Operation Contract - Admin

Contract: addAuctionHouse

Operation: addAuctionHouse(name: String, location: String, schedule: DateTime)
Cross reference(s): Use Case Add Auction House
Preconditions:
- The administrator is logged into the system.
- The auction house does not already exist in the system.

Postconditions:
- A new instance of AuctionHouse is created. [instance creation]
- The attributes name, location, and contactInfo are initialized. [modification of variable]
- The AuctionHouse instance is added to the system. [association formation]

Contract: updateAuctionHouse

Operation: updateAuctionHouse(auctionHouse: AuctionHouse)
Cross reference(s): Use Case Update Auction House
Preconditions:
- The administrator is logged into the system.
- The auction house exists in the system.

Postconditions:
- The AuctionHouse instance is updated with newInfo. [modification of variable]

Contract: deleteAuctionHouse

Operation: deleteAuctionHouse(auctionHouse: AuctionHouse)
Cross reference(s): Use Case Delete Auction House
Preconditions:
- The administrator is logged into the system.
- The auction house exists in the system.

Postconditions:
- The AuctionHouse instance is removed from the system. [instance deletion]

Contract: addObject

Operation: addObject(name: String, category: String, description: String, auctionHouse: AuctionHouse)
Cross reference(s): Use Case Add Object of Interest
Preconditions:
- The administrator is logged into the system.
- The AuctionHouse exists.

Postconditions:
- A new ObjectOfInterest instance is created. [instance creation]
- The attributes name, category, and description are initialized. [modification of variable]
- The object is linked to AuctionHouse. [association formation]

Contract: updateObjectOfInterest

Contract: updateObjectOfInterest
Operation: updateObject(object: ObjectOfInterest)
Cross reference(s): Use Case Update Object of Interest
Preconditions:
- The administrator is logged into the system.
- The object exists in the system.

Postconditions:
- The ObjectOfInterest instance is updated with newInfo. [modification of variable]

Contract: deleteObjectOfInterest

Operation: deleteObject(object: ObjectOfInterest)
Cross reference(s): Use Case Delete Object of Interest
Preconditions:
- The administrator is logged into the system.
- The object exists in the system.

Postconditions:
- The ObjectOfInterest instance is removed from the system. [instance deletion]

Contract: addExpert

Operation: addExpert(name: String, contactInfo: String, licenseNumber: String, expertiseAreas: List<String>)
Cross reference(s): Use Case Add Expert
Preconditions:
- The administrator is logged into the system.

Postconditions:
- A new Expert instance is created. [instance creation]
- The attributes name, contactInfo, licenseNumber, and expertiseAreas are initialized. [modification of variable]
- A user account for the expert is created. [association formation]

Contract: updateExpert

Operation: updateExpert(expert: Expert)
Cross reference(s): Use Case Update Expert
Preconditions:
- The administrator is logged into the system.
- The expert exists in the system.

Postconditions:
- The Expert instance is updated with newInfo. [modification of variable]

Contract: deleteExpert

Operation: deleteExpert(expert: Expert)
Cross reference(s): Use Case Delete Expert
Preconditions:
- The administrator is logged into the system.
- The expert exists in the system.

Postconditions:
- The Expert instance is removed from the system. [instance deletion]

Contract: approveClient

Operation: approveClient(client: Client)
Cross reference(s): Use Case Approve Client
Preconditions:
- The administrator is logged into the system.
- The client exists in "Pending" status.

Postconditions:
- The client's account status is changed to "Approved". [modification of variable]

Contract: updateClient

Operation: updateClient(client: Client, newInfo: Map<String, String>)
Cross reference(s): Use Case Update Client
Preconditions:
- The administrator is logged into the system.
- The client exists in the system.

Postconditions:
- The Client instance is updated with newInfo. [modification of variable]

Contract: deleteClient

Operation: deleteClient(client: Client)
Cross reference(s): Use Case Delete Client
Preconditions:
- The administrator is logged into the system.
- The client exists in the system.

