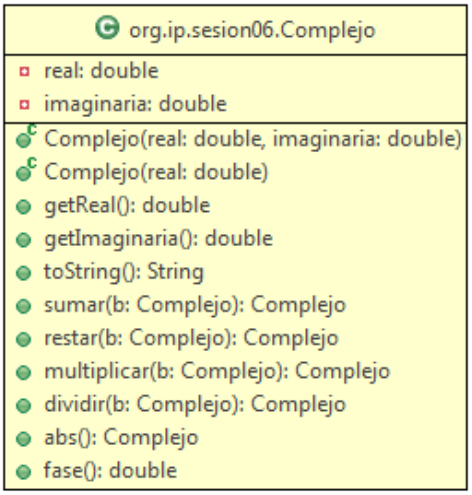
Nota importante: Siga el esquema de nombrado de paquetes que se indicó en la sesión 01 es decir: **org.ip.sesion06**. En ese paquete se crearán todos los programas que se proponen en la sesión dándoles un nombre al programa y que se indica en cada ejercicio entre **paréntesis y en negrita**.
Al final de la sesión, el alumno deberá cargar el trabajo realizado a su repositorio personal indicando la clave correspondiente a la sesión.

Ejercicios propuestos

1. Un número complejo es un número de la forma $a + bi$, donde a y b son números reales e i es $\sqrt{-1}$. Los números a y b se conocen como la parte real e imaginaria de un número complejo respectivamente. Algunas de las operaciones habituales con números complejos son la suma, resta, multiplicación, división, valor absoluto y fase cuyas fórmulas son:

$$a + bi + c + di = (a + c) + (b + d)i$$
$$a + bi - (c + di) = (a - c) + (b - d)i$$
$$(a + bi) * (c + di) = (ac - bd) + (bc + ad)i$$
$$(a + bi) / (c + di) = (ac + bd) / (c^2 + d^2) + (bc - ad)i / (c^2 + d^2)$$
$$|a + bi| = \sqrt{a^2 + b^2}$$
$$fase = arcotangente(b/a)$$

Implementa una clase llamada *Complejo* **(Complejo)** para representar dichos números y con las operaciones que se indican en el siguiente diagrama de clases.



Sobrescribe el método *toString* para que muestre un número complejo de la forma:

a	Si la parte imaginaria es 0
bi	Si la parte real es 0
a - bi	Si la parte imaginaria es menor que cero
a + bi	En cualquier otro caso

Crea el correspondiente diagrama de clases **(complejo.cld).**
A continuación, implementa un programa **(TestComplejo)** que permita probar la clase anterior. A modo de ejemplo, se proporciona una salida que te permitirá comprobar si las operaciones que se han implementado en la clase *Complejo* son correctas.

Ejemplo de ejecución:

```
PRIMER COMPLEJO
Introduce la parte real
3.5
Introduce la parte imaginaria
5.5
SEGUNDO COMPLEJO
Introduce la parte real
-3.5
Introduce la parte imaginaria
1.0

RESULTADOS DE LAS OPERACIONES
(3.5 + 5.5i) + (-3.5 + 1.0i) = 6.5i
(3.5 + 5.5i) - (-3.5 + 1.0i) = 7.0 + 4.5i
(3.5 + 5.5i) * (-3.5 + 1.0i) = -17.75 - 15.75i
(3.5 + 5.5i) / (-3.5 + 1.0i) = -0.5094339622641509 - 1.7169811320754718i
|3.5 + 5.5i| = 6.519202405202649
La fase del primer complejo es 1.0040671092713902 radianes
```

Trabajo autónomo

2. Implementa en este ejercicio la jerarquía de clases vistas en teoría (*ObjGeometrico* (clase padre), *RectanguloHer* (clase derivada) y *CirculoHer* (clase derivada)) con algún atributo adicional tal y como se muestra en el siguiente diagrama de clases. Los nombre de las clases deben ser: **ObjGeometrico**, **RectanguloHer**, y **CirculoHer**.

