Chalmers On The Go
– the Complete Chalmers Experience

An Android app as an interactive map over Chalmers, pre-release documentation

1 Installation

1.1 How to install the app

1. Download the ZIP-file containing the app
2. Unzip
3. Double click the .apk file to install

1.2 How to use

When you have installed the app and opened it, your own location will be marked on the application's map. You can search for locations via the search window, either with complete location name or by parts of a name, resulting in different matching locations being suggested. Furthermore, you can search for a specific room type, such as computer room or group room, and thereby getting suggestions for all such rooms. When you have chosen a location, it will be marked on the application's map. When a location is marked, you can click on the marked location, receiving a popup window containing the name and floor of the location. When you click on the info window, the shortest path from your current position to the marked location will be printed. In addition, you can switch between night mode and day mode using the left-most menu button on the screen. The modes contain different aspects of Chalmers; the day mode features lecture halls, group room and such study focused sites, whereas the night mode features the different pub locations.

2 Release Notes

2.1 Core features

- Search for a location on the map
- Get suggestions when searching for locations
- Have the searched location marked on the map
- Have your own location marked
- Be able to switch between night mode and day mode
- Be able to switch between different layers on the map
- Have the marked location display relevant information
- Have the shortest path from the current position to the marked location printed

2.2 Change log

Change log will be available in final release.

3 Tests

3.1 Acceptance tests

Acceptance tests have been run for the different user stories.
Those tested concern design, possibilities and restrictions on the map and application features.

3.1.1 Test details

User stories, examples:

- As a user who opens the program, I want to have a fixed view on the Chalmers map, always with the same starting coordinates.
- As a user, I should not be able to zoom out to see larger areas than that of Chalmers.
- As a user, I want to be able to search for buildings and the result should be all rooms.
- As a user, when I type in the search for an item, a dropdown menu with word-completed suggestions should appear.
- As a user I want to see where I am on campus with just one click.
- As a user, I want to be able to search for two separate locations, and get the path between them.
- As a user I want to be able to have a Checkbox Menu in which I can choose between lecture halls, computer rooms and group rooms.

3.2 Unit tests

All database methods in the DAO (Data Access Object) class have been unit tested, using the public Assert class.

3.2.1 Test details

- Insertion and getting in table 4 (buildings table)
    - insertIntoTable4 and getAllFromTable4 were tested together:
        - A building name (String) was inserted into table 4 via insertIntoTable4 and fetched with getAllFromTable4
- Insertion and getting in table 2 (room types table)
    - insertIntoTable2 and getAllFromTable2 were tested together:
        - Three room types (String) were inserted into table 2 via insertIntoTable2 and fetched with getAllFromTable2
- Insertion and getting in table 1 (coordinates and buildings table)
    - insertIntoTable1 and getClosestEntry were tested together:
        - A pair of coordinates (Double) and a building name (String) were inserted into table 1 via insertIntoTable1.
        - The coordinates (Double) were used to create an object (LatLng) containing latitude and longitude.
        - The object (LatLng) and the building name (String) served as input in getClosestEntry.
        - The result of getClosestEntry (LatLng) and the object (LatLng) containing the coordinates were compared and found to be equal.
- Calculating the closest entry
    - insertIntoTable1 and getClosestEntry were tested together:
        - An object (LatLng) containing zero coordinates, the current coordinates, were created.
        - Five different coordinate pairs (Double) and a building name (String)