Postconditions:
- The Client instance is removed from the system. [instance deletion]

Operation Contract - Expert

Contract: Find Object Information

Operation: findObject(name: String): Object
Cross reference(s): Use Case - Find Info (Object)
Preconditions:
- The Expert has already logged into the system.
- The object database exists.

Postconditions:
- If the object exists, its details are retrieved and displayed to the expert.
- If the object does not exist, an error message is displayed.

Contract: Find Auction Information

Operation: findAuction(name: String): Auction
Cross reference(s): Use Case - Find Info (Auction)
Preconditions:
- The Expert has already logged into the system.
- The auction database exists.

Postconditions:
- If the auction exists, its details (auction house, date, and location) are displayed.
- If the auction does not exist, an error message is displayed.

Contract: Find Auction Schedule

Operation: findAuctionSchedule(time: DateTime, date: Date, auctionName: String): AuctionSchedule
Cross reference(s): Use Case - Find a Specific Auction Schedule
Preconditions:
- The Expert has already logged into the system.
- The auction schedule database exists.

Postconditions:
- If the schedule exists, it is displayed with the date, location, and auction house.
- If no schedule is found, a "not found" message is displayed.

Contract: Add Availability

Operation: addAvailability(time: DateTime, date: Date, serviceType: String)
Cross reference(s): Use Case - Add Availability
Preconditions:

- The Expert has already logged into the system.
- The system must not have a conflicting availability at the same time.

Postconditions:

- A new availability entry is created in the system.
- The availability details (time, date, and service type) are stored.
- If a conflict exists, an error message is displayed.

Contract: Find Object Information

Operation: findObject(name: String): Object
Cross reference(s): Use Case - Find Info (Object)
Preconditions:

- The Expert has already logged into the system.
- The object database exists.

Postconditions:

- If the object exists, its details are retrieved and displayed to the expert.
- If the object does not exist, an error message is displayed.

Contract: Add Area of Expertise

Operation: addAreaOfExpertise(name: String)
Cross reference(s): Use Case - Add Area of Expertise
Preconditions:

- The Expert has already logged into the system.
- The expertise category must not already exist for the expert.

Postconditions:

- The system registers the new area of expertise for the expert.
- If the expertise already exists, an error message is displayed.

Contract: Update Availability

Operation: updateAvailability(newTime: DateTime, newDate: Date, newServiceType: String)
Cross reference(s): Use Case - Update Availability
Preconditions:

- The Expert has already logged into the system.
- The availability to be modified exists in the system.
- The new time and date do not conflict with an existing availability.

Postconditions:

- The existing availability is updated with the new time, date, and service type.
- If a conflict exists, an error message is displayed.

Contract: Delete Availability

Operation: deleteAvailability(time: DateTime, date: Date)
Cross reference(s): Use Case - Delete Availability
Preconditions:
- The Expert has already logged into the system.
- The availability to be removed exists in the system.

Postconditions:
- The availability is successfully removed from the system.
- The system updates the expert's schedule accordingly.
- If the availability does not exist, an error message is displayed.

Contract: Client Request Handling

Operation: findClientRequest(email: String)
Cross reference(s): Use Case - Client Requests
Preconditions:
- The Expert has already logged into the system.
- The client exists in the system.

Postconditions:
- A new consultation is scheduled and stored in the system.
- The consultation is linked to both the expert and the client.

Contract: Confirm Attendance at an Auction

Operation: confirmAuctionAttendance(auction: Auction, expert: Expert, role: String)
Cross reference(s): Use Case - Confirm Attendance at an Auction
Preconditions:
- The Expert has already logged into the system.
- The expert has been assigned or invited to the auction.

Postconditions:
- The system registers the expert’s attendance and role at the auction (e.g., advisor, representative, observer).
- The auction record is updated with the expert’s participation.
- If attendance confirmation fails (e.g., auction not found, already confirmed, or scheduling conflict), an error message is displayed.