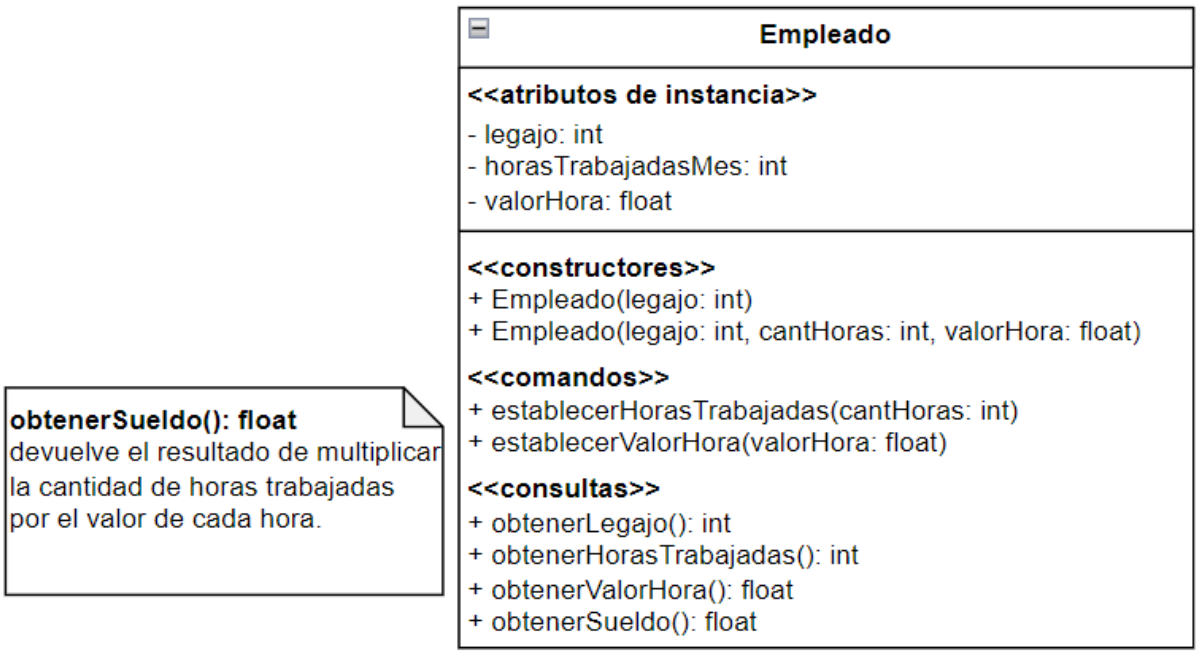
Trabajo práctico N°4

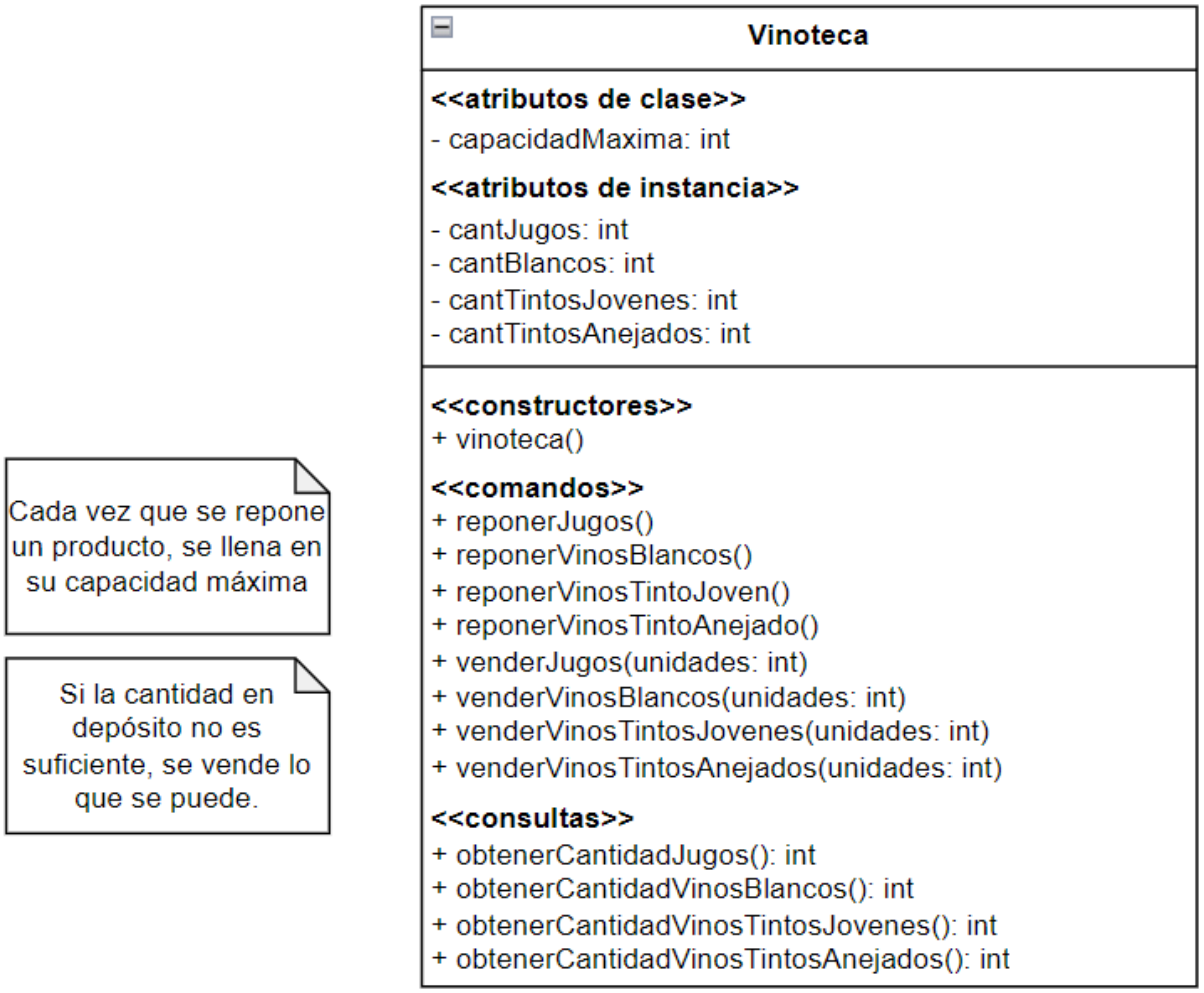
1. Analice si las siguientes definiciones son correctas
    a. Desde el punto de vista del diseño de un sistema orientado a objetos, una clase es un “molde” que establece solamente los atributos de una entidad.
    b. En la Programación Orientada a Objetos se busca tratar a toda entidad como si fuera un objeto en la vida real, donde cada objeto tiene algunas propiedades y funcionalidades propias.
    c. Los objetos tienen atributos (características) y métodos (funciones) asociados.
    d. Un constructor es un método que se invoca cuando se crea un objeto.
    e. Un comando es un método que modifica los valores del objeto.
    f. Una consulta es un método que no modifica los valores del objeto.
2. Una organización desea mantener información básica sobre sus empleados para calcular los sueldos, para ello diseñó la siguiente clase:



    a. implementar en python la clase Empleado.
    b. Escribir una clase tester para verificar los servicios de la clase Empleado utilizando el constructor con todos los parámetros. Esta verificación debe permitir al usuario ingresar los datos de un empleado (legajo, cantidad de horas trabajadas en el mes, y valor de la hora) y luego mostrar su legajo y el sueldo calculado.

