Sarah Goh Shi Ning - Project Portfolio

PROJECT: PrisonBook

Overview

Prisonbook is a desktop address book application used for Prison Guards/Inmates management. The user interacts with it using a CLI, and it has a GUI created with JavaFX. It is written in Java, and has about 10 kLoC.

Summary of contributions

- **Major enhancement**: Allowing users to manage the prison cells.
  - What it does: There is a cell map that has a fixed size and certain number of cells. It allows guards and prison wardens to add prisoners to cells and to remove them. It also allows guards and prison wardens to view a list of prisoners in a specified cell to help them do the previously mentioned features.
  - Justification: This helps to keep track of prisoners and their locations and to take into consideration prisoner combinations when adding and removing them from cells.
- **Minor enhancement**: Map of prison cells on GUI. It hides information on GUI before logging in and after logging out. It will also show the list of persons in PrisonBook and people in the map when logged in successfully.
  - Justification: This helps to boost security for the application, by preventing users who have insufficient security level to view classified information.
- **Code contributed**: [Functional code (https://github.com/CS2103JAN2018-T11-B2/main/blob/master/collated/functional/sarahgoh97.md)] [Test code (https://github.com/CS2103JAN2018-T11-B2/main/blob/master/collated/test/sarahgoh97.md)]
- **Other contributions**:
  - Project management:
    - Managed release `v1.3` on GitHub
    - Opened and tagged issues
    - Assigned issues to different team members
  - Enhancements to existing features:
    - Status bar shows number of people in the PrisonBook
    - Map panel to show cell system of prison

- Documentation:
  - Did revisions to existing contents of the User Guide and Developer Guide

Contributions to the User Guide

Given below are sections I contributed to the User Guide. They showcase my ability to write documentation targeting end-users.

Adding a prisoner to a cell: `addcell` or `ac` [2] (v1.3)

Adds a person to one of the cells in the prison.
Format: `addcell INDEX CELL`

Person added must be a prisoner and not already in a cell within the prison.
The cell must be in the map and not full.
The maximum number of people in a cell is 2 people.

Examples:

- `addcell 1 1-1`
- `ac 2 2-1`

Deleting a person from a cell : `deletecell` or `dc` [2] (v1.4)

Deletes the specified person from his/her cell in the PrisonBook.
Format: `deletecell INDEX`

- Deletes the person at the specified `INDEX`.
- The index refers to the index number shown in the most recent listing.
- The index **must be a positive integer** 1, 2, 3, …
- The person must be imprisoned in the prison already.

Examples:

- `dc 1`
  Deletes the first person in the prison book from his/her cell if he/she is imprisoned here.

- `listcell 1-1`
  `deletecell 1`
  Deletes the first person in the cell 1-1 if there are any prisoners inside it.

Listing prisoners in a cell : `listcell` or `lc` [1] (v1.5rc)

Shows a list of all persons in a prison cell in the prison book.
Format: `listcell CELL`

Examples:

- `listcell 1-1`
- `lc 3-3`

Showing cells in the map: `map` or `m` [1] (v1.1)

Shows a table of prison cells like a map in the command display (used for debugging and checking UI).
Format: `map`

