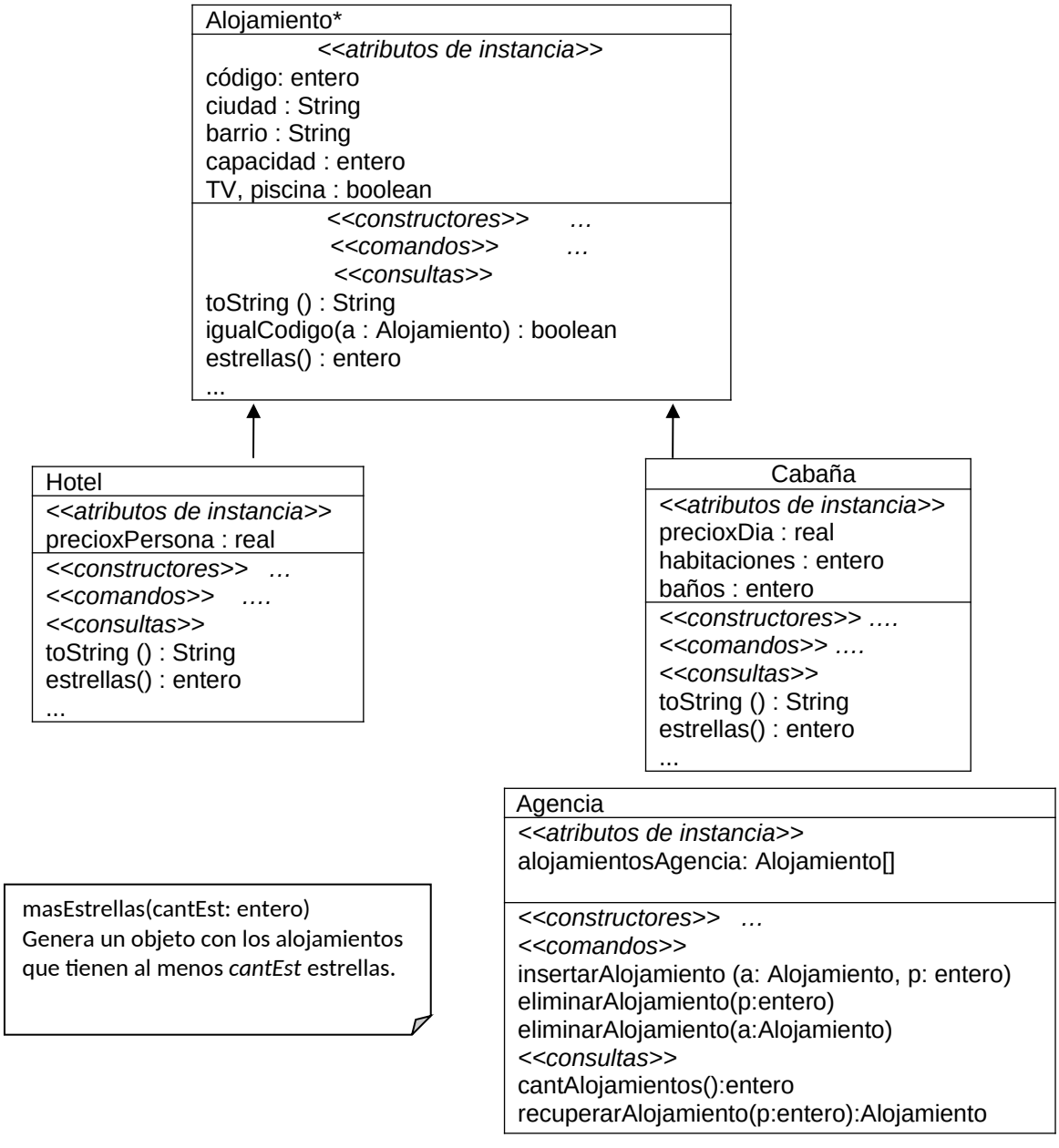
PRACTICO N° 14
Clases Abstractas

EJERCICIO 1. Una agencia de viajes mantiene información referida a los hoteles y cabañas que ofrece en alquiler.
La clase Alojamiento se introduce para factorizar atributos y comportamiento: en la aplicación todos los alojamientos son hoteles cabañas.
La clase Agencia se implementa como una tabla. La clase cliente tiene acceso a los alojamientos por posición. Algunas posiciones pueden ser nulas.