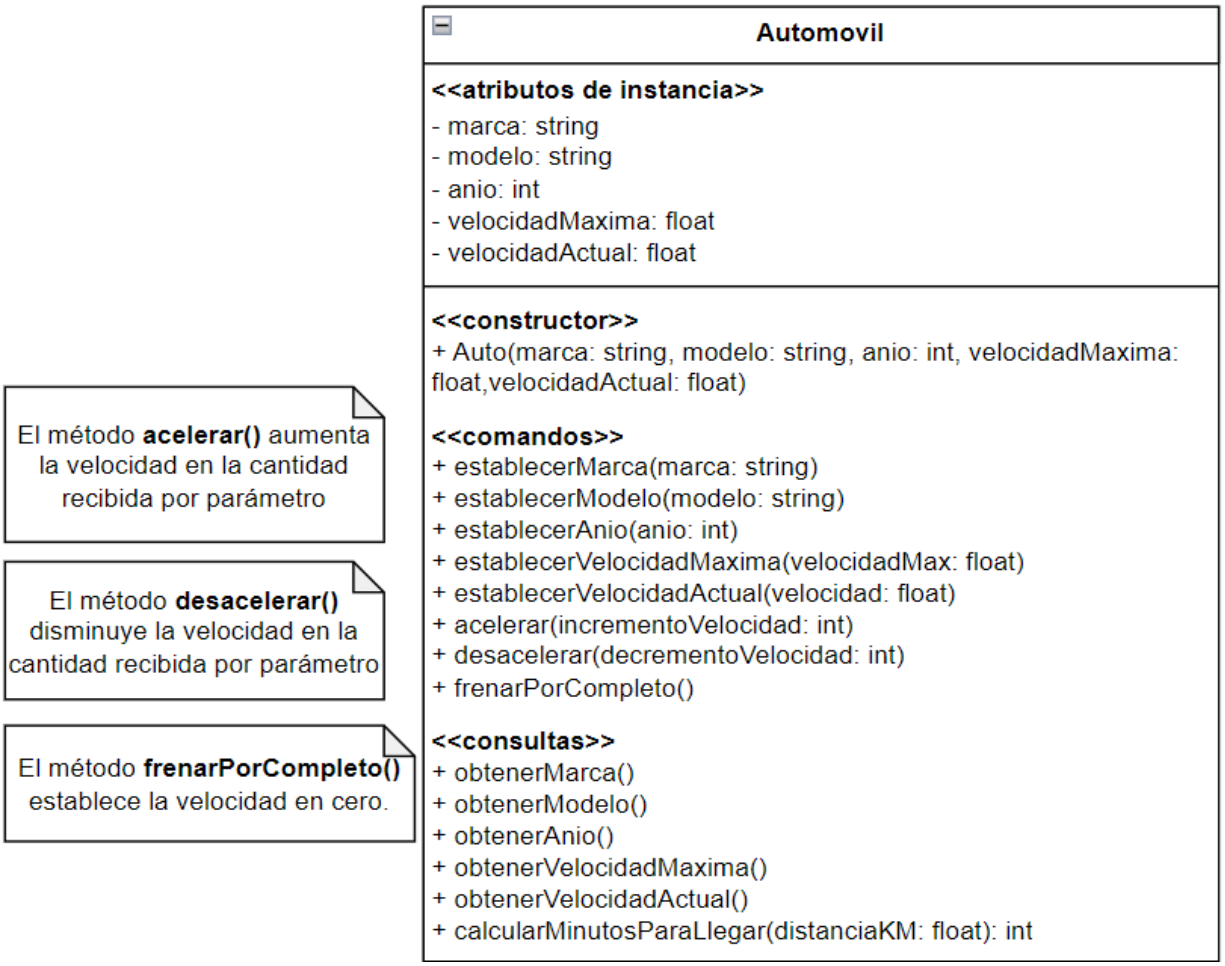
c. Escribir otra clase tester para verificar los servicios de la clase Empleado. Esta verificación debe permitir al usuario ingresar los datos de un empleado (legajo, cantidad de horas trabajadas en el mes, y valor de la hora), crear el objeto empleado utilizando el constructor con el parámetro legajo, y luego modificar los demás atributos del objeto con los servicios provistos por la clase. Finalmente debe mostrar por pantalla el legajo y el sueldo del empleado.

3. Una vinoteca tiene su depósito dividido en cuatro grandes secciones: Jugos sin alcohol, Vinos blancos, Vinos tintos jóvenes y Vinos tintos añejados. Cada sección del depósito puede almacenar un máximo de 5000 unidades de su respectivo producto. Al crearse la vinoteca, cada sección comienza con su capacidad máxima. La vinoteca puede vender productos de cualquiera de las cuatro secciones, lo que reduce la cantidad disponible en esa sección. Si la cantidad solicitada es mayor a la disponible en una sección, se vende lo que se pueda y se informa que no se pudo completar la venta. Además, es posible reponer la cantidad de un producto en una sección hasta alcanzar su capacidad máxima. Cada vinoteca puede modelarse con el siguiente diagrama:



a. implementar en python la clase vinoteca
b. implementar una clase tester que verifique los servicios de la clase Vinoteca con valores significativos.

4. Se requiere un programa que modele el concepto de un Automóvil. Para simplificar consideraremos que un automóvil tiene solamente los siguientes atributos:
Marca: el nombre del fabricante.
Modelo: nombre del modelo.
Año: año de fabricación.
Velocidad máxima: velocidad máxima soportada por el vehículo en km/h.
Velocidad actual: velocidad del vehículo en un momento dado en km/h.
El siguiente diagrama modela el Automóvil:



Es importante tener en cuenta que no se debe acelerar más allá de la velocidad máxima permitida para el automóvil. De igual manera, tampoco es posible desacelerar a una velocidad negativa. Si se cumplen estos casos, se debe establecer la velocidad en el límite correspondiente y mostrar por pantalla un mensaje.
El tiempo estimado en horas se calcula como el cociente entre la distancia en

kilómetros a recorrer y la velocidad actual (en km/h), multiplicando este valor por 60 obtenemos la cantidad de minutos. Considerar el caso especial cuando el auto se encuentre detenido, en este caso mostrar un mensaje indicando que “el auto está detenido y no se puede calcular el tiempo para llegar”.