Contributions to the Developer Guide

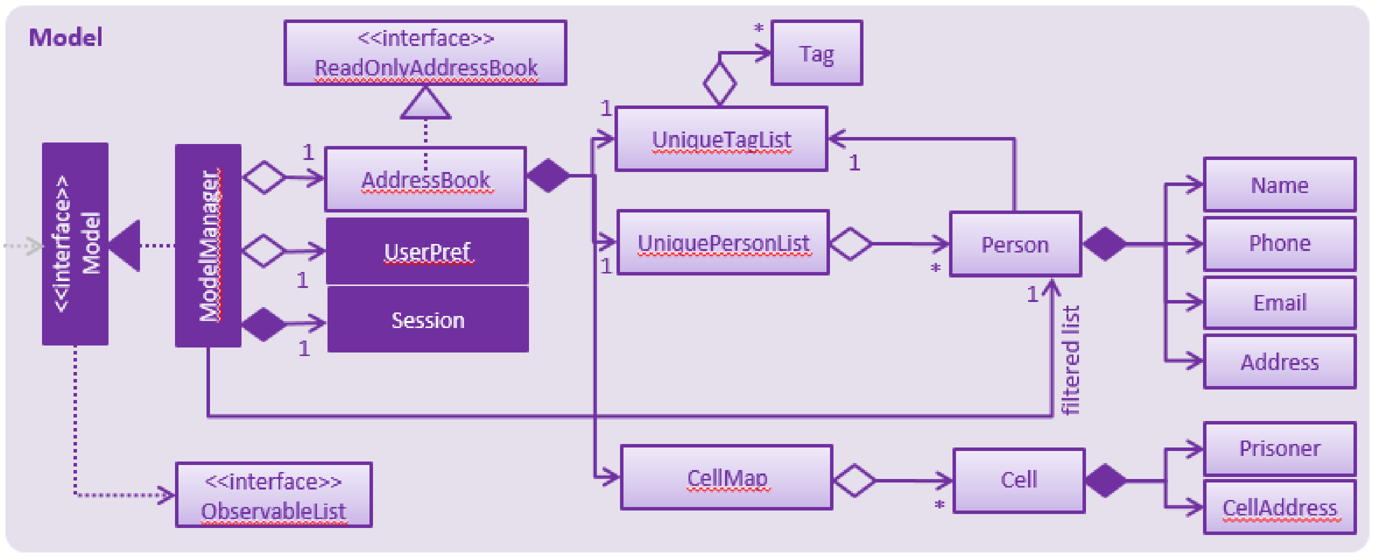
Given below are sections I contributed to the Developer Guide. They showcase my ability to write technical documentation and the technical depth of my contributions to the project.

Prison Cell System

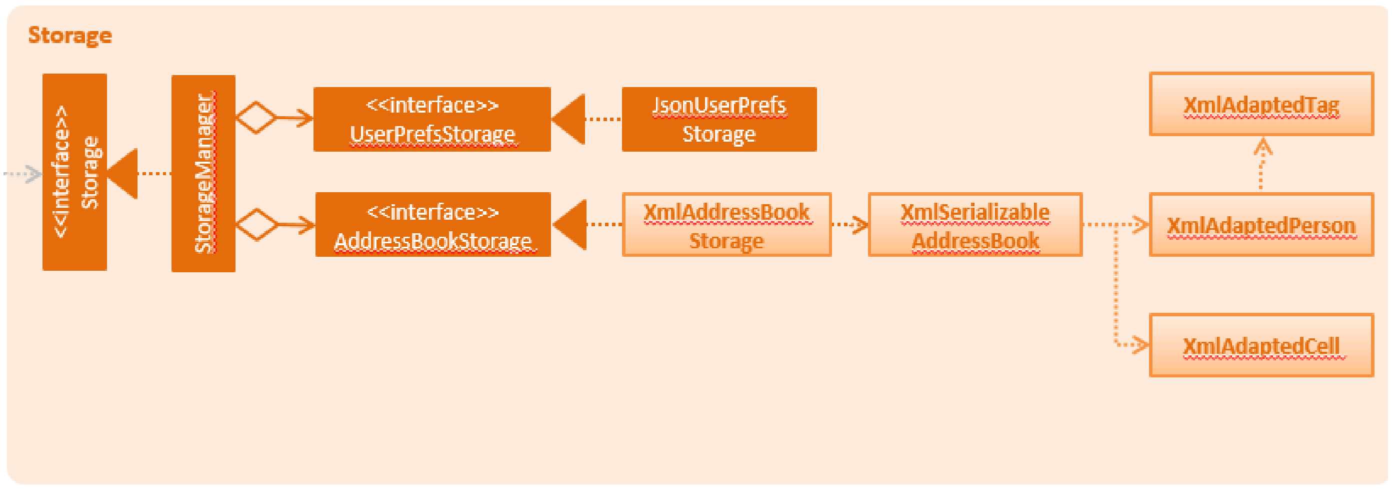
Current Implementation

In a prison, it is necessary to have cells where prisoners stay in. It is important to keep track of the prisoners in the cell and whether the `Cell` is full.

