Universidad Autónoma Metropolitana

Unidad Iztapalapa

UEA: Análisis y diseño de sistemas de cómputo

Profesor: Orlando Muñoz Texzocotetla 

Equipo: UAMITOS

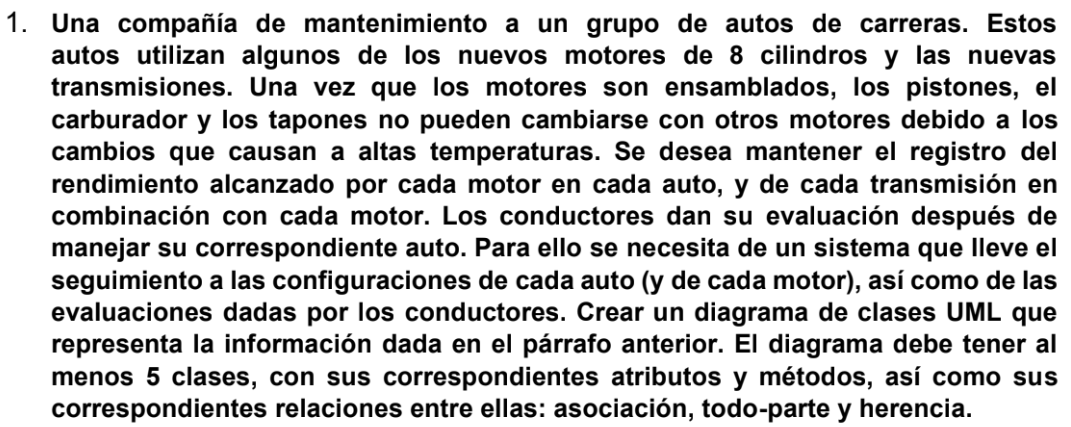
Integrantes:

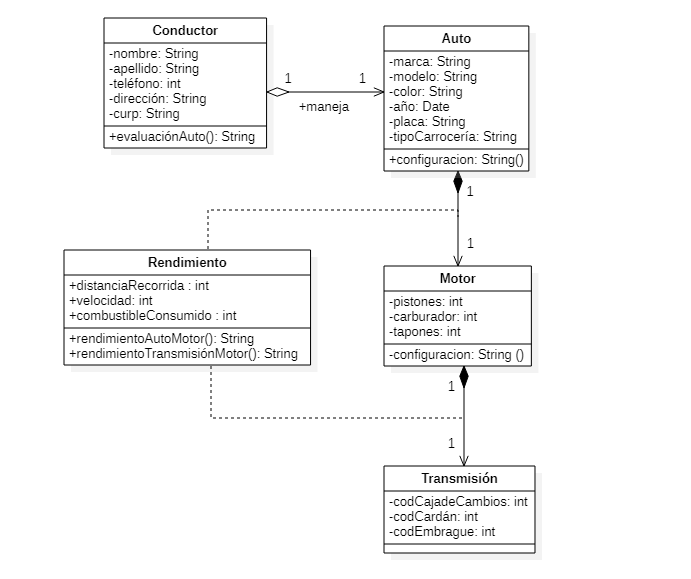
* Hernández Enriquez Geovanny 2173010361
* López de la Rosa Luis Fernando 2183010837
* Lovera Reyes José Rodrigo 2183052237
* Ramos Ramírez Antonio 2183011512

