_5class_poo/gpt_class.py

```python
from abc import ABC, abstractmethod
from datetime import datetime

# Principios SOLID y Abstracción
class Vehicle(ABC):
    def __init__(self, make, model, year, license_plate):
        self.make = make
        self.model = model
        self.year = year
        self._license_plate = license_plate  # Atributo protegido
        self.__rented = False  # Atributo privado

    @abstractmethod
    def get_rental_price(self):
        pass

    def rent(self):
        if not self.__rented:
            self.__rented = True
            return True
        return False

    def return_vehicle(self):
        if self.__rented:
            self.__rented = False
            return True
        return False

    def is_rented(self):
        return self.__rented

    def __str__(self):
        return f"{self.year} {self.make} {self.model} (License: {self._license_plate})"

# Herencia y Polimorfismo
class Car(Vehicle):
    def __init__(self, make, model, year, license_plate, doors):
        super().__init__(make, model, year, license_plate)
        self.doors = doors

    def get_rental_price(self):
        return 50  # Precio fijo para coches

class Motorcycle(Vehicle):
    def __init__(self, make, model, year, license_plate, engine_capacity):
        super().__init__(make, model, year, license_plate)
        self.engine_capacity = engine_capacity

    def get_rental_price(self):
        return 30  # Precio fijo para motos

```

```python
# Composición
class Customer:
    def __init__(self, name, license_number):
        self.name = name
        self.license_number = license_number

    def __str__(self):
        return f"Customer: {self.name}, License: {self.license_number}"

class Rental:
    def __init__(self, vehicle, customer, rental_date, return_date=None):
        self.vehicle = vehicle
        self.customer = customer
        self.rental_date = rental_date
        self.return_date = return_date

    def end_rental(self, return_date):
        self.return_date = return_date
        self.vehicle.return_vehicle()

    def __str__(self):
        return f"Rental - {self.vehicle} to {self.customer} from {self.rental_date} to {self.return_date}"

# Administración del Sistema
class RentalSystem:
    def __init__(self):
        self.vehicles = []
        self.customers = []
        self.rentals = []

    def add_vehicle(self, vehicle):
        self.vehicles.append(vehicle)

    def add_customer(self, customer):
        self.customers.append(customer)

    def rent_vehicle(self, vehicle, customer, rental_date):
        if vehicle.rent():
            rental = Rental(vehicle, customer, rental_date)
            self.rentals.append(rental)
            return rental
        else:
            print(f"Vehicle {vehicle} is already rented.")
            return None

    def return_vehicle(self, rental, return_date):
        rental.end_rental(return_date)

    def __str__(self):
        vehicles_info = "\n".join(str(vehicle) for vehicle in self.vehicles)
        customers_info = "\n".join(str(customer) for customer in self.customers)
        rentals_info = "\n".join(str(rental) for rental in self.rentals)
        return (f"Rental System\n\nVehicles:\n{vehicles_info}\n\n"
                f"Customers:\n{customers_info}\n\n"
                f"Rentals:\n{rentals_info}")

# Ejemplo Completo
if __name__ == "__main__":
    # Creación del sistema de alquiler
    rental_system = RentalSystem()

    # Añadir vehículos
    car1 = Car("Toyota", "Camry", 2020, "ABC123", 4)
    moto1 = Motorcycle("Honda", "CBR", 2019, "XYZ789", 1000)
    rental_system.add_vehicle(car1)
    rental_system.add_vehicle(moto1)

    # Añadir clientes
    customer1 = Customer("John Doe", "D1234567")
    customer2 = Customer("Jane Smith", "S7654321")
    rental_system.add_customer(customer1)
    rental_system.add_customer(customer2)

    # Alquilar vehículos
    rental1 = rental_system.rent_vehicle(car1, customer1, datetime.now())
    rental2 = rental_system.rent_vehicle(moto1, customer2, datetime.now())

    # Mostrar el sistema
    print(rental_system)

    # Devolver vehículos
    rental_system.return_vehicle(rental1, datetime.now())
    rental_system.return_vehicle(rental2, datetime.now())

    # Mostrar el sistema después de las devoluciones
    print(rental_system)

```