a) Implemente el modelo en Java considerando que cada una de las clases brinda los comandos y consultas triviales y constructores adecuados.
b) Implemente una clase tester que verifique los servicios provistos por la clase Agencia.
c) Agregue en la clase tester un método que muestre los códigos de las cabañas cuyos precios estén en el rango establecido por dos valores ingresados por el usuario.



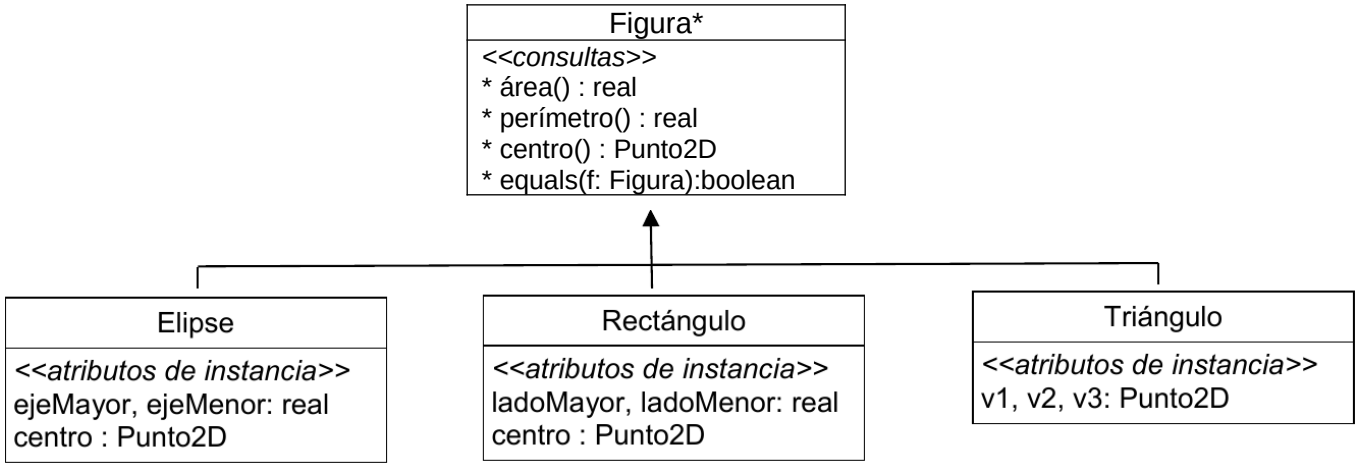
recuperarPosicion(a:Alojamiento):entero estaAlojamiento (a : Alojamiento) : boolean estaLlena () : boolean hayAlojamientos () : boolean masEstrellas (cantEst : entero) : Agencia ...

Los hoteles como mínimo tienen 1 estrella. Si cuentan con TV, son de 2 estrellas. Si cuentan con piscina son de 3 estrellas, y si tiene piscina y TV son de 4 estrellas.
Las cabañas son como mínimo de 2 estrellas y siempre cuentan con TV. Si tienen más de 1 baño son 3 estrellas, y si además tienen piscina son 4 estrellas.
En las consultas recuperarPosición y estáAlojamiento la comparación de alojamientos se hace por identidad.

EJERCICIO 2. Dado el siguiente diagrama donde se muestra la jerarquía de herencia para representar figuras compuestas.