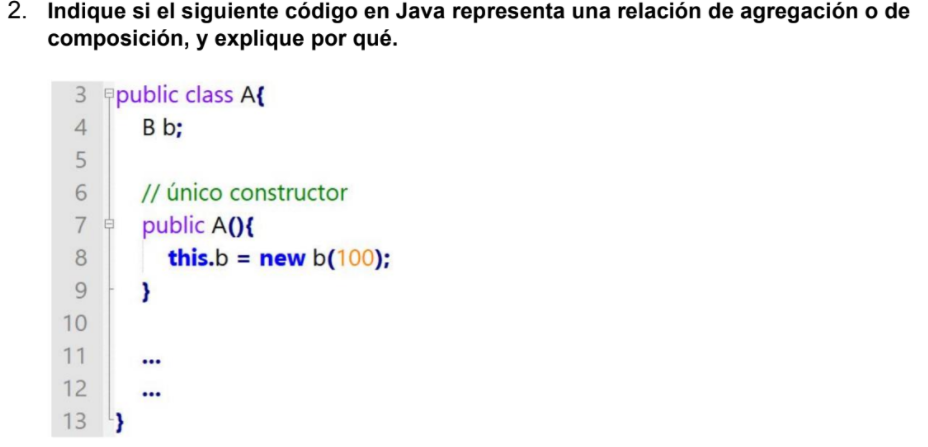
Trimestre 21-O

**Práctica en equipo**

**(Relación entre clases)**

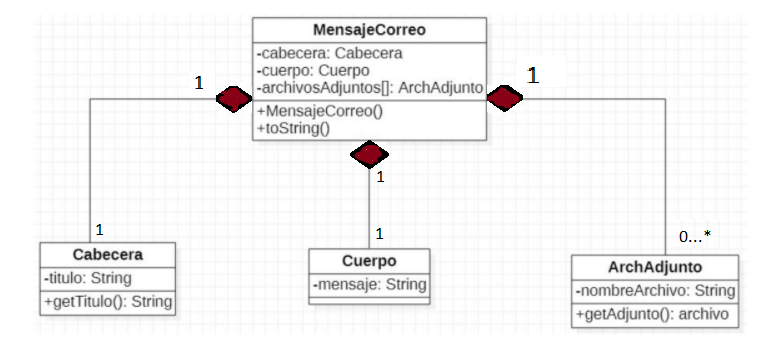


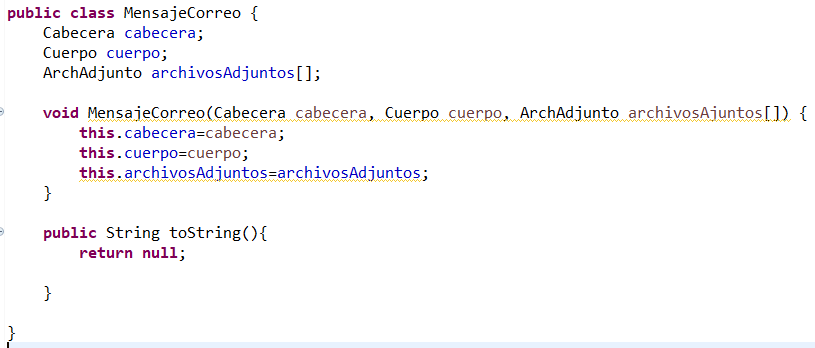


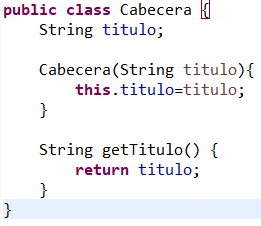


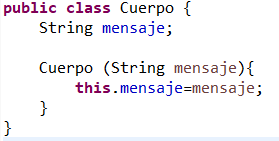
El código representa una relación de composición, por qué cada vez que se llegue a crear un nuevo objeto de la clase A, la variable de b de la clase A siempre se va a inicializar a 100 y presenta una relación de un todo.

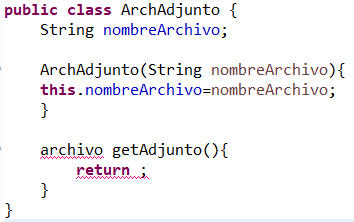
**3. Basados en el siguiente diagrama de clases, indique cuáles son las relaciones que se indican con un signo de interrogación, y escriba un programa en Java que implemente dichas relaciones.**