También se debe implementar la clase **TrianguloHer** que extiende la clase *ObjGeometrico*. Esta nueva clase derivada debe contener:

- Tres atributos `double` denominados `lonitudLado1`, `longitudLado2` y `longitudLado3` con valor por defecto 1.0 y representan la longitud de cada uno de los lados del triángulo.
- Debe implementar tres constructores: `public TrianguloHer()`, `public TrianguloHer(double longitudLado1, double longitudLado2, double longitudLado3)` y `public TrianguloHer(double longitudLado1, double longitudLado2, double longitudLado3, String color, boolean relleno)`.
- También hay que implementar los métodos getters and setters relativos a los tres atributos anteriores.
- Implementar un método `toString()` que redefina dicho método de la clase base, como para los casos de *RectanguloHer* y *CirculoHer*.
- Implementar dos nuevos métodos para obtener el perímetro (`public double calcularArea()`) y el área (`public double calcularPerimetro()`), para este último utilizar la *fórmula de Herón*.

$Area = \sqrt{s(s-l_1)(s-l_2)(s-l_3)}$, donde *s* es el semiperímetro del triángulo

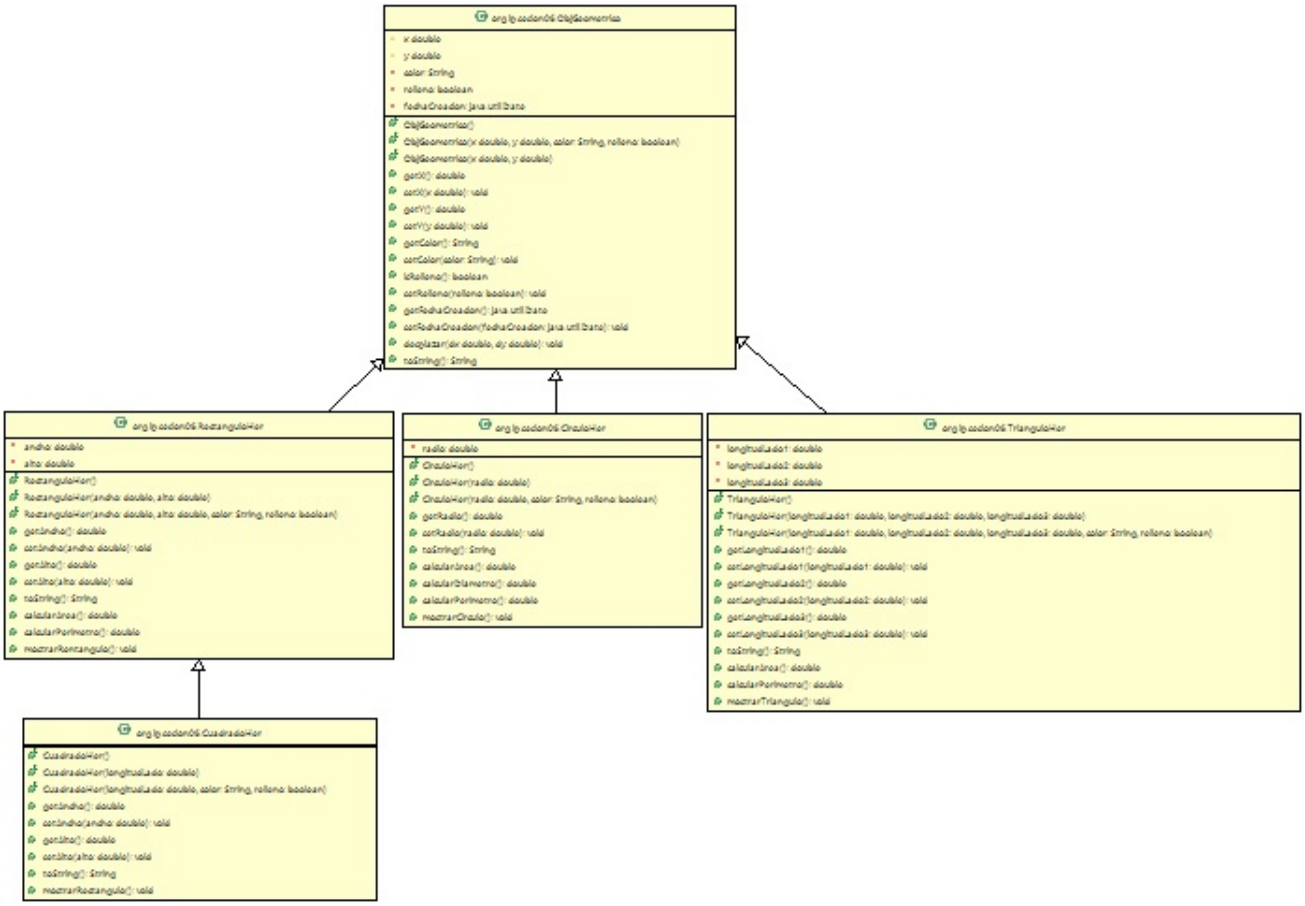
$$s = \frac{l_1 + l_2 + l_3}{2}$$

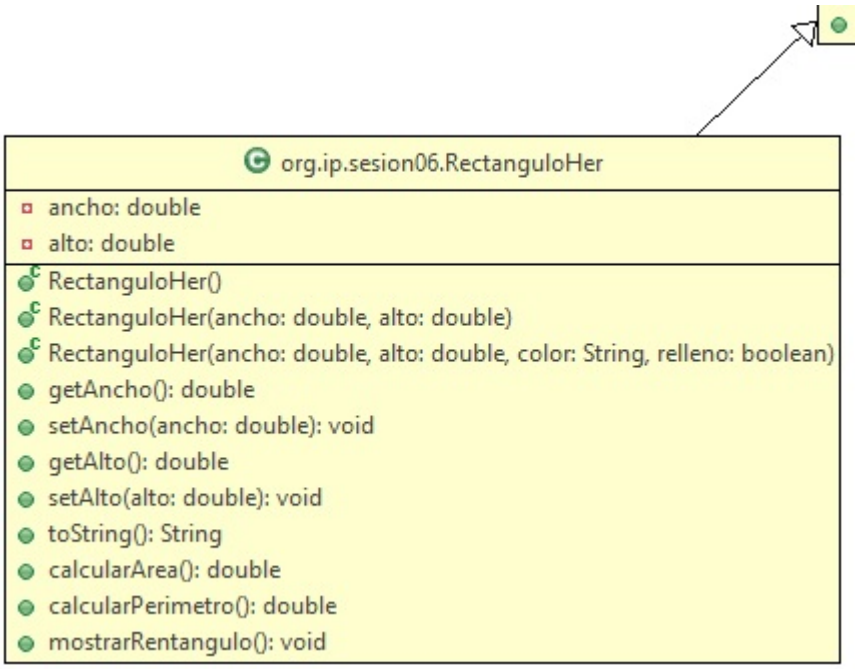
- Y por último, debe implementar también el método `public void mostrarTriangulo()` para mostrar toda la información relevante y clara del triángulo en cuestión, como para los casos de *RectanguloHer* y *CirculoHer*.

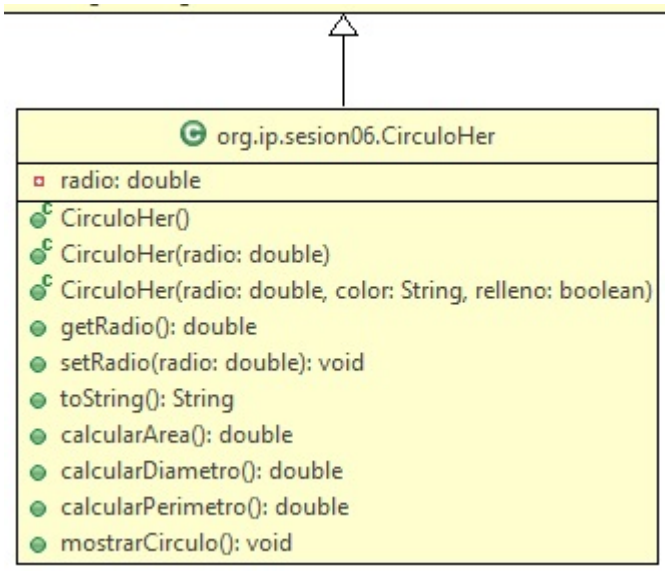
Finalmente se debe implementar la clase **CuadradoHer** que extiende la clase *RectanguloHer*. Esta nueva clase derivada debe contener:

- No tiene atributos propios.
- Tres constructores: `public CuadradoHer()`, `public CuadradoHer(double longitudLado)` y `public TrianguloHer(double longitudLado, String color, boolean relleno)`.
- Hay que refinar los métodos getters and setters heredados, al igual que el método `mostrarRectangulo()`, también heredado.
- Implementar un método `toString()` que redefina dicho método de la clase base, como para el caso de *RectanguloHer*.