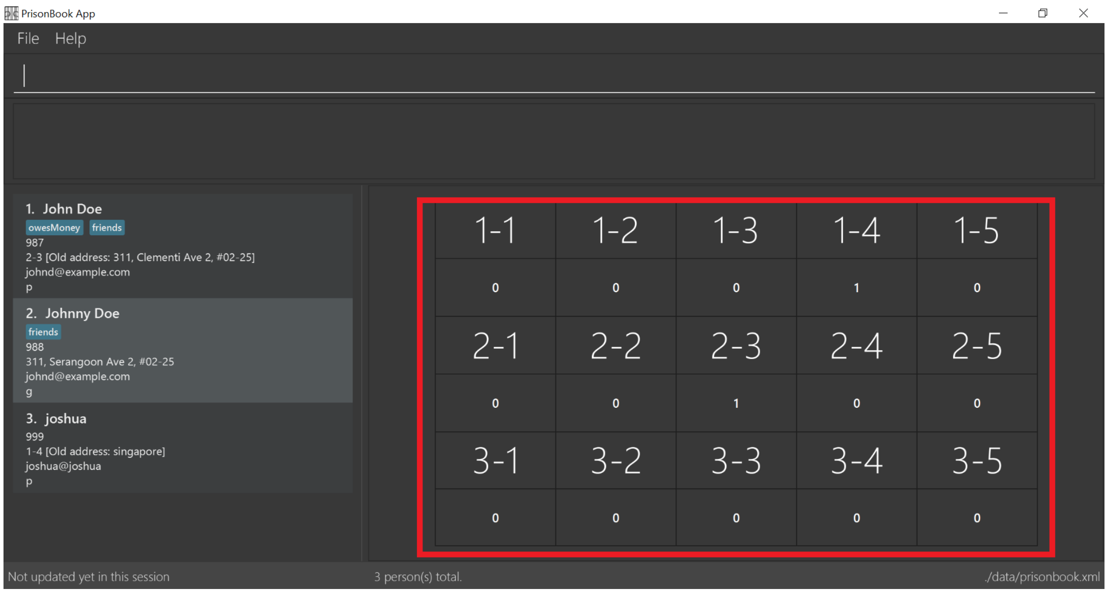
This cell system is most clearly represented using a 2-D array. Hence, this was used as the basis of the map system of the prison. However, to prevent accidental changes, a read-only version of the cells are returned in an `ObservableList`. Since this is part of the `Model` component of the PrisonBook, the `Model` component has been updated accordingly as shown below. At the bottom of the `Model` class diagram, it has been updated to show how the cell system is linked to the other components.



To save and load the data of the cell system, storage also played an important component to ensure no data is lost when closing the app. Below is a class diagram of `Storage` to show how the cells are saved. A new type is added to the storage so that it knows how to save and load data for cells.



This map is shown on the user interface on the right side area, shown in the picture below. Since the app is still being tested, the `map` command will also show the actual cells in the result display box. This basic map shows the cell system and helps keep track of what is happening in the cells.