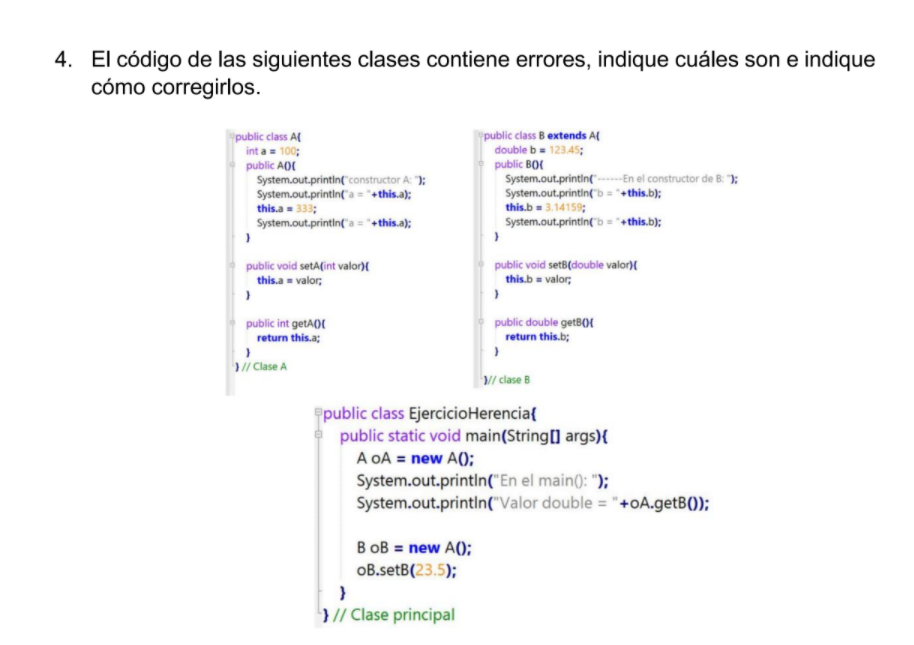












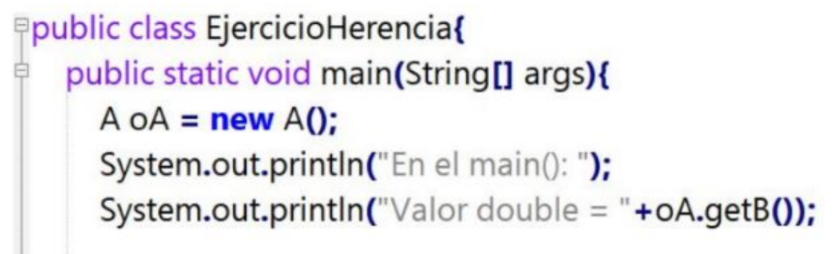
Hay errores en la Clase B y en la clase Ejercicio Herencia la clase A nos sirve porque hay varios errores que se derivan de ella.

En la clase b

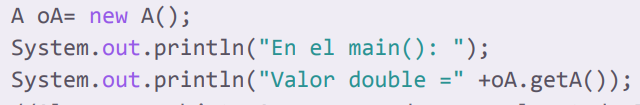


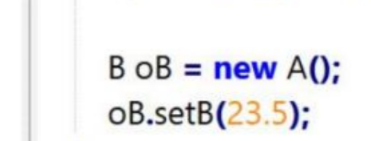
no tiene sentido puesto que no se hereda nada de la clase A, podríamos quitar el extends A.

En la clase Ejercicio de Herencia.



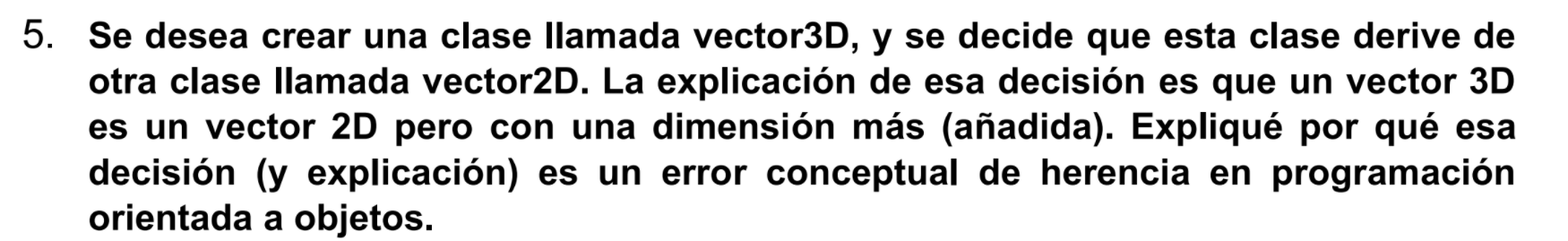
No podemos usar oA.getB porque estamos creando un objeto de A y usando un getB. Lo correcto sería.