FiguraCompuesta
<<*atributos de instancia*>> partes : Figura[] cantidad : entero
<<*constructores*>> FiguraCompuesta(max: entero) <<*comandos*>> insertar(f: Figura) eliminar(i: entero) <<*consultas*>> cantidad() : entero obtenerPosicion(f: Figura) : entero obtenterFigura(i: entero) : Figura area() : real longitudBordes() : real

Punto2D
<<*atributos de instancia*>> x, y : real
<<*constructor*>> Punto2D(x, y : real) <<*consultas*>> obtenerX() : real obtenerY() : real



(a) Implemente en Java la clase abstracta Figura que representa una figura simple.
(b) Implemente en Java la clase FiguraCompuesta, consideando que el área de una figura compuesta es la suma de las áreas de todas las figuras que la componen, y la longitud de bordes de una figura compuesta es la suma de los perímetros de las figuras que la componen.
(c) Implemente las clases Elipse, Rectángulo y Triángulo respetando la jerarquía propuesta. Tenga en cuenta que:
    1. El perímetro de la elipse se aproxima como $\mathbf{2\pi * \sqrt{((a^2+b^2)/2)}}$ donde *a* y *b* son el semieje mayor (*mitad del eje*) y el semieje menor respectivamente.

2. La superficie de la elipse se calcula como **π* eje mayor * eje menor**.
3. El centro del triángulo es el promedio de sus vértices.
4. La distancia entre dos puntos (x_1, y_1) (x_2, y_2) se calcula con la siguiente fórmula:

$$\mathbf{d} = \sqrt{(\mathbf{x}_2 - \mathbf{x}_1)^2 + (\mathbf{y}_2 - \mathbf{y}_1)^2}$$

5. El área del triángulo se calcula utilizando la *fórmula de Herón* a partir de su semiperímetro (s) y la longitud de sus lados (a, b y c).

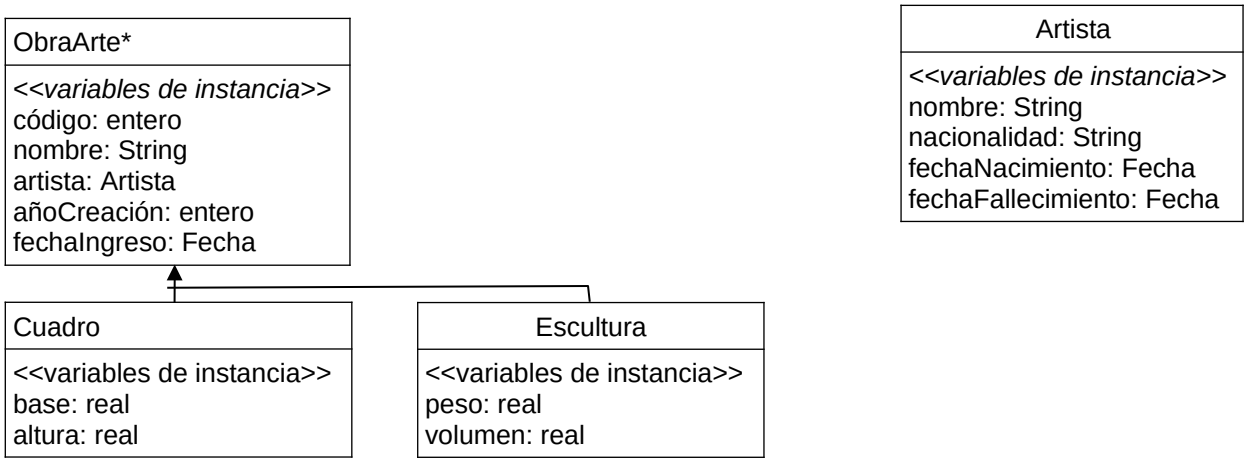
$$Área = \sqrt{s(s-a)(s-b)(s-c)}$$

siendo a, b, c los tres lados y s el semiperímetro $s = \frac{a+b+c}{2}$

(d) Implemente una clase tester que:
1. Cree una figura compuesta con diferentes figuras simples con centro en diferentes posiciones del plano.
2. Elimine de la figura compuesta todas aquellas figuras con área mayor a un determinado valor (que puede ser ingresado por teclado) y cuyo centro tenga *y<0*.
3. Muestre por pantalla las figuras simples que componen la figura.

EJERCICIO 3: El siguiente diagrama de clases modela un centro cultural donde se mantiene información referida a obras de arte y artistas. El código de cada obra la identifica unívocamente y se asume que no hay dos artistas con el mismo nombre. Las obras de arte del centro cultural son únicamente cuadros o esculturas. Un centro cultural puede prestar obras de arte a otro centro cultural.