1. John Doe
owesMoney friends
987
2-3 [Old address: 311, Clementi Ave 2, #02-25]
johnd@example.com
p

2. Johnny Doe
friends
988
311, Serangoon Ave 2, #02-25
johnd@example.com
g

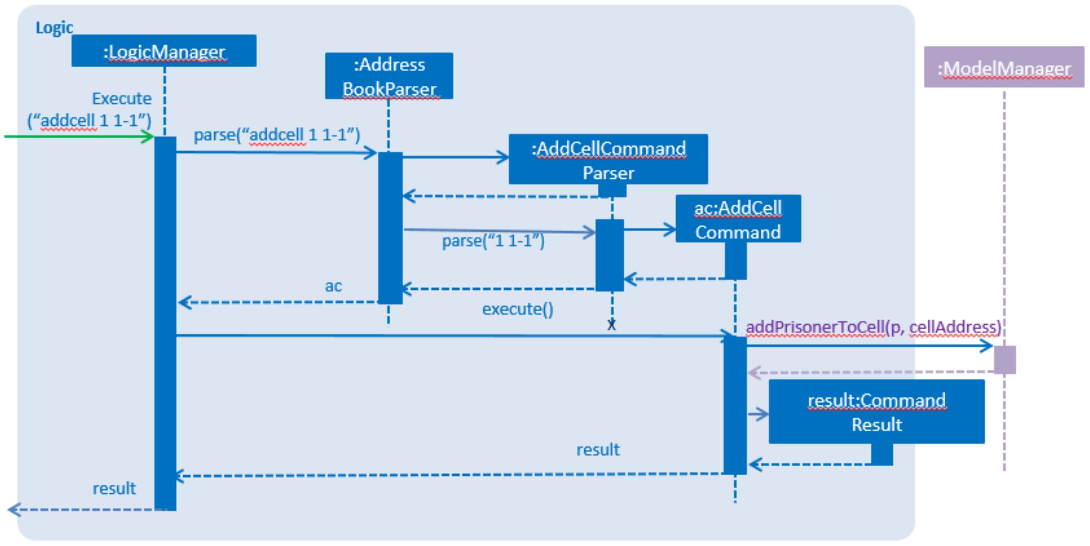
3. joshua
999
1-4 [Old address: singapore]
joshua@joshua
p

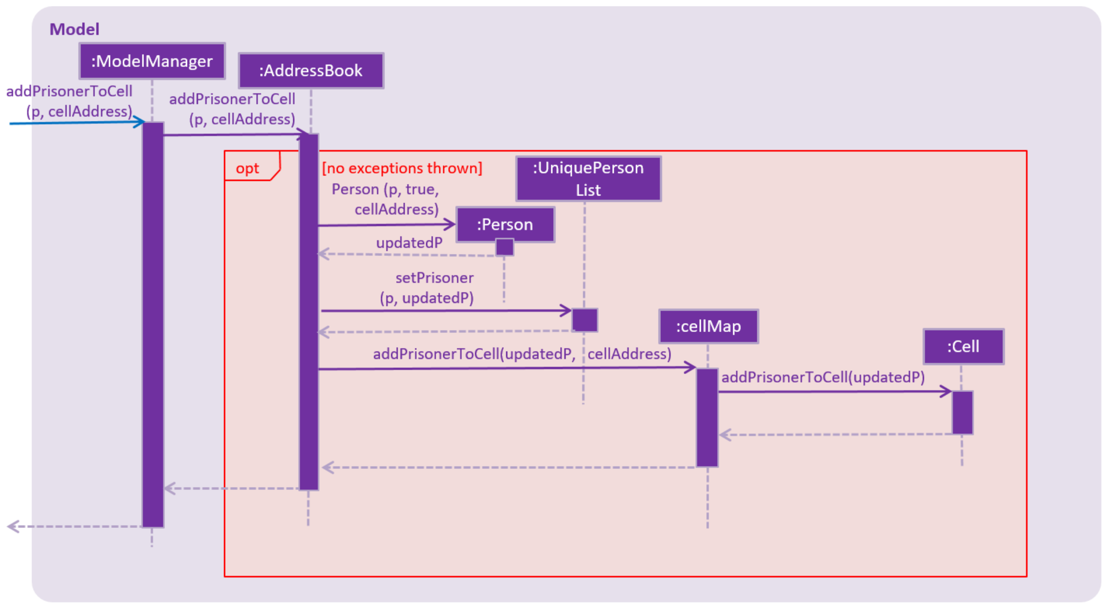
1-1	1-2	1-3	1-4	1-5
0	0	0	1	0
2-1	2-2	2-3	2-4	2-5
0	0	1	0	0
3-1	3-2	3-3	3-4	3-5
0	0	0	0	0

Not updated yet in this session 3 person(s) total. ./data/prisonbook.xml

This cell system helps users visualise the locations of the prisoners more easily. Not only that, prisoners can also be added and removed from them. These changes can be viewed in the bigger map which displays the number of people in individual cells below each cell address.