were inserted into table 1 via insertIntoTable1.
        - The pair of coordinates (Double) closest to the zero coordinates, were in addition used to create an object (LatLng) containing latitude and longitude.
        - The zero coordinates object (LatLng) and the building name (String) served as input in getClosestEntry.
        - The result of getClosestEntry (LatLng) and the closest coordinate pair object (LatLng) were compared and found to be equal.
- Insertion and getting in table 3 (room name, coordinates, room type, building and floor table)
    - insertIntoTable3 and getRoomCoordinates were tested together:
        - A pair of coordinates (Double) were used to create an object (LatLng) containing latitude and longitude.
        - A room type (String) was inserted into table 2 via insertIntoTable2.
        - A building name (String) was inserted into table 4 via insertIntoTable4.
        - The room name (String), the coordinates (Double, the room type (String), the building name (String) and a floor (String) were inserted into table 3 via insertIntoTable3.
        - The room name (String) served as input in getRoomCoordinates.
        - The result of getRoomCoordinates (LatLng) and the object (LatLng) containing the coordinates were compared and found to be equal.
- Getting all rooms in a specific building
    - insertIntoTable2, insertIntoTable3, insertIntoTable4 and getAllRoomsInBuilding were tested together:
        - A room type (String) was inserted into table 2 via insertIntoTable2.
        - A real building name (String) was inserted into table 4 via insertIntoTable4
        - A false building name (String) was inserted the same way.
        - Room name1 (String), coordinate pair1 (Double), the room type (String), the true building name (String) and floor1 (String) were inserted into table 3 via insertIntoTable3.
        - Room name2 (String), coordinate pair1 (Double), the room type (String), the true building name (String) and floor2 (String) were inserted into table 3 via insertIntoTable3.
        - A false room name (String), coordinate pair1 (Double), the room type (String), the false building name (String) and floor1 (String) were inserted into table 3 via insertIntoTable3.
        - The true building name (String) served as input in getAllRoomsInBuilding.
        - The result of getAllRoomsInBuilding (ArrayList<String>) was tested using methods size and contains, and found to be satisfactory.
- Getting all rooms with a specific type
    - insertIntoTable2, insertIntoTable3, insertIntoTable4 and getAllRoomsInBuilding were tested together:
        - A room type (String) was inserted into table 2 via insertIntoTable2.
        - A false room type (String) was inserted the same way.
        - A building name (String) was inserted into table 4 via insertIntoTable4
        - Room name1 (String), a coordinate pair1 (Double), the true room type (String), the building name (String) and floor1 (String) were inserted into table 3 via insertIntoTable3.
        - Room name2 (String), coordinate pair1 (Double), the true room type (String), the building name (String) and floor2 (String) were inserted

into table 3 via insertIntoTable3.
        - A false room name (String), coordinate pair1 (Double), the false room type (String), the building name (String) and floor1 (String) were inserted into table 3 via insertIntoTable3.
        - The building name (String) served as input in getAllRoomsInBuilding.
        - The result of getAllRoomsInBuilding (ArrayList<String>) was tested using methods size and contains, and found to be satisfactory.
- Getting suggestions
    - insertIntoTable2, insertIntoTable3, insertIntoTable4 and suggestions were tested together:
        - A room type (String) was inserted into table 2 via insertIntoTable2.
        - A building name (String) was inserted into table 4 via insertIntoTable4
        - A room name (String), a coordinate pair (Double), the room type (String), the building name (String) and a floor (String) were inserted into table 3 via insertIntoTable3.
        - Different strings of letters matching the strings in table 3 served as input in suggestions.
        - The result of suggestions (ArrayList<String>) was tested using methods for size and null, and found to be satisfactory.
- Getting room names
    - insertIntoTable3 and getName were tested together:
        - A room name (String), a coordinate pair (Double), a room type (String), a building name (String) and a floor (String) were inserted into table 3 via insertIntoTable3.
        - The room name (String) served as input in getType.
        - The result of getName (String) and the room name were compared and found to be equal.
- Getting room types
    - insertIntoTable3 and getType were tested together:
        - A room name (String), a coordinate pair (Double), a room type (String), a building name (String) and a floor (String) were inserted into table 3 via insertIntoTable3.
        - The room name (String) served as input in getType.
        - The result of getType (String) and the room type were compared and found to be equal.
- Getting floor
    - insertIntoTable3 and getFloor were tested together:
        - A room name (String), a coordinate pair (Double), a room type (String), a building name (String) and a floor (String) were inserted into table 3 via insertIntoTable3.
        - The room name (String) served as input in getFloor.
        - The result of getFloor (String) and the floor were compared and found to be equal.