Informal Test Plan for the project

**Test Objectives:**

* Verify correct behavior of constructors.
* Ensure proper handling of meeting times and dates.
* Validate attendee management functionality.
* Guarantee accurate room association.
* Confirm proper description handling.
* Test information retrieval methods (getters).

# Calendar.java:

**Test default constructor:**

* Create a new Calendar object.
* Verify that the calendar is initialized with 13 empty ArrayLists, each containing 32 empty ArrayLists.
* Verify that each ArrayList contains an empty Meeting object.

**Test isBusy method:**

* Create a new Calendar object with some pre-existing meetings.
* Verify that the isBusy method returns false for times when no meetings are scheduled.
* Verify that the isBusy method returns true for times when meetings overlap.

**Test addMeeting method:**

* Create a new Calendar object.
* Call the addMeeting method with a valid Meeting object.
* Verify that the Meeting object is added to the correct month and day ArrayLists.
* Call the addMeeting method with an overlapping Meeting object.
* Verify that an exception is thrown.

**Test clearSchedule method:**

* Create a new Calendar object with some pre-existing meetings.
* Call the clearSchedule method with a valid month and day.
* Verify that the day ArrayList is empty.
* Call the clearSchedule method with an invalid month and day.
* Verify that no exception is thrown.

**Test printAgenda methods:**

* Create a new Calendar object with some pre-existing meetings.
* Call the printAgenda method with a valid month.
* Verify that the returned string contains the correct meetings for that month.
* Call the printAgenda method with a valid month and day.
* Verify that the returned string contains the correct meetings for that day.

# Meeting class:

**Test Default Constructor:**

* Create a Meeting object using the default constructor.
* Assert that all fields are initialized to expected default values.

**Meeting Times and Dates:**

**Valid Times and Dates:**

* Create Meetings with valid month, day, start time, and end time values.
* Assert that no exceptions are thrown and values are stored correctly.

**Invalid Times and Dates:**

* Create Meetings with invalid combinations of month, day, start time, and end time values (e.g., month 14, day 32, start time after end time).
* Assert that appropriate exceptions are thrown.

**Adding and Removing Attendees:**

* Create a Meeting with initial attendees.
* Add and remove attendees using the respective methods.
* Verify that the attendees list is updated correctly.

**Assigning a Room:**

* Create a Meeting and assign a Room to it.
* Assert that the Room object is correctly associated with the Meeting.

**Description:**

* Create a Meeting with a description.
* Retrieve the description using the getter method.
* Assert that the retrieved description matches the original one.

**Getter Methods:**

* Create Meetings with various values for fields.
* Use getter methods to retrieve each field's value.
* Assert that retrieved values match those set during object creation.

**toString() Method:**

* Create Meetings with different combinations of values.
* Call the toString() method for each Meeting object.
* Assert that the generated string matches the expected format.

# Testing Organization Class:

Test Cases:

**Default Employee List:**

* Verify the list size is 5.
* Check the names of the first and last employee.

**Default Room List:**

* Verify the list size is 5.
* Check the IDs of the first and last room.

**getEmployee (Existing):**

* Retrieve an existing employee by name.
* Verify the returned employee object has the correct name.

**getEmployee (Non-existent):**

* Attempt to retrieve a non-existent employee.
* Verify a Exception is thrown.

**getRoom (Existing):**

* Retrieve an existing room by ID.
* Verify the returned room object has the correct ID.

**getRoom (Non-existent):**

* Attempt to retrieve a non-existent room.
* Verify a Exception is thrown.

# Testing Person Class:

Test Cases:

**Default Constructor:**

* Verify the name is an empty string.
* Verify the calendar object is initialized.

**Name Constructor:**

* Create a person with a specific name.
* Verify the name is set correctly.
* Verify the calendar object is initialized.

**addMeeting (Conflict):**

* Add overlapping meetings to the calendar.
* Verify a TimeConflictException is thrown.

**addMeeting (No Conflict):**

* Add a meeting to the calendar.
* Verify the meeting is added successfully.
* Verify the isBusy method returns true for the meeting time.

**printAgenda (Month) (Delegate):**

* Call the printAgenda method with a month parameter.
* Verify the returned string represents the agenda for that month (assuming delegate method works).

# Testing Room Class:

Test Cases:

**Default Constructor:**

* Verify the ID is an empty string.
* Verify the calendar object is initialized.

**ID Constructor:**

* Create a room with a specific ID.
* Verify the ID is set correctly.
* Verify the calendar object is initialized.

**addMeeting (Conflict):**

* Add overlapping meetings to the calendar.
* Verify a TimeConflictException is thrown.

**addMeeting (No Conflict):**

* Add a meeting to the calendar.
* Verify the meeting is added successfully.
* Verify the isBusy method returns true for the meeting time.

**printAgenda (Month) (Delegate):**

* Call the printAgenda method with a month parameter.
* Verify the returned string represents the agenda for that month (assuming delegate method works).