Assuming the user has access to the functions to add prisoners into prison cells, they can use the `addcell` command to do so. Below is the sequence diagram of the logic and model components.




Model
:ModelManager
:AddressBook
addPrisonerToCell (p, cellAddress)
addPrisonerToCell (p, cellAddress)
opt
[no exceptions thrown]
Person (p, true, cellAddress)
:UniquePerson List
:Person
updatedP
setPrisoner (p, updatedP)
:cellMap
:Cell
addPrisonerToCell(updatedP, cellAddress)
addPrisonerToCell(updatedP)

Besides the cell map system being updated, we will also update the addresses of these prisoners added to the prison, since they now live in the prison, rather than in their own homes.

Prisoners can also be removed from their cells using the `deletecell` command when they are being transferred to another cell, or if they can leave prison and go back home to their loved ones.

To aid in adding prisoners to cells and deleting them from the cells, a `listcell` command has been implemented. With this command, users can see who are in a specified cell. This helps users know whether the cell is full and if some of these prisoners should not be added to the same cell.

For future implementations, we can also allow only the prison wardens/owners to redefine the maximum size of the prison as their prisons may grow or shrink in the future. In this application, we will only show an example with 15 cells with a maximum of 2 prisoners per cell. These values can be easily changed and we will customise it based on your needs. The naming of the cells can also be adjusted accordingly.

Design Considerations

Aspect: Data Structure of Cells in CellMap

- **Alternative 1 (current choice):** Use a 2-D array
  - Pros: Easy to visualise the system.
  - Cons: Harder to do more functions that the Java API has provided. Does not fit in with the other models of Person and Tag and their internalList. It also makes it harder to implement undo and redo functions for actions below.
- **Alternative 2:** Use a 2-level ArrayList
  - Pros: Can use Java functions meant for Lists, can have different number of units for each block
  - Cons: Not uniform number of units for each block will be harder to take into consideration when using the individual cells.

- **Alternative 3:** Use an ObservableList
  - Pros: Easier to save state of cells when undoing and redoing.
  - Cons: Harder to visualise where each row of cells/prisons end.

Aspect: Implementation of ShowCellsCommand (`map`)

- **Alternative 1 (current choice):** Keep the map always on display.
  - Pros: Easier to see when adding prisoners to cells and removing prisoners from cells.
  - Cons: Anyone walking by from behind can see the information easily and compromises security.
- **Alternative 2:** Create a new Command to show the map.
  - Pros: Easy to implement, similar to listing. Extending the current list of commands. Only when function is called can the map be seen.
  - Cons: Easy to forget which cell in the map can add prisoners but the map is not permanently there.

Aspect: Implementation of AddCellCommand (`addcell`)

- **Alternative 1 (current choice):** Prisoners are added into cells in a cellMap and their addresses will be updated.
  - Pros: Cells are always there and can hold prisoners. Since persons in the PrisonBook can be either prisoners or guards, not all persons may have a cell. Easier to delete prisoners from cells and update their addresses accordingly when they are released from prison.
  - Cons: Need to have extra implementation to ensure undo and redo commands work properly when deleting a prisoner who is imprisoned as well as to remove them or add them to cells.
- **Alternative 2:** Not have a cell map but to add cells to prisoners.
  - Pros: Might be more intuitive for the user, to check a prisoner’s cell location.
  - Cons: Harder to manage a system of cells. Easy to accidentally overcrowd a cell. Not all persons in PrisonBook are prisoners. Some of them are guards.
- **Alternative 3:** Automatically assigning cells to prisoners
  - Pros: Removes the burden of choosing a cell for user. Fewer commands required as prisoners are automatically added.
  - Cons: More complex in automatically choosing cells for prisoners. User might want to have control on which prisoners should be in which cell.

Aspect: Implementation of ListCellCommand (`listcell`)

- **Alternative 1 (current choice):** Users to give cell address to show prisoners in them.
  - Pros: Easy to implement, similar in structure to `Find` Command and its listing of persons.

