Nombre y Apellido: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Cantidad de hojas entregadas: \_\_ Nota:\_\_\_\_\_\_\_\_

|  |
| --- |
| **Lea atentamente TODO el enunciado antes de empezar a desarrollar, recodar generar todas las firmas de las funciones y las especificaciones correspondientes para que el proyecto compile antes de programar la solución** |

1) Se desea implementar una clase Vehiculo que tenga como atributo un conjunto de 4 Ruedas, un conjunto de 4 Ventanas y un solo motor. Además el Vehiculo, cualquiera sea debe poder soportar el método void anvanzar() y void frenar().  
Luego en una ampliación del programa, se agregar vehículos que cobren por el servicio que brindan. En ese caso los vehículos que cobren por el servicio que brindan deberán tener el método cobrar() (Ejemplo: Subte, Colectivo, Taxi). Quienes cobran también deberán tener un método que sea pagar(). Para no tener problemas de diseño se decidió utilizar el principio de segregación de la interfaz y generar dos interfaces diferentes Imponible y Tarifable (para los casos de pagar y cobrar respectivamente).

Recordar que Uber solo puede cobrar y no pagar.

a) Implementar la clase Vehiculo, con los atributos mencionados y los correspondientes getter junto a la implementación de avanzar() y frenar() vacía.  
PISTA: Los conjuntos son objetos de la clase ArrayList. Los nombres de los getters son getRuedas, getVentanas, getMotor.  
  
b) Implementar las subclases de Vehiculo Colectivo, Taxi y Uber, con únicamente los métodos que les corresponder implementar a cada uno. El mismo programa debe hacer implementar las funciones correspondientes a cada subclase. Es decir, el programa debe hacer implementar cobrar y pagar a todos, menos a Uber.

Las firmas de las funciones cobrar y pagar son   
public double cobrar(double mrecorridos)  
public double pagar(double mrecorridos)

Donde mrecorridos es una medida de lo que se recorrió y entra por parametro.  
  
Como se calculan los métodos según la clase  
Taxi:  
 cobrar: 5\*mrecorridos^2 – 2\*mrecorridos + 50;  
 pagar: 2\*mrecorridos

Colectivo:  
 cobrar: 18 si mrecorridos ∃[0;200] ; 20 si mrecorridos ∃[201;1000]; 21 si m recorridos ∃[1001;+∞]   
 pagar: 5000 (Este valor es independiente del parametro, aunque lo reciba)

Uber:  
 cobrar: 6\*mrecorridos

2) Se desea implementar una clase Animal. Junto con con distintas sublcases Leon, Perro, Gato.

Cada una de ellas emitirá un sonido distinto. Ninguna clase que prentenda ser Animal podrá evitar reproducir un sonido, aunque de ninguna manera de debe poder instanciar un animal.

Además no existen animales que no tengan un tipo de animal definido. Codificar el ejemplo recién propuesto entendiendo por hacer sonido: return “Sonido”

En el caso de perro devolver “Guau”

En el caso de gato devolver “Miau”

En el caso de leon devolver “Groar”

El metodo polimorfico debe llamarse hacerSonido();