a. implementar la clase en python
b. implementar una clase tester que cree un objeto de clase Automovil, y que pida al usuario la cantidad de iteraciones a realizar, luego para cada iteración deberá generar un número aleatorio entre 0 y 3 que determinará la operación a realizar:
    - 0: acelerar(20)
    - 1: desacelerar(15)
    - 2: frenarPorCompleto()
    - 3: calcularMinutosParaLlegar(50)

    En cada iteracion se debe mostrar el valor que se modificará, la acción a realizar y el valor modificado luego de la acción. Por ejemplo, si tocó el valor 0 muestra: “velocidad = 100, acelerar, velocidad actual = 120”. En el caso del calculo del tiempo para llegar debe mostrar el tiempo para llegar.

c. Realizar otra clase tester con la misma lógica, pero las operaciones a realizar deben recibir parámetros aleatorios (coherentes).

5. El tamagotchi es un juego donde el objetivo es mantener con vida a una “mascota virtual” la mayor cantidad de tiempo posible. Cada mascota virtual se identifica por medio de su nombre. Pueden realizar diferentes acciones, las que aumentan o disminuyen su nivel de energía, el cual nunca puede disminuir por debajo de 0 ni superar el máximo de 100. Mientras duerme no puede hacer ninguna otra acción, a menos que antes se lo despierte. Ninguna mascota puede realizar más de tres acciones de desgaste de energía en forma consecutiva. Al cuarto intento, ignora la orden dada y directamente se pone a dormir. Cuando se vuelva a despertar, podrá volver a realizar a lo sumo tres acciones de desgaste. La mascota tiene tres estados que debes gestionar: Energía, Diversión e Higiene. Cada estado se representa con un valor numérico que puede variar entre 0 (muy bajo) y 100 (muy alto). Al crear la mascota, todos los estados se inicializan en su

máximo valor posible. La mascota tiene varias acciones que pueden influir en estos estados:

a. Comer: aumenta el nivel de energía en 20 puntos.
b. Beber: aumenta el nivel de energía en 10 puntos.
c. Jugar: aumenta la diversión en 40 puntos, pero reduce la energía en 20 puntos y la higiene en 15 puntos.
d. Caminar: aumenta la diversión en 20 puntos, reduce energía en 10 puntos y la higiene en 8 puntos.
e. Saltar: aumenta la diversión en 10 puntos, reduce energía en 15 puntos y la higiene en 10 puntos.
f. Dormir: aumenta la energía en 20 puntos, pero reduce la diversión en 10 puntos.
g. Bañar: aumenta la higiene en 40 puntos, pero reduce la diversión en 10 puntos.

