Entregable 1

Semana 3 Evaluación Sumativa Grupal

Grupo de trabajo 6
Erick Caro Hidalgo
Juan Pablo García
Felipe Herrera Lagos
Jean Piffaut Ramírez
Jonathan Reyes Rebolledo

Profesor Guía:

Tomas Sepúlveda Caroca

Programación Avanzada NRC: 2005

1. *Índice.*

Contenido

1. ÍNDICE. 2
2. INTRODUCCIÓN. 3
3. DESCRIPCIÓN DEL PROBLEMA (ENUNCIADO DEL CASO). 3
4. ANÁLISIS Y DISEÑO DE LA SOLUCIÓN. 4
5. CONCLUSIONES. 6
6. BIBLIOGRAFÍA. 6

2. *Introducción.*

En este trabajo abordaremos la construcción de un programa aplicando los fundamentos de la programación orientado al objeto a través del diseño de un diagrama de clases UML, para dar solución a problemas computacionales simples.

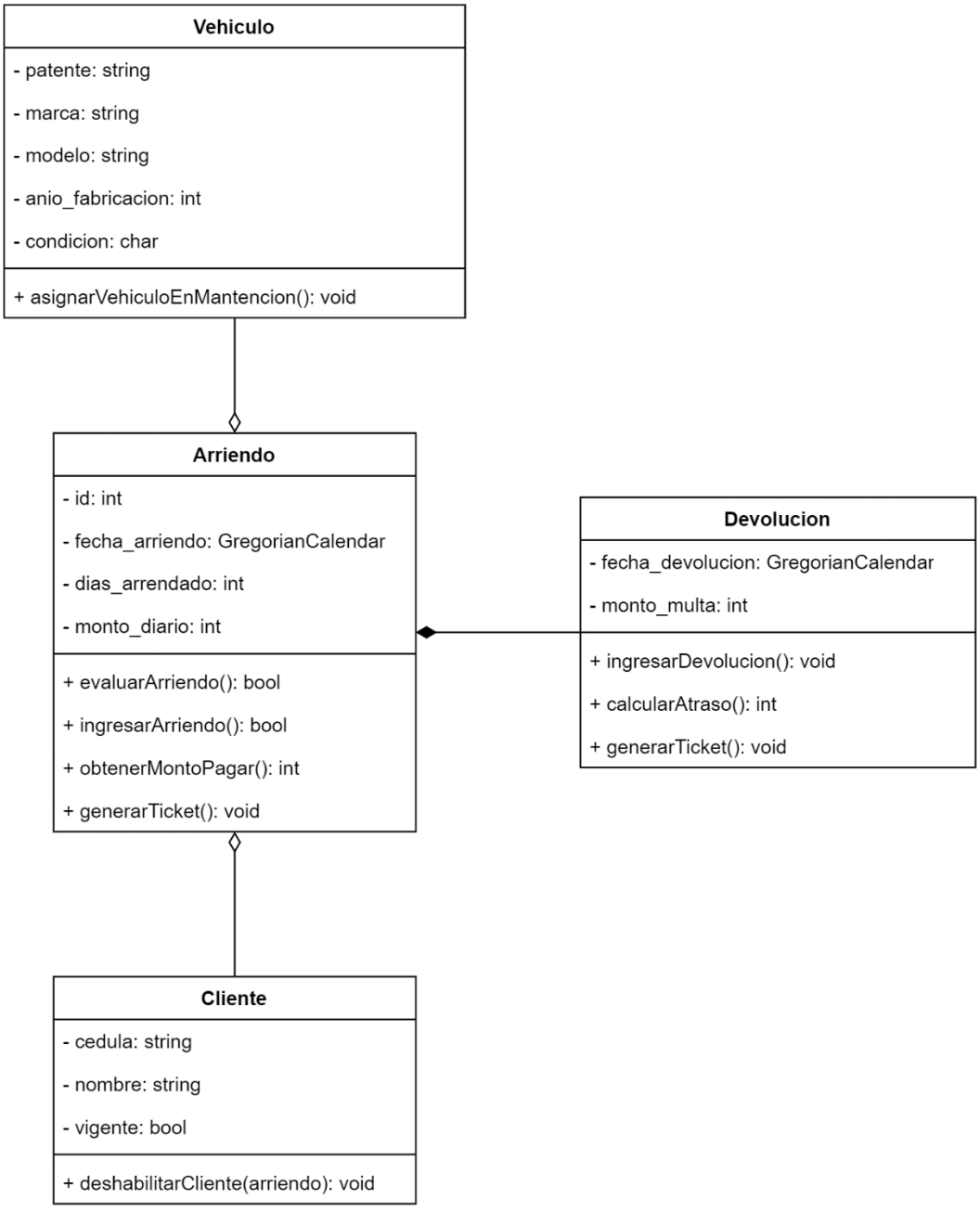
En particular, centraremos los esfuerzos en resolver las gestiones cotidianas que debería cursar un programa de gestión de un “Car-Rent”.

3. *Descripción del problema (enunciado del caso).*

Buscamos dar una solución a empresa de arriendo de vehículos “Car-Rent”, la que necesita implementar un sistema informático que permita mejorar la gestión de la empresa. El gerente espera que podamos diseñar y programar una solución que administre y almacene la información asociada a sus automóviles, clientes, arriendos y devoluciones. El requisito fundamental es utilizar la programación orientada a objetos y sus buenas prácticas. La especificación de lo que el gerente requiere se detalla a continuación:

I. Individualización de vehículos y sus características
II. Individualización de cliente
III. Cálculo de tarifas según los servicios solicitados
IV. Gestiones de arriendo y devolución
V. Deshabilitar clientes
VI. Asignación de vehículos
VII. Etc.

4. *Análisis y diseño de la solución.*



VEHÍCULO

asignarVehiculoEnMantencion(): Asigna el vehículo a la condición de mantención e imprime el estado del vehículo.

CLIENTE

deshabilitarCliente(): Deshabilita al cliente dependiendo de si el arriendo enviado por parámetro no está activo

ARRIENDO

generarTicket(): Imprime un ticket el cual contiene toda la información del arriendo generado

evaluarArriendo(): Valida que el cliente que arrendara siga vigente. Y valida que el vehículo a arrendarse este disponible.

ingresarArriendo(): Valida que el arriendo sea válido con el método evaluarArriendo. Si es válido cambia la condición del vehículo a Arrendado. Genera el ticket con toda la información del arriendo.

obtenerMontoPagar(): Retorna el monto a pagar el cual se calcula en base al monto y los días

DEVOLUCIÓN

ingresarDevolucion() Valida si el cliente y que el arriendo sean vigentes. Calcula en base a la fecha de arriendo y la fecha de devolución si es que se debe asignar una multa al cliente. Ingresa la devolución al sistema

calcularAtraso():Calcula y retorna en base a la fecha de arriendo y devolución la diferencia de días

generarTicket(): Imprime un ticket el cual contiene toda la información de la devolución generado

5. Conclusiones.

Nuestro grupo determinó de forma unánime que las inhabilitaciones de clientes de manera arbitraria por el gerente significan un riesgo para la integridad de los datos. Por tanto, se implementó que previo a una inhabilitación se debe verificar que no existan arriendos activos asociados.

En búsqueda de la mejora continua y correcto funcionamiento del sistema, hemos incorporado validaciones adicionales – por cierto, perfectibles - que nos permitirán robustecer la lógica, verificando que previo a una devolución se revise los días de arriendo pactados contra la fecha de devolución, para con ello determinar eventuales multas en el proceso de retorno del vehículo.

6. Bibliografía.

Joyanes, L. y Zahonero, I. (2011). *Programación en Java. Algoritmos, programación orientada a objetos e interfaz gráfica de usuario*. Capítulo 2, (pp. 32-43). 1era Edición. McGRAW-HILL. Interamericana Editores, S.A. México. Programación en Java. Algoritmos, programación orientada a objetos e interfaz gráfica de usuario.

McClane, J. (17 de 04 de 2018). *Stackoverflow*. Obtenido de Stackoverflow: https://es.stackoverflow.com/questions/118104/validacion-de-rut-en-java