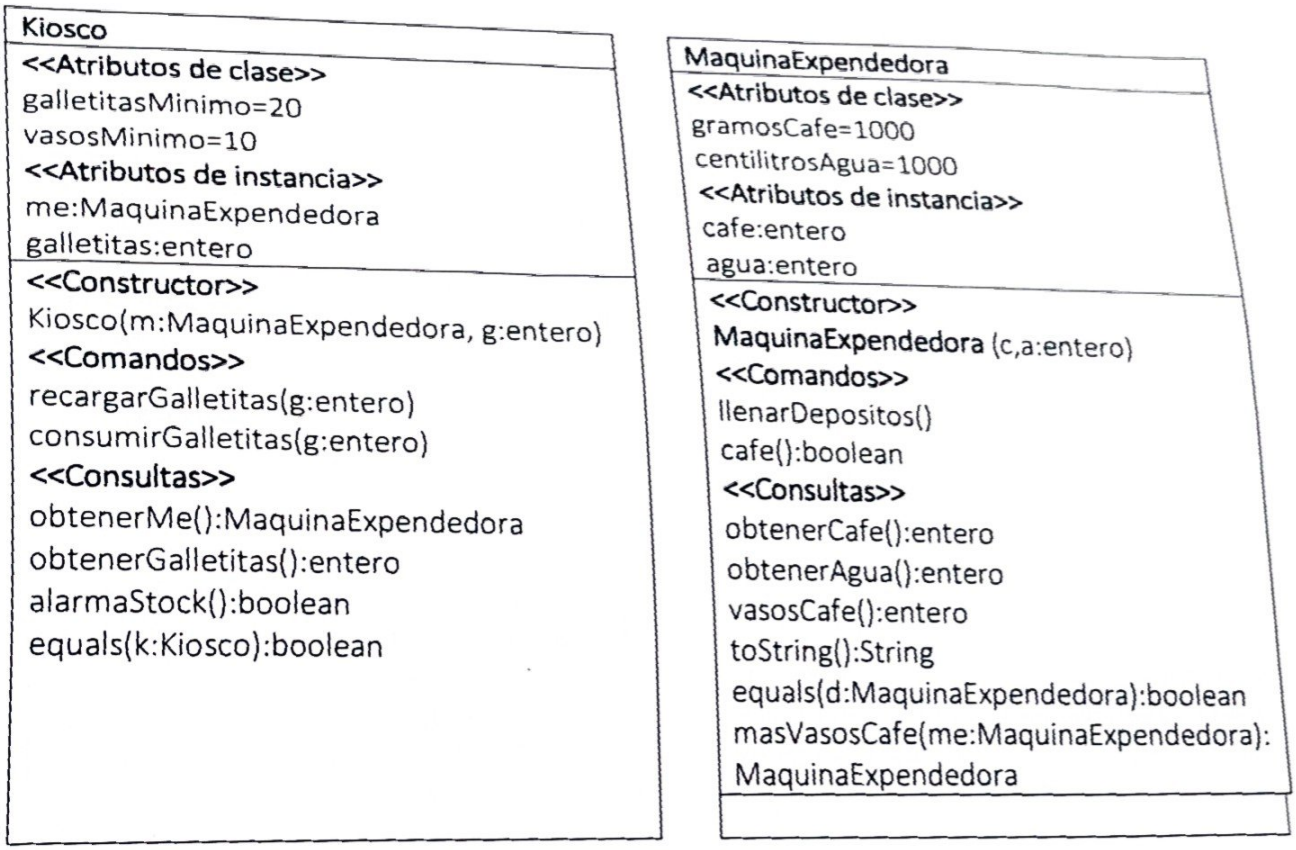
PROBLEMA 1

Un kiosco dispone de una máquina expendedora y un stock de paquetes de galletitas. Una máquina expendedora dispone de un depósito con capacidad para mantener hasta 1000 gramos de café molido y otro depósito con capacidad para mantener hasta 1000 centilitros de agua. Para preparar un vasos de café se necesitan 20 centilitros de agua y 20 gramos de café molido. Implemente el siguiente diagrama de clases de acuerdo a la especificación.



En la clase **MaquinaExpendedora**:

- **MaquinaExpendedora** (c,a:entero) Si **c** es menor a **gramosCafe** inicializa **cafe** con **c**, sino inicializa con la capacidad máxima del depósito de café. Si **a** es menor a **centilitrosAgua** inicializa **agua** con **a**, sino inicializa con la capacidad máxima del depósito de agua. Requiere que **c** y **a** sean mayores a 0.
- **cafe() : boolean** Si la máquina tiene al menos 20 gramos de café y al menos 20 centilitros de agua, reduce cada depósito en esa cantidad y retorna verdadero. Si la máquina no dispone de ingredientes para preparar un vaso de café, el comando retorna false.
- **vasosCafe():entero** retorna la cantidad de vasos de café que puede preparar con los ingredientes disponibles en el depósito.
- **equals(me:MaquinaExpededora):boolean** Requiere **me** ligada. Retorna true si las máquinas tienen el mismo estado interno.