Por último, es necesario implementar un programa Java **(TestObjetosGeometricos)** que permita probar el funcionamiento de todo lo anterior.


org.ip.sesion06.ObjGeometrico
x: double
y: double
color: String
relleno: boolean
fechaCreacion: java.util.Date
ObjGeometrico()
ObjGeometrico(x: double, y: double, color: String, relleno: boolean)
ObjGeometrico(x: double, y: double)
getX(): double
setX(x: double): void
getY(): double
setY(y: double): void
getColor(): String
setColor(color: String): void
isRelleno(): boolean
setRelleno(relleno: boolean): void
getFechaCreacion(): java.util.Date
setFechaCreacion(fechaCreacion: java.util.Date): void
desplazar(dx: double, dy: double): void
toString(): String
org.ip.sesion06.RectanguloHor
ancho: double
alto: double
RectanguloHor()
RectanguloHor(ancho: double, alto: double)
RectanguloHor(ancho: double, alto: double, color: String, relleno: boolean)
getAncho(): double
setAncho(ancho: double): void
getAlto(): double
setAlto(alto: double): void
toString(): String
calcularArea(): double
calcularPerimetro(): double
mostrarRectangulo(): void
org.ip.sesion06.CirculoHor
radio: double
CirculoHor()
CirculoHor(radio: double)
CirculoHor(radio: double, color: String, relleno: boolean)
getRadio(): double
setRadio(radio: double): void
toString(): String
calcularArea(): double
calcularDiametro(): double
calcularPerimetro(): double
mostrarCirculo(): void
org.ip.sesion06.TrianguloHor
longitudLado1: double
longitudLado2: double
longitudLado3: double
TrianguloHor()
TrianguloHor(longitudLado1: double, longitudLado2: double, longitudLado3: double)
TrianguloHor(longitudLado1: double, longitudLado2: double, longitudLado3: double, color: String, relleno: boolean)
getLongitudLado1(): double
setLongitudLado1(longitudLado1: double): void
getLongitudLado2(): double
setLongitudLado2(longitudLado2: double): void
getLongitudLado3(): double
setLongitudLado3(longitudLado3: double): void
toString(): String
calcularArea(): double
calcularPerimetro(): double
mostrarTriangulo(): void
org.ip.sesion06.CuadradoHor
CuadradoHor()
CuadradoHor(longitudLado: double)
CuadradoHor(longitudLado: double, color: String, relleno: boolean)
getLongitudLado(): double
setLongitudLado(longitudLado: double): void
toString(): String
mostrarRectangulo(): void