En la primera línea se pone objeto tipo B pero se intenta crear un objeto tipo A.

En la segunda línea se usa setB, pero nunca se uso getB por lo tanto no verá la información ingresada.

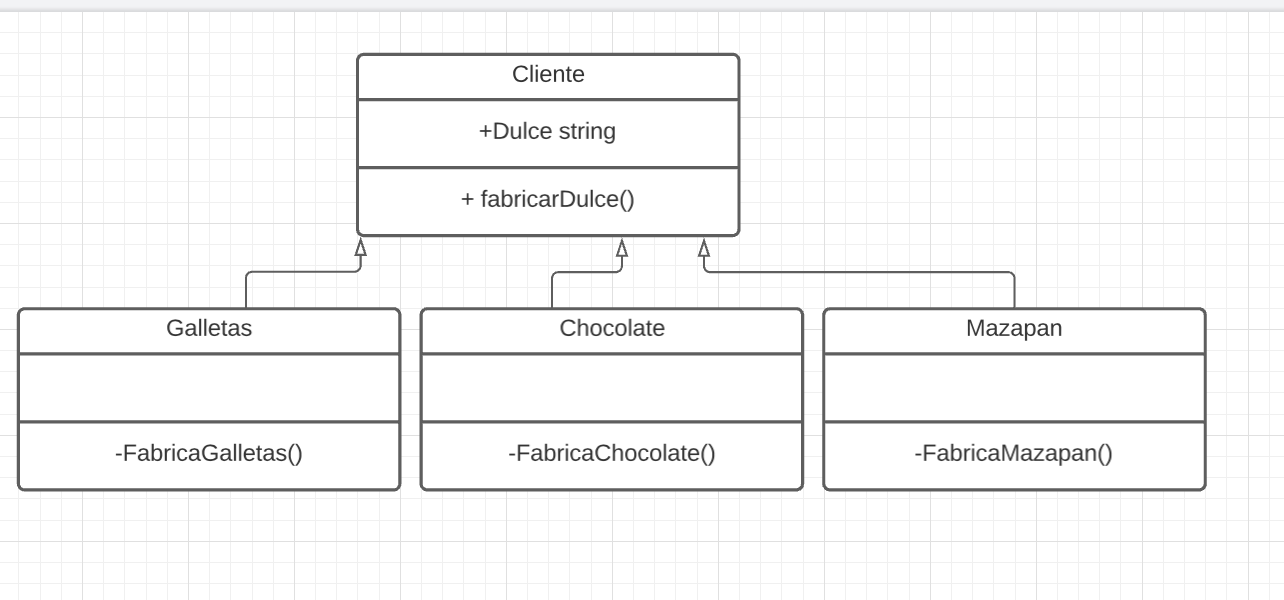


Esto es incorrecto pues no se está usando la herencia para ir de algo general a algo específico, en este ejemplo una forma correcta de usar herencia sería tener la superclase vector para que vector2d y vector3d fueran subclases de esta, pues al usar vector2d como superclase de vector3d estamos usando dos conceptos que no tienen una relación de jerarquía entre ellos, más bien estos dos se encuentran al mismo nivel.

**6. El siguiente diagrama de clases contiene tres clases que fabrican un tipo particular de dulce: galleta, chocolate y mazapán. Se desea implementar UN SOLO método llamado fabricarDulce(¿Tipo? x). Este método recibirá un objeto x (ya sea una fábrica de galleta, de chocolate o de mazapán), el cual, creará el dulce correspondiente.**

**a. Complete el diagrama UML para que por medio de la herencia y la asociación sea posible que el método fabricarDulce reciba un solo objeto x, el cual, pueda fabricar cualquiera de los tres tipos de dulce dependiendo del objeto x recibido.**

**b. Implemente el método fabricarDulce(...).**



**public** **class** Dulce {

**private** String dulces;

**public** **void** setfabricarDulce(String n) {

dulces=n;

}

**public** String getfabricarDulce() {

**return** dulces;

}

}