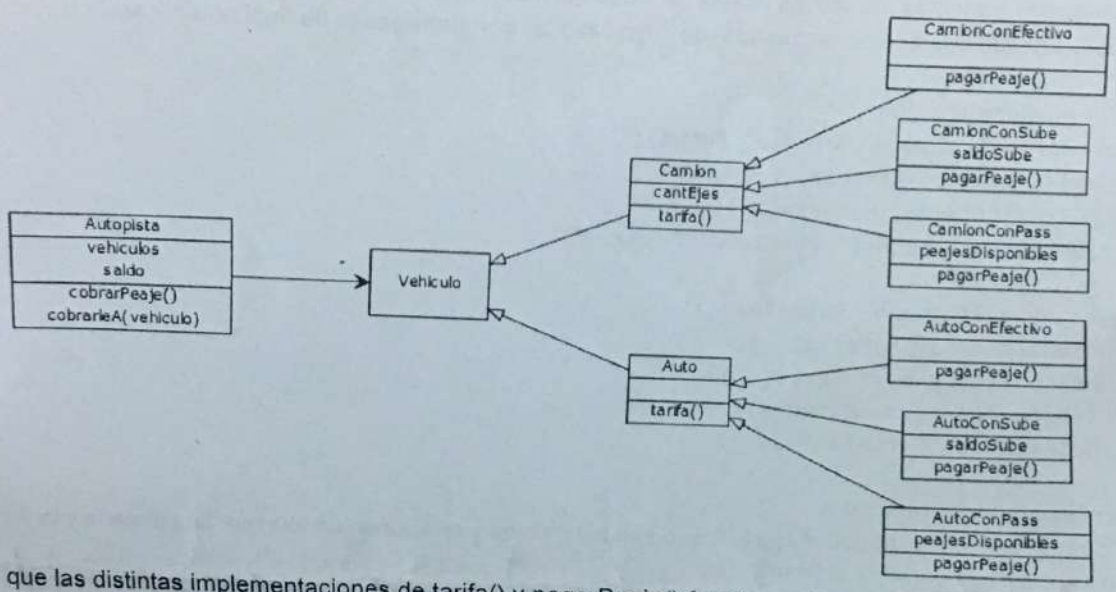
Paradigmas de Programación **Examen Final**

Parte C

07/12/2019

Se sabe que en una autopista se cobra peaje a los vehículos que la usan. Los autos y camiones tienen diferentes formas de calcular la tarifa (en el auto es un valor único de $60 y en los camiones es $20 por cada eje), y además hay vehículos que pagan con telepeaje "Pass", otros con efectivo y otros con la tarjeta Sube. Al efectuarse el cobro suceden dos cosas: el saldo de la empresa que mantiene la autopista aumenta, y además cada vehículo efectúa el pago. Los que pagan con Sube disminuyen el saldo de su tarjeta en un 10% más por gasto de servicio (el 10% no va para la autopista), y los que pagan por telepeaje disminuyen en 1 la cantidad de peajes disponibles. No necesitamos registrar nada adicional para los que pagan en efectivo.

Se tiene el siguiente diagrama de la solución propuesta:



Suponiendo que las distintas implementaciones de tarifa() y pagarPeaje() funcionan de acuerdo al requerimiento indicado y además esto es parte del código del sistema:

```
class Autopista {
  const property vehiculos = []
  var property saldo = 0
  method cobrarPeaje(){
       vehiculos.forEach { v => self.cobrarleA(v) }
  }
  method cobrarleA(vehiculo){
     saldo = saldo + vehiculo.tarifa()
     vehiculo.pagarPeaje(saldo)
  }
}
```

1. Indicar verdadero o falso y justificar en cada caso.
   a. Para que la solución propuesta funcione, es necesario que exista la clase Vehículo, ya que de lo contrario no podrían incluirse los distintos vehículos en la lista de la autopista.
   b. Hay buen uso de polimorfismo en la solución.
   c. Por cómo se planteó el modelo, se va a repetir lógica en las implementaciones de pagarPeaje().
   d. Si un auto que paga con Sube decide pasarse al sistema Pass de telepeaje va a poder hacerlo sin problema con este modelo.
   e. Al cobrarle a un vehículo que paga con Pass y este se queda sin disponibles y lanza un error, se frena la ejecución y el estado del sistema queda coherente.
2. Desarrollar una nueva solución (con código y diagrama) que solucione los problemas encontrados, incluyendo el código de los vehículos.
3. ¿Qué haría falta hacer en cada solución para poder contemplar otros tipos de vehículos, como ser las motos?