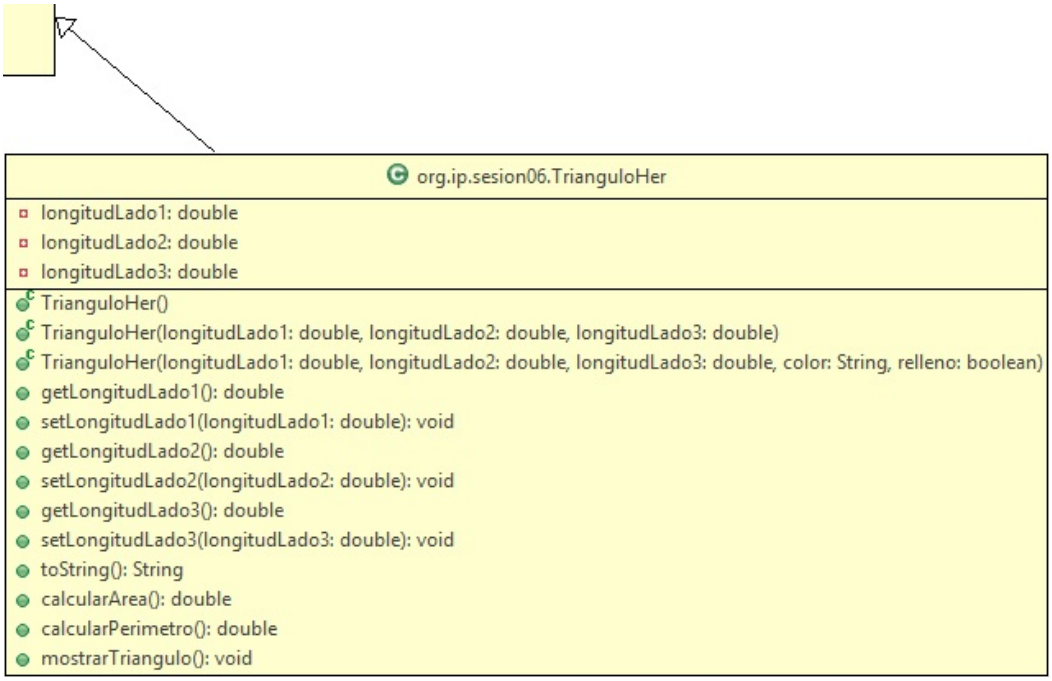

org.ip.sesion06.RectanguloHer
ancho: double
alto: double
RectanguloHer()
RectanguloHer(ancho: double, alto: double)
RectanguloHer(ancho: double, alto: double, color: String, relleno: boolean)
getAncho(): double
setAncho(ancho: double): void
getAlto(): double
setAlto(alto: double): void
toString(): String
calcularArea(): double
calcularPerimetro(): double
mostrarRentangulo(): void


org.ip.sesion06.CirculoHer
radio: double
CirculoHer()
CirculoHer(radio: double)
CirculoHer(radio: double, color: String, relleno: boolean)
getRadio(): double
setRadio(radio: double): void
toString(): String
calcularArea(): double
calcularDiametro(): double
calcularPerimetro(): double
mostrarCirculo(): void

Ejemplo de ejecución (**CirculoHer** y **RectanguloHer**):