- **masVasosCafe(me:MaquinaExpededora):MaquinaExpendedora** si la máquina que recibe el mensaje puede preparar más vasos de café que la máquina ligada a **me**, retorna la referencia a la máquina que recibe el mensaje. Sino retorna **me**. Requiere **me** ligada

En la clase **Kiosco**:

- **Kiosco(m:MaquinaExpendedora, g:entero)** Requiere **m** ligado y **g** mayor a 0.
- **recargarGalletitas(g:entero)** Requiere **g** mayor a 0. Incrementa el valor de **galletitas** en **g**.
- **consumirGalletitas(g:entero)** Requiere **g** mayor a 0. Si **g** es menor que galletitas, decrementa galletitas en **g**. sino asigna 0 a galletitas.
- **alarmaStock():boolean** retorna verdadero si **galletitas** es menor que **galletitasMinimo** o la máquina expendedora, con los ingredientes disponibles, puede preparar menos de vasosMinimo.
- **equals(k:Kiosco):boolean** Requiere **k** ligado, computa igualdad en profundidad.

PROBLEMA 2

Un color puede representarse con una terna de números. El primero representa la cantidad de color rojo, el segundo la cantidad de color verde y el tercero la cantidad de color azul. El rango para cada valor es 0..255. Combinando el máximo de los tres se obtiene el blanco (255, 255, 255). La ausencia de los tres produce el negro (0, 0, 0). El rojo es (255, 0, 0). Cuando se mantiene el mismo valor para las tres componentes se obtiene gris. La terna (50, 50, 50) representa un gris oscuro, (150, 150, 150) es un gris más claro.
Un punto con color en la pantalla se puede representar con sus coordenadas 2D y el color asociado a ese punto.

ASUMA QUE YA SE HAN IMPLEMENTADO CORRECTAMENTE LAS CLASES Color Y PuntoColor de acuerdo al siguiente diseño:

Color
<<Atributos de instancia>> rojo, verde, azul:entero
<<Constructor>> Color(r:entero, v:entero, a:entero) **<<Comandos>>** establecerRojo(val:entero) establecerAzul(val:entero) establecerVerde(val:entero) **<<Consultas>>** obtenerRojo():entero obtenerAzul():entero obtenerVerde():entero esGris():boolean equals(p:Color):boolean clone () : Color

PuntoColor
<<Atributos de instancia>> x,y:entero color: Color
<<Constructor>> PuntoColor(nX, nY : entero, col:Color) **<<Comandos>>** establecerX(v : entero) establecerY(v : entero) establecerColor(col:Color) **<<Consultas>>** obtenerX(): entero obtenerY(): entero obtenerColor() : Color equals(pc: PuntoColor): boolean clone(): PuntoColor

En la clase **PuntoColor**:

- **PuntoColor(nX, nY : entero, col:Color)** Requiere **col** ligado
- **establecerColor (col:Color)** Requiere **col** ligado
- **equals(pc: PuntoColor): boolean** Requiere **pc** ligado. Se implementa en forma superficial
- **clone(): PuntoColor** Se implementa en profundidad

En la clase **Color**:

- **equals(p:Color):boolean** Requiere **p** ligado

USTED DEBE:

Introducción a la Programación Orientada a Objetos

DCIC - UNS

Segundo cuatrimestre 2022

a) Dibujar el diagrama de objetos al finalizar la ejecución de las instrucciones **5 y 10**.

```
1. Color c1,c2,c3;
2. PuntoColor p1,p2,p3,p4;
3. c1=new Color(255,0,0);
4. c2=c1;
5. c3=c1.clone();
6. p1 = new PuntoColor(0,0,c1);
7. p2 = new PuntoColor(0,0,c2);
8. p3 = new PuntoColor(0,0,c3);
9. p4 = new PuntoColor(0,1,c3);
10. p4.establecerY(0);
```

b) Mostrar los valores de verdad que computan cada una de las siguientes expresiones, luego de ejecutarse las instrucciones del segmento anterior:

```
c1==c3
p1.equals(p2)
p1.equals(p3)
p1.obtenerColor() == p2.obtenerColor()
p1.obtenerColor() == p3.obtenerColor()
```