Paradigmas de Programación **Examen Final** **07/12/2019**

Condiciones de aprobación

Para aprobar es necesario simultáneamente: • completar el 60% del examen, y • obtener al menos la mitad de los puntos **en cada paradigma**.	**En todas tus respuestas sé puntual**, no pierdas el foco de lo que se pregunta. Respuestas en exceso generales son tan malas como respuestas incompletas. 

Parte A

Necesitamos hacer un sistema para que los alumnos de la facultad se inscriban a los finales de una materia. Sabemos que un alumno se puede anotar a un final cuando ya aprobó la cursada de la materia en cuestión, no aprobó el final de esa materia todavía, y además aprobó los finales de todas las materias correlativas, a menos que haya pasado menos de un año lectivo desde que aprobó la cursada, en cuyo caso las correlatividades de final no aplican.

Se propuso la siguiente solución:

```
puedeAnotarseAFinal(Alumno, Materia, Fecha):-
  aproboCursada(Alumno, Materia, FechaFirma),
  not(aproboFinal(Alumno, Materia, _)),
  añosLectivosTranscurridos(FechaFirma, Fecha, 0).

puedeAnotarseAFinal(Alumno, Materia, _):-
  aproboCursada(Alumno, Materia, _),
  not(aproboFinal(Alumno, Materia, _)),
  correlativa(Correlativa, Materia),
  aproboFinal(Alumno, Correlativa, _).
```

Sabiendo que existen los predicados:

- `aproboCursada/3` y `aproboFinal/3` que son inversibles y relacionan un alumno, una materia y la fecha de aprobación de la cursada o final;
- `correlativa/2` que relaciona dos materias tal que la primera es correlativa de la otra, y es inversible;
- `añosLectivosTranscurridos/3` que relaciona dos fechas cualesquiera con los años lectivos que transcurrieron entre ambas fechas, y sólo es inversible para su tercer parámetro.

1) La solución propuesta, ¿cumple con la lógica pedida? Justificar y plantear algún ejemplo que sirva para fundamentar esa respuesta.
2) Analizar la inversibilidad de `puedeAnotarseAFinal/3`. En caso de que no sea inversible para uno o más parámetros, explicar qué sería necesario modificar o agregar para que lo sea.
3) Realizar cualquier arreglo que crea conveniente sobre `puedeAnotarseAFinal/3` en base a las respuestas anteriores. Corregir también la repetición de lógica existente.

Parte B

Se desea modelar en el paradigma funcional un sistema de admisión de locales nocturnos. Existen personas que llegan a la puerta de un local que tiene una serie de requisitos, de manera que no se deja pasar a los que incumplan alguno de ellos. Puede haber diversos requisitos, como por ejemplo que sólo entren personas que sean de una cierta nacionalidad, que no se deje entrar a los que tienen menos de cierta edad, o que no permita pasar personas que lleven puesta alguna prenda de vestir de las que se consideran inadecuadas. Podría haber locales que tengan más requisitos que otros, que no tengan ninguno, o que tengan requisitos similares, por ejemplo, que restrinjan el acceso según la vestimenta, pero con diferentes conjuntos de vestimentas inadecuadas. Por lo tanto, de las personas se debe conocer la edad, la nacionalidad y las vestimentas que lleva puestas.

Se pide:

1. Definir tipos de datos y funciones (explicitando el tipo de todas ellas) para cubrir las necesidades explicadas.
2. Mostrar cómo se representa un local de ejemplo que tenga tres requisitos como los mencionadas anteriormente.
3. Desarrollar la función patovica, que permita saber qué personas de la fila que espera en la puerta, pueden entrar al local.
4. Indicar dónde y para qué se utilizaron los siguientes conceptos: composición, aplicación parcial y orden superior.