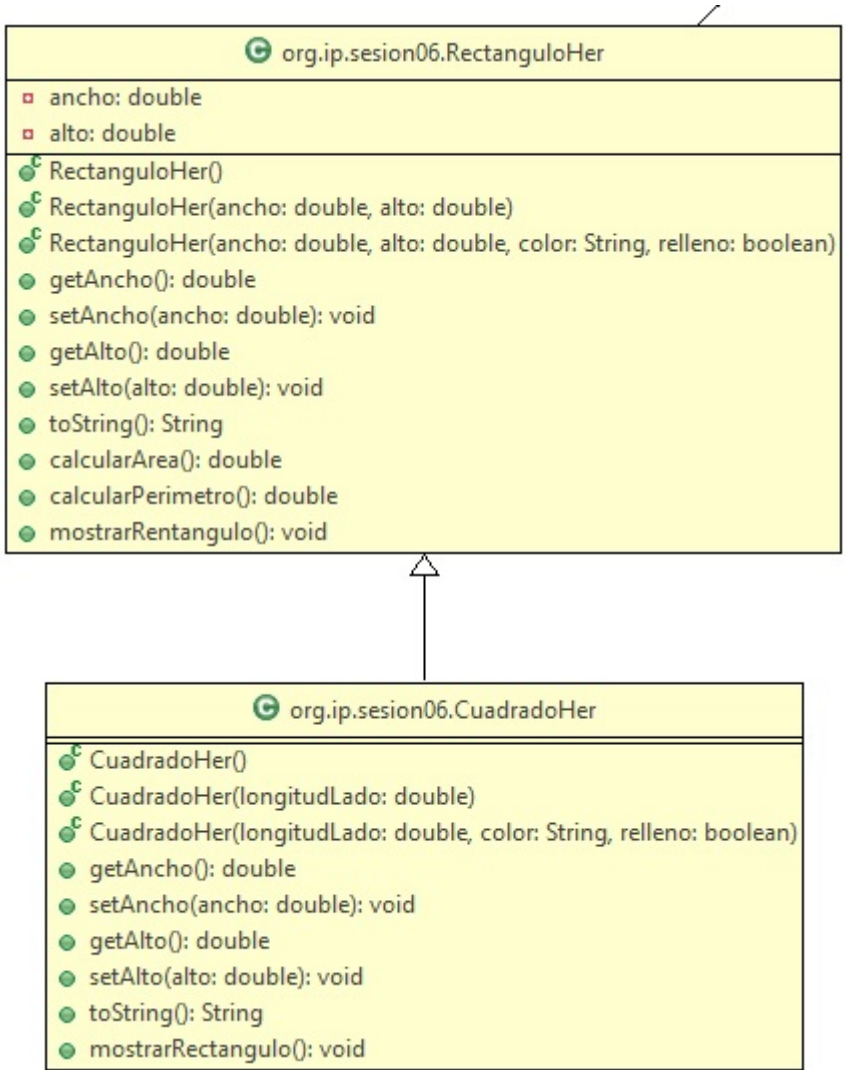
```
Un CirculoHer = ObjGeometrico {(x=0.0, y=0.0), color=blanco, relleno=false, fecha Creacion=Sun Nov 13 23:59:01 CET 2016}, radio=1.0
El color es blanco
El radio es 1.0
La fecha de creacion es Sun Nov 13 23:59:01 CET 2016
El area es 3.141592653589793
El diametro es 2.0
El perimetro es 6.283185307179586
* Datos del circulo son:
El circulo ha sido creado: Sun Nov 13 23:59:01 CET 2016,
es de color blanco, sin relleno,
ubicado en (0.0, 0.0) y el radio es 1.0
* Datos del circulo modificado son:
El circulo ha sido creado: Sun Nov 13 23:59:01 CET 2016,
es de color negro, sin relleno,
ubicado en (1.0, 2.0) y el radio es 5.0

Un RectanguloHer ObjGeometrico {(x=0.0, y=0.0), color=blanco, relleno=false, fecha Creacion=Sun Nov 13 23:59:01 CET 2016}, ancho=2.0, alto=4.0
El area es 8.0
El perimetro es 12.0
* Datos del rectangulo son:
El rectangulo ha sido creado: Sun Nov 13 23:59:01 CET 2016,
es de color blanco, sin relleno,
ubicado en (0.0, 0.0), el ancho es 2.0 y el alto es 4.0
* Datos del rectangulo modificado son:
El rectangulo ha sido creado: Sun Nov 13 23:59:01 CET 2016,
es de color verde, con relleno,
ubicado en (3.0, 4.0), el ancho es 3.0 y el alto es 8.0
```



Ejemplo de ejecución (**TrianguloHer**):