  - Cons: Need to find address of cell first that is required.
- **Alternative 2:** Clicking on a cell to show prisoners in them
  - Pros: Might be more intuitive for the user to click on a cell.
  - Cons: Not ideal for users who prefer CLI.

Aspect: Implementation of DeleteCellCommand (`deletecell`)

- **Alternative 1 (current choice):** Removes a prisoner from a cell by choosing the prisoner and updates his/her address back accordingly.
  - Pros: Faster way to remove since only the index of the prisoner is required. User does not have to figure out which cell the prisoner he or she is in.
  - Cons: Since cells have a size limit of 2, it is easier to delete a person after selecting a cell.
- **Alternative 2:** Choosing a cell and the person to delete from the cell after seeing the list.
  - Pros: Easier to choose a cell when there are many people in the PrisonBook and is difficult to see who the prisoners are.
  - Cons: Requires more steps to delete a person from the cell, by having to first list then delete. Current implementation can work in a similar way that makes it easier to delete a prisoner from a cell.
- Sarah Goh Shi Ning (sarahgoh97)
  1. Major enhancement:
     Creating a cell system of all the prison cells in the prison. This helps users add and remove prisoners to their prison and prevent users from overcrowding their prison. It can also help users keep track of locations of the cells of each prisoner. Users can also add and remove prisoners more easily by first listing people in each cell to decide where to add or remove them.
  2. Minor enhancement:
     Show map of the cell system on the UI. Also implemented the UI to show the list of persons and the map when first logged in and to hide it when logging out and when the application is first opened.

Minor Feature: Showing prisoners in the UI

1. Map panel in UI showing the map of cells in the system.
   a. Prerequisites: Have sufficient security level[1]
   b. Test case: Starting up the app and before logging in
      Expected: No details regarding cells shown on the map. Map shows that user has insufficient access to view information.
   c. Test case: Using features below while logged in
      Expected: UI matches what `map` command shows on the display.

Major Feature: Managing prisoners in a prison

Adding a prisoner to a cell

1. Adding a prisoner to a cell when all prisoners are listed and has sufficient security level [2].
   a. Prerequisites: List all persons using the `list` command. Multiple persons in the list, from initial state of program.
   b. Test case: `addcell 5 1-5` (person at index is a prisoner and is not in a cell and cell is not full)
   Expected: Fifth person is added to the cell with address 1-5. Address of the prisoner updated in the person card and message of prisoner added to cell is displayed. The map on the right side of the UI will also update correspondingly by adding to the number corresponding to cell address 1-5. Timestamp in the status bar is updated.
   c. Test case: `addcell 0 1-4`
   Expected: No prisoner is added to a cell because of the index. Error message displayed regarding format. The map and status bar will not be updated.
   d. Other incorrect add cell commands to try: `ac`, `addcell 2 1=1`
   Expected: No change to UI and displays error message.

Deleting prisoner from a cell

1. Deleting a prisoner from a cell when all prisoners are listed and has sufficient security level [2].
   a. Prerequisites: List all persons using the `list` command. Multiple persons in the list, from initial state of program.
   b. Test case: `deletecell 1`
   Expected: Prisoner is deleted from his/her cell and their original cell will be updated. Message regarding prisoner being released will be displayed. The map on the right will be updated, as well as the prisoner's address back to their old address.
   c. Test case: `deletecell 0`
   Expected: No change to map and to the list of persons. Message regarding error will be shown for correct format of command.
   d. Other incorrect delete cell commands to try: `dc`, `dc 0`
   Expected: Similar to previous.

Listing prisoners in a cell

1. Listing a prisoner from a cell when here is sufficient security level [1].
   a. Test case: `listcell 1-1`
   Expected: List of prisoners in cell address 1-1 will be updated and shown.
   b. Test case: `listcell 10-10`
   Expected: There is no change in the UI except the error message regarding an invalid cell.

c. Other incorrect list cell commands to try: `lc` , `lc 1`
   Expected: Similar to previous.

Last updated 2018-04-14 08:49:59 UTC