```
# class Car:
#     def __init__(self, car_id, model, status='available'):
#         self.car_id = car_id
#         self.model = model
#         self.status = status  # 'available' or 'rented'
#
# class Customer:
#     def __init__(self, customer_id, name):
#         self.customer_id = customer_id
#         self.name = name
#
# class RentalSystem:
#     def __init__(self):
#         self.cars = []
#         self.customers = []
#         self.rented_cars = []
#
#     def add_car(self, car):
#         self.cars.append(car)
#
#     def add_customer(self, customer):
#         self.customers.append(customer)
#
#     def rent_car(self, customer_id, car_id):
#         customer = self.find_customer_by_id(customer_id)
#         car = self.find_car_by_id(car_id)
#
#         if customer and car and car.status == 'available':
#             car.status = 'rented'
#             self.rented_cars.append((customer, car))
#             print(f"{customer.name} has rented {car.model}")
#         else:
#             print("Invalid customer ID or car ID, or the car is not available.")
#
#     def return_car(self, customer_id, car_id):
#         rental_entry = self.find_rental_entry(customer_id, car_id)
#
#         if rental_entry:
#             customer, car = rental_entry
#             car.status = 'available'
#             self.rented_cars.remove(rental_entry)
#             print(f"{customer.name} has returned {car.model}")
#         else:
#             print("Invalid customer ID or car ID, or the specified car was not rented by the customer.")
#
#     def display_rented_cars(self):
#         print("List of Rented Cars:")
#         for customer, car in self.rented_cars:
#             print(f"{customer.name} - {car.model}")
#
#     def find_customer_by_id(self, customer_id):
#         for customer in self.customers:
#             if customer.customer_id == customer_id:
#                 return customer
#         return None
#
#     def find_car_by_id(self, car_id):
#         for car in self.cars:
#             if car.car_id == car_id:
#                 return car
#         return None
#
#     def find_rental_entry(self, customer_id, car_id):
#         for entry in self.rented_cars:
#             customer, car = entry
#             if customer.customer_id == customer_id and car.car_id == car_id:
#                 return entry
#         return None
#
#
#
# # Example Usage:
# rental_system = RentalSystem()
#
# # Adding Cars
# rental_system.add_car(Car(1, 'Toyota Camry'))
# rental_system.add_car(Car(2, 'Honda Accord'))
# rental_system.add_car(Car(3, 'Ford Fusion'))
#
# # Adding Customers
# rental_system.add_customer(Customer(101, 'Alice'))
# rental_system.add_customer(Customer(102, 'Bob'))
#
# # Renting Cars
# rental_system.rent_car(101, 1)
#
# # Displaying Rented Cars
# rental_system.display_rented_cars()
#
# # Returning Cars
# rental_system.return_car(101, 1)
#
# # Displaying Rented Cars after returning
# rental_system.display_rented_cars()

class car:
    def __init__(self,car_id,model,status='availble'):
        self.car_id=car_id
        self.model=model
        self.status=status

class customer:
    def __init__(self,customer_id,customer_name):
        self.customer_id=customer_id
        self.customer_name=customer_name

class Rental_System:
    def __init__(self):
        self.cars=[]
        self.customers=[]
        self.rented_cars=[]
    def addCars(self,car):
        self.cars.append(car)
        print(f"The {car} was added successfully")
    def addCustomer(self,customer):
        self.customers.append(customer)
```

Start coding or generate with AI.