Dado el siguiente diagrama (en el diagrama de clases sólo se muestran las variables de instancias y las relaciones de herencia):



Prestamo
<<*variables de instancia*>> obras: ObraArte destino: CentroCultural fechaDevolucion: Fecha

ColeccionPrestamos
<<*variables de instancia*>> prestamos: Prestamo[] cantidad: entero
<<*constructor*>> … <<*comandos*>> nuevoPrestamo(p: Prestamo) <<*consultas*>> cantidadPrestamos():entero prestamo(i: entero): Prestamo estaPrestado(b: ObraArte): boolean hayPrestamos(): boolean …

a) **Implemente** la clase ColeccionPrestamos que representa una colección de obras que están prestadas. Considere que el método nuevoPrestamo(p: Prestamo) inserta un nuevo préstamo al final de la tabla.
b) **Implemente** las clases TablaObras y TablaArtistas que encapsulan un arreglo de obras de arte y de artistas respectivamente.
c) **Implemente** la clase CentroCultural que representa un centro cultural con obras de arte.

TablaObras
<<variables de instancia>> obras: ObraArte[] cantidad:entero
<<constructor>> … <<comandos>> insertarObra(o: ObraArte) eliminarObra(o: ObraArte) … <<consultas>> cantidadObras():entero existeObra(o: ObraArte): boolean estáLlena() : boolean hayObras(): boolean recuperarObra(cod: entero): ObraArte obrasArtista(nom: String): TablaObras

TablaArtistas
<<variables de instancia>> artistas : Artista[] cantidad : entero
<<constructor>> … <<comandos>> insertarArtista(A: Artista) … <<consultas>> cantidadArtistas():entero estaArtista(a: Artista): boolean estaLlena() : boolean hayArtistas(): boolean recuperarArtista(p:entero): Artista recuperarArtista(Nom: String): Artista artistasNac(nac:String): TablaArtistas

CentroCultural
<<variables de instancia>> nombre: String obrasExpuestas : TablaObras obrasPrestadas: ColeccionPrestamos totalidadArtistas : TablaArtistas
<<constructor>> … <<comandos>> prestarObra (b:ObraArte, c: CentroCultural) … <<consultas>> obrasNacionalidad(nac:String): TablaObras montoAseguradoPorArtista(art: String): real mayorMontoAsegurado(n: entero): TablaObras artistasPopulares (n:entero): TablaArtistas

En la clase TablaObras:

- insertarObra(o: ObraArte) Inserta la obra de arte ***o*** en la primera posición libre.
- obrasArtista(nom: String): TablaObras Devuelve todas las obras correspondientes a un artista de nombre dado.

En la clase TablaArtistas:

- insertarArtista(a:Artista) inserta el artista ***a*** en la primera posición libre.
- artistasNac(nac:String): TablaArtistas Devuelve todos los artistas de una nacionalidad dada.

En la clase CentroCultural:

- prestarObra(b: ObraArte, c: CentroCultural) Presta la obra b al centro cultural c. Por lo tanto la obra deja de estar en exposición. Todos los préstamos se hacen por un mes.
- obrasNacionalidad(nac:String): TablaObras Genera una tabla con las obras de arte expuestas y que correspondan a artistas de una nacionalidad dada.
- montoAseguradoPorArtista(art: String): real Devuelve el monto asegurado total de todas las obras de un artista de determinado nombre.
- mayorMontoAsegurado(n: entero): TablaObras Devuelve las n obras con mayor monto asegurado.
- artistasPopulares(n: entero):TablaArtistas Devuelve una tabla con los artistas de los cuales haya más de n obras expuestas y al menos una obra prestada.

El monto asegurado para los cuadros se calcula como $500 por cada año de antigüedad y para las esculturas como $800 por cada año de antigüedad más $250 por cada 100 gramos de peso. ¿En qué clase debe declararse el método montoAsegurado():real? ¿En qué clases debe implementarse?