

Dormie User Guide

Dormie is a **desktop app for managing contacts, optimized for use via a Command Line Interface** (CLI) while still having the benefits of a Graphical User Interface (GUI). If you can type fast, Dormie can get your contact management tasks done faster than traditional GUI apps.

Dormie User Guide

[Command Summary](#)

[Using This User Guide](#)

[Text Types](#)

[Searching for Keywords \(Ctrl-F\)](#)

[Quick Start](#)

[Contact Fields](#)

[Features](#)

[Creates a new contact for a dorm mate : add](#)

[Add Free Time Tag : addTime](#)

[Clear the entire contact list : clear](#)

[Deleting a person : delete](#)

[Delete Free Time Tag : deleteTime](#)

[Editing a person : edit](#)

[End the application : exit](#)

[Filter the contact list by name : find](#)

[Provide more detail on existing commands : help](#)

[Listing all persons in Dormie : list](#)

[Check who is free : whoisfree](#)

[Saving the data](#)

[Editing the data file](#)

[FAQ](#)

[Known Issues](#)

[Glossary](#)

Command Summary

Action	Format, Examples
Add	<code>add n/NAME r/ROOM_NUM</code> e.g., <code>add Alice Lim r/02-03</code>
Clear	<code>clear</code>
Delete	<code>delete INDEX</code> e.g., <code>delete 3</code>
Edit	<code>edit INDEX [n/NAME] [r/ROOM_NUM]</code> e.g., <code>edit 1 n/Alex r/05-11</code>
Exit	<code>exit</code>
Find	<code>find KEYWORD</code> e.g., <code>find Alice</code>
Help	<code>help</code>
List	<code>list</code>
Add Free Time	<code>addTime INDEX ft/TIME</code>

Action	Format, Examples
	e.g., <code>addTime 1 ft/Mon:0800-1200</code>
Delete Free Time	<code>deleteTime INDEX ft/TIME</code> e.g., <code>deleteTime 1 ft/Mon:0800-1200</code>
Who Is Free	<code>whoisfree TIME</code> e.g., <code>whoisfree Mon:0800</code>

Using This User Guide

Text Types

Type	What it means
Bold	Command word e.g., add , which adds a new contact
<code>Code Block</code>	A line of command that can be entered into Dormie's input field e.g., <code>add n/John Doe p/98765432 e/johnd@example.com r/sw-01-01 t/johnDoe b/12/12/2000 ft/Mon:1300-1400</code>
<code><value></code>	Value for the respective field e.g., <code>add n/<name> p/<phoneNumber> e/<email> r/<roomNumber> t/<telegramHandle> b/<birthday></code>
<code>[optionalField]</code>	Indicates an optional field e.g., <code>add n/<name> [t/<telegramHandle>]</code> , where <code>telegramHandle</code> is an optional field.

Searching for Keywords (Ctrl-F)

