Operation Contracts

The operation contracts outlined below include the instantiation of pages since the application is based on Java Swing user interface, where a page is its own class. There will not be a dedicated sequence diagram for those specific operation contracts.

For the three getAvailableOfferings contracts, the sequence diagram will also be omitted since the main logic behind these methods is handled by database queries.

Highlighted contracts will have a dedicated sequence diagram.

SSD: Process Offerings

1. Contract: OfferingsCRUD

    Operation: OfferingsCRUD ()

    Cross Reference: Use Case: Process Offerings

    Preconditions:

    1. The admin account has successfully logged into the system
    2. The admin selects “Offerings”

    Postconditions:

    1. A new OfferingsCRUD page is instantiated [instance creation]

2. Contract: createOffering

    Operation: createOffering (offering: Offering)

    Cross reference: Use Case: Process Offerings

    Preconditions:

    1. The admin enters the info for each field.
    2. The day and time are validated, to prevent conflicts.

    Postconditions:

    1. An instance of Offering, offering, has been created. [instance creation]
    2. An offering id is initialized for offering. [attribute modification]
    3. Title is initialized for offering. [attribute modification]
    4. Organization is initialized for offering. [attribute modification]
    5. City is initialized for offering. [attribute modification]
    6. Time is initialized for offering. [attribute modification]
    7. Capacity is initialized for offering. [attribute modification]
    8. Location is initialized for offering. [attribute modification]
    9. After further validation the offering is inserted into the database.
    10. Success message is displayed.

SSD: Instructor Selects Offering

1. Contract: SelectOffering

    Operation: SelectOffering(Instructor instructor)
    Cross Reference: Use Case: Instructor Selects Offering
    Preconditions:

    1. An instructor account has successfully logged into the system.
    2. The instructor selects “Select an Offering”.

    Postconditions:

    1. A new SelectOffering page is instantiated. [instance creation]

2. Contract: getAvailableOfferings()

    Operation: getAvailableOfferings()
    Cross Reference: Use Case: Instructor Selects Offering
    Preconditions:

    1. The offering’s instructor_id attribute is null.
    2. The offering’s city attribute matches one of the instructor’s chosen cities
    3. The offering’s title attribute matches the instructor’s specialty attribute

    Postconditions:

    1. All offerings that respect the preconditions are loaded on screen.

3. Contract: selectOffering

    Operation: selectOffering(offeringId: int)

    Cross reference: Use Case: Instructor Selects Offering

    Preconditions:

    1. The instructor has selected an offering from the loaded offerings.
    2. The instructor selects “Select Offering”.
    3. The selected offering exists.
    4. The selected offering’s time does not conflict with another offering the instructor has previously selected.

    Postconditions:

    1. The selected offering’s instructor_id attribute is set to the instructor’s id [attribute modification]

SSD: Process Bookings (Client)

1. Contract: ClientMakeBooking

   Operation: ClientMakeBooking (Client client)
   Cross Reference: Use Case: Process Bookings (Client)
   Preconditions:
   1. A client has successfully logged into the system.
   2. The client selects “Make a Booking”.

   Postconditions:
   1. A new ClientMakeBooking page is instantiated. [instance creation]

2. Contract: getAvailableOfferings

   Operation: getAvailableOfferings()
   Cross Reference: Use Case: Process Bookings (Client)
   Preconditions:
   1. The offering’s instructor_id is not null.
   2. The offering’s num_students is less than capacity.
   3. The offering has not been booked yet by the client.

   Postconditions:
   1. All offerings that respect the preconditions are loaded on screen.

3. Contract: bookOffering

   Operation: bookOffering (offeringId: int)

   Cross reference: Use Case: Process Bookings (Client)

   Preconditions:

   1. The client has selected an offering from the loaded offerings.
   2. The client selects “Book Offering”
   3. The client has not booked an offering on the same day and time.

   Postconditions:

   1. A new ClientBooking is associated with the offeringId. [association formation]
   2. The ClientBooking is associated with the clientId. [association formation]
   3. The ClientBooking is inserted into the database.
   4. The selected offering’s num_students attribute is incremented by 1. [attribute modification]

SSD: Process Bookings (Guardian)

1. Contract: GuardianMakeBooking

   Operation: GuardianMakeBooking (Guardian guardian)
   Cross Reference: Use Case: Process Bookings (Guardian)
   Preconditions:
   1. A guardian has successfully logged into the system.
   2. The guardian selects “Make a Booking”.

   Postconditions:
   1. A new GuardianMakeBooking page is instantiated.

2. Contract: getAvailableOfferings

   Operation: getAvailableOfferings()
   Cross Reference: Use Case: Process Bookings (Guardian)
   Preconditions:
   1. The offering’s instructor_id is not null.
   2. The offering’s num_students is less than capacity.

   Postconditions:
   1. All offerings that respect the preconditions are loaded on screen.

3. Contract: bookOfferingForMinor

   Operation: bookOfferingForMinor (selectedMinorName: String, offeringId: int)

   Cross reference: Use Case: Process Bookings (Guardian)

   Preconditions:

   1. The guardian has selected an offering from loaded offerings.
   2. The guardian has selected one of its minors.
   3. The guardian selects “Book Offering”.
   4. The guardian has not booked the same offering for the selected minor
   5. The guardian has not booked an offering for the minor at the same day and time.

   Postconditions:

   1. A new MinorBooking is associated with the offeringId. [association formation]
   2. The MinorBooking is associated with the minorId. [association formation]
   3. The MinorBooking is inserted into the database.
   5. The selected offering’s num_students attribute is incremented by 1.
      [attribute modification]