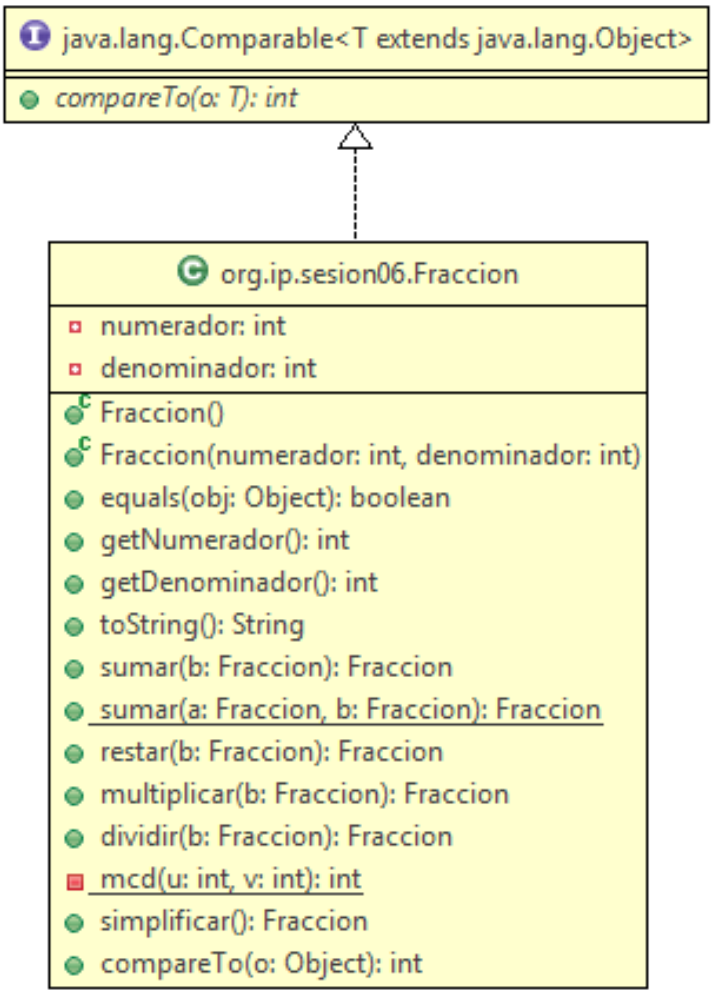
```
Un TrianguloHer = ObjGeometrico {(x=0.0, y=0.0), color=blanco, relleno=false, fecha Creacion=Sun Nov 13 23:59:01 CET 2016}, longitud de lados=[4.0, 4.0, 6.0]
El area es 7.937253933193772
El perimetro es 14.0
* Datos del triangulo son:
El triangulo ha sido creado: Sun Nov 13 23:59:01 CET 2016,
es de color blanco, sin relleno,
ubicado en (0.0, 0.0) y con longutud de lados [4.0, 4.0, 6.0]
* Datos del triangulo modificado son:
El triangulo ha sido creado: Sun Nov 13 23:59:01 CET 2016,
es de color azul, con relleno,
ubicado en (0.0, 0.0) y con longutud de lados [4.0, 4.0, 2.0]
El area es 3.872983346207417
El perimetro es 10.0
```



Ejemplo de ejecución (**CuadradoHer**):

```
Un CuadradoHer RectanguloHer ObjGeometrico {(x=0.0, y=0.0), color=blanco, relleno=false, fecha Creacion=Mon Nov 14 11:14:23 CET 2016}, ancho=5.0, alto=5.0
El area es 25.0
El perimetro es 20.0
* Datos del cuadrado son:
El cuadrado ha sido creado: Mon Nov 14 11:14:23 CET 2016,
es de color blanco, sin relleno,
ubicado en (0.0, 0.0), la longitud del lado es 5.0
* El valor del lado del cuadrado es: 3.0
El area es 9.0
Un CuadradoHer RectanguloHer ObjGeometrico {(x=3.0, y=4.0), color=rojo, relleno=true, fecha Creacion=Mon Nov 14 11:14:23 CET 2016}, ancho=3.0, alto=3.0
* El valor del lado del cuadrado es: 8.0
El perimetro es 32.0
* Datos del cuadrado modificado son:
El cuadrado ha sido creado: Mon Nov 14 11:14:23 CET 2016,
es de color rojo, con relleno,
ubicado en (3.0, 4.0), la longitud del lado es 8.0
Un CuadradoHer RectanguloHer ObjGeometrico {(x=3.0, y=4.0), color=rojo, relleno=true, fecha Creacion=Mon Nov 14 11:14:23 CET 2016}, ancho=8.0, alto=8.0
```

3. Implementa una nueva clase *Fraccion* que implemente la interface *Comparable*. **(Fraccion).** Para ello, utiliza la clase que ya tienes diseñada de la sesión 05 añadiéndole que implemente la interface anterior (deberás implementar el método *compareTo)*.



Crea el diagrama de clases anterior **(fraccion.cld).**
Haz un programa **(TestFraccion)** que pruebe el funcionamiento de la clase anterior.

Ejemplo de ejecución:

```
¿Cuántas comparaciones de fracciones deseas hacer?
3
Comparación 1
Introduce numerador y denominador de la primera fracción
1 5
Introduce numerador y denominador de la segunda fracción
1 5
1/5 es igual que 1/5

Comparación 2
Introduce numerador y denominador de la primera fracción
-1 2
Introduce numerador y denominador de la segunda fracción
1 2
-1/2 es menor que 1/2

Comparación 3
Introduce numerador y denominador de la primera fracción
1 5
Introduce numerador y denominador de la segunda fracción
1 8
1/5 es mayor que 1/8
```