TouristBook - Test Script

1. How to load sample data

Create a directory named “data” in the same file as the .jar, then place the sampledata.xml into the newly created “data” directory. Rename the file to addressbook.xml.

2. Testing Autocomplete

Pressing tab while in the command box activates autocomplete

2.1. Example:

Input: ‘ad’
Expected result: Autocorrects to ‘add’
Input: ‘e’
Expected result: A dropdown box appears with all possible commands for e, including exit, export, and edit
Input: ‘xl’
Expected result: The text turns red because there is no possible command for the given input
Input: ‘add’
Expected result: Since this is a valid command, the parameters for the command are displayed so that the user can check without having the command fail

3. Testing Groups

3.1. Command Format

`group [GROUP NAME] [NAMES…]`

- `group [GROUP NAME]` creates a group called [GROUP NAME] if one doesn’t already exist. If one does, it will be deleted.
- `group [GROUP NAME] [NAMES…]` creates a group called [GROUP NAME] and will add the people in the following arguments (NAMES…) to it

3.2. Creating new group with people

Command: `group trip Alex Bernice Grant`
Expected result: Creates a group called trip with Alex, Bernice, and Grant

3.3. Deleting group

Command: `group trip`
Expected result: Deletes the previously created 'trip' group

3.4. Creating new group without members

Command: `group Vietnam`
Expected result: creates a new group called Vietnam

3.5. Adding persons to existing group

Command: `group Vietnam Alex Bernice Don`
Expected result: Add Alice, Bernice, and Don to the existing Vietnam group

4. Testing Group Filters

`filter` will attempt to apply the given groupname as a filter to the person display === Command Format `filter [groupname]`

4.1. Filtering to default group "none"

Command: `filter none`
Expected result: Applies the filter for the group ‘none’ to the displayed person list. This is the default group.

4.2. Applying filter to group

Command: `filter trip`
Expected result: Will change the display to only show people in the ‘trip’ group

4.3. Applying filter to nonexistent group

Command: `filter [nonexistent group]`
Expected result: Returns an error that the group doesn’t exist

4.4. Applying filter to empty group

Command: `filter [empty group]`
Expected result: Returns an error that the group is empty

5. Testing Image

All persons should have a default image when added

5.1. Command Format

`image INDEX FILEPATH`

5.2. Setting image

Command: `image 4 /home/user/somepic.jpg`
Expected result: The command will set the picture “somepic.jpg” for the person at index “4”

5.3. Removing image

Images can be removed for the person using
Command: `image INDEX`
Expected result: The image for the person at the index will be removed and the default image will be set back

6. Testing Export

Used to export the current working addressbook

6.1. Command Format

```
export [INDEX] [FILEPATH]
```

6.2. Export

Command: `export`
Expected result: The addressbook database and images to the application directory will be exported as “AddressbookData.zip”

6.3. Export to designated filepath

Command: `export /home/user/addressbookfilename`
Expected result: Will export the addressbook database and images to the filepath as “addressbookfilename.zip”

7. Testing Add

7.1. Command Format

```
add n/NAME p/PHONE_NUMBER e/EMAIL a/ADDRESS [t/TAG] [d/EXPIRY DATE]…
```

7.2. Adding person with expiry date field

Command: `add n/Kaye Williams p/96182716 e/kayew@example.com a/16-201, North Tower, UTown d/2017-10-10`
Expected result: A new person “Kaye Williams” with an expiry date 2017-10-10 and other details will be added. New person card will be created with expiry date and other details fields.

8. Testing Edit

8.1. Command Format

`edit INDEX [n/NAME] [p/PHONE] [e/EMAIL] [a/ADDRESS] [t/TAG] [d/EXPIRY DATE]…`

8.2. Editing person and adding a new expiry date

Assuming the person at index 1 has no expiry date initially
Command: `edit 1 n/Amy Chan d/2017-09-09`
Expected result: The name of the person at index 1 is changed to “Amy Chan”, expiry date of 2017-09-09 is also added to the person. Person card changes content of name and displays expiry date field.

8.3. Editing person and his/her existing expiry date

Assuming the person at index 1 already has an expiry date
Command: `edit 1 p/12345678 d/2017-10-10`
Expected result: The phone number of the person is changed to 12345678, his/her expiry date is changed to 2017-10-10. Person card shows the changes correspondingly.

9. Testing Expire

By default, a person would not have an expiry date.

9.1. Command Format

`expire INDEX [d/EXPIRY DATE]`

9.2. Adding expiry date

Command: `expire 1 d/2012-01-01`
Expected result: Expiry date of person at index 1 set to 2012-01-01. Person card shows the newly added expiry date

9.3. Editing expiry date

Assuming the person at index 1 already has an expiry date, to edit the date, do the following:
Command: `expire 1 d/2017-09-09`
Expected result: Expiry date of the person now set to 2017-09-09. Person card shows the new expiry date.

9.4. Removing expiry date

Assuming the person at index 1 already has an expiry date, to remove the expiry date, do the following:
Command: `expire 1`
Expected result: Expiry date of the person at index 1 removed. Person card no longer has the expiry date field.