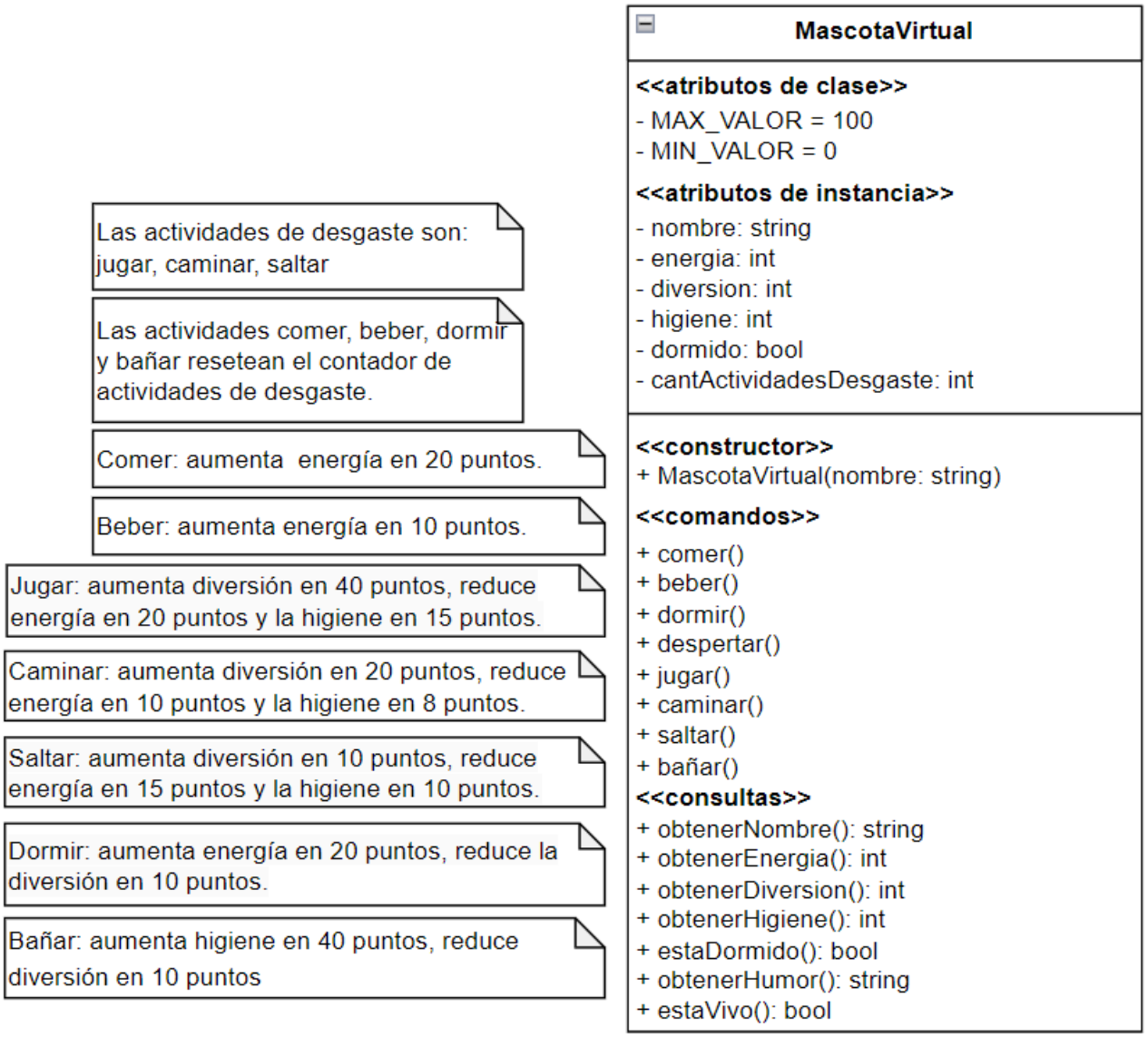
Cada estado tiene un límite máximo de 100 y un mínimo de 0, por lo que si al realizar una acción se supera o se disminuye ese límite, el valor se ajusta al máximo o mínimo correspondiente.

Si el nivel de energía de la mascota llega a 0, lamentablemente dejará de existir y no podrá realizar ninguna actividad más.

El humor de la mascota está determinado por la combinación de los valores de energía, diversión e higiene de la siguiente forma:

- Humor Feliz: Si energía, diversión e higiene están todos por encima de 70.
- Humor Alegre: Si al menos dos de los tres (energía, diversión, higiene) están entre 50 y 70.
- Humor Neutral: Si al menos dos de los tres están entre 30 y 50.
- Humor Triste: Si al menos dos de los tres están entre 10 y 30.
- Humor Muy Triste: Si al menos dos de los tres están por debajo de 10.

Implementar la clase MascotaVirtual modelada por el siguiente diagrama, pueden agregarle los métodos que consideren necesarios.


MascotaVirtual
<<atributos de clase>>
- MAX_VALOR = 100
- MIN_VALOR = 0
<<atributos de instancia>>
- nombre: string
- energia: int
- diversion: int
- higiene: int
- dormido: bool
- cantActividadesDesgaste: int
<<constructor>>
+ MascotaVirtual(nombre: string)
<<comandos>>
+ comer()
+ beber()
+ dormir()
+ despertar()
+ jugar()
+ caminar()
+ saltar()
+ bañar()
<<consultas>>
+ obtenerNombre(): string
+ obtenerEnergia(): int
+ obtenerDiversion(): int
+ obtenerHigiene(): int
+ estaDormido(): bool
+ obtenerHumor(): string
+ estaVivo(): bool
Las actividades de desgaste son: jugar, caminar, saltar
Las actividades comer, beber, dormir y bañar resetean el contador de actividades de desgaste.
Comer: aumenta energía en 20 puntos.
Beber: aumenta energía en 10 puntos.
Jugar: aumenta diversión en 40 puntos, reduce energía en 20 puntos y la higiene en 15 puntos.
Caminar: aumenta diversión en 20 puntos, reduce energía en 10 puntos y la higiene en 8 puntos.
Saltar: aumenta diversión en 10 puntos, reduce energía en 15 puntos y la higiene en 10 puntos.
Dormir: aumenta energía en 20 puntos, reduce la diversión en 10 puntos.
Bañar: aumenta higiene en 40 puntos, reduce diversión en 10 puntos