Sistema de Gestión de Reservas de Vehículos

Santy Ramirez

7 de octubre de 2024

1. Estimación

El tiempo estimado para la realización de este examen es de 2 horas con 30 minutos.

2. Enunciado

Estás a cargo de desarrollar un sistema de gestión de reservas para una compañía de alquiler de vehículos que tiene una flota diversa. El sistema debe permitir a los clientes consultar y reservar vehículos, gestionar las reservas y calcular el costo total basado en varios factores. Además, debe permitir al administrador de la flota añadir nuevos vehículos y verificar su disponibilidad. Mucha suerte muchachos, demuestren lo que han aprendido :)

3. Requerimientos Funcionales

3.1. Clases Principales

Debes crear las siguientes clases base:

- **Vehiculo**: Esta clase será la base para todos los vehículos en la flota.
- **Reserva**: Representa una reserva de un vehículo por un cliente.
- **Cliente**: Contendrá la información del cliente.
- **Administrador**: Gestionará la flota y las reservas.

3.2. Características de la Clase Vehículo

Cada vehículo debe tener los siguientes atributos:

- **idVehiculo**: Identificador único del vehículo.
- **marca**: Marca del vehículo (e.g., "Toyota").
- **modelo**: Modelo del vehículo (e.g., Çorolla").

- **año**: Año del vehículo.
- **costoDiario**: Costo de alquiler por día.
- **disponibilidad**: Indica si el vehículo está disponible para alquilar o no.

Método:

- `public double calcularPrecio(int dias, boolean seguro, boolean gps)`: Calcula el costo total del alquiler basado en el número de días y la inclusión opcional de seguro y GPS. Los costos adicionales son:
  - Seguro: 10 % adicional al costo diario.
  - GPS: Cargo fijo de $5 por día.

3.3. Subclases de Vehículo

Debes crear subclases para diferentes tipos de vehículos con características adicionales:

- **Auto**:
  - **tipoCombustible**: Gasolina, Diésel o Eléctrico.
- **Moto**:
  - **cilindrada**: La cilindrada de la moto en cc.
- **Camioneta**:
  - **capacidadCarga**: Capacidad de carga en kilogramos.
- **Autobús**:
  - **capacidadPasajeros**: Número de pasajeros que puede transportar.

3.4. Clase Reserva

La clase **Reserva** debe tener los siguientes atributos:

- **idReserva**: Identificador único de la reserva.
- **cliente**: Instancia de la clase **Cliente**.
- **vehiculo**: Instancia de la clase **Vehiculo**.
- **fechaInicio**: Fecha de inicio del alquiler (utiliza la clase `LocalDate` de Java).
- **fechaFin**: Fecha de finalización del alquiler.
- **costoTotal**: Costo total de la reserva (calculado con el método `calcularPrecio` de la clase **Vehiculo**).

Método:

- `public void confirmarReserva()`: Marca el vehículo como reservado durante las fechas indicadas y calcula el costo total.

3.5. Clase Cliente

Atributos:

- **idCliente**: Identificador único del cliente.
- **nombre**: Nombre del cliente.
- **reservas**: Lista de instancias de la clase **Reserva** asociadas al cliente.

Método:

- `public void reservarVehiculo(Vehiculo vehiculo, LocalDate fechaInicio, LocalDate fechaFin, boolean seguro, boolean gps)`: Crea una reserva para el cliente si el vehículo está disponible y lo añade a la lista de reservas.

3.6. Clase Administrador

Métodos:

- `public void añadirVehiculo(Vehiculo vehiculo)`: Para agregar un nuevo vehículo a la flota.
- `public boolean verificarDisponibilidad(Vehiculo vehiculo, LocalDate fechaInicio, LocalDate fechaFin)`: Para verificar si un vehículo está disponible en un rango de fechas.
- `public List<Vehiculo>listarVehiculosDisponibles()`: Para listar todos los vehículos disponibles.

4. Reglas Adicionales

- Los vehículos solo pueden ser reservados si están disponibles durante el período solicitado.
- Un cliente no puede realizar otra reserva si ya tiene una activa.
- El sistema debe manejar excepciones, como intentos de reservar un vehículo no disponible o crear reservas con fechas incorrectas.

5. Ejemplo de Uso del Sistema

- Un cliente intenta reservar una camioneta por 5 días, con GPS y sin seguro. El sistema calcula el costo, verifica la disponibilidad y confirma la reserva.
- Un administrador añade un nuevo auto a la flota y revisa su disponibilidad.

6. Gestión del Proyecto con Git

El proyecto debe gestionarse utilizando Git, siguiendo estas directrices:

6.1. Subida del Proyecto a GitHub

- Cada estudiante debe crear un repositorio en su cuenta de GitHub.
- El repositorio debe ser privado y compartir acceso con el profesor.

6.2. Estructura de Ramas en Git

- **master**: Contendrá la versión final y estable del código.
- **develop**: Integrará las características del proyecto.
- **RamaNombreEstudiante**: Cada estudiante trabajará en su propia rama.

6.3. Flujo de Trabajo

- El desarrollo de funcionalidades debe realizarse en **RamaNombreEstudiante**.
- Una vez completada una funcionalidad, se hará un `merge` a la rama **develop**.
- Después de revisar el código, se integrará en **master**.

7. Consideraciones Técnicas

- Utiliza herencia para modelar las subclases de vehículos.
- Implementa manejo de fechas con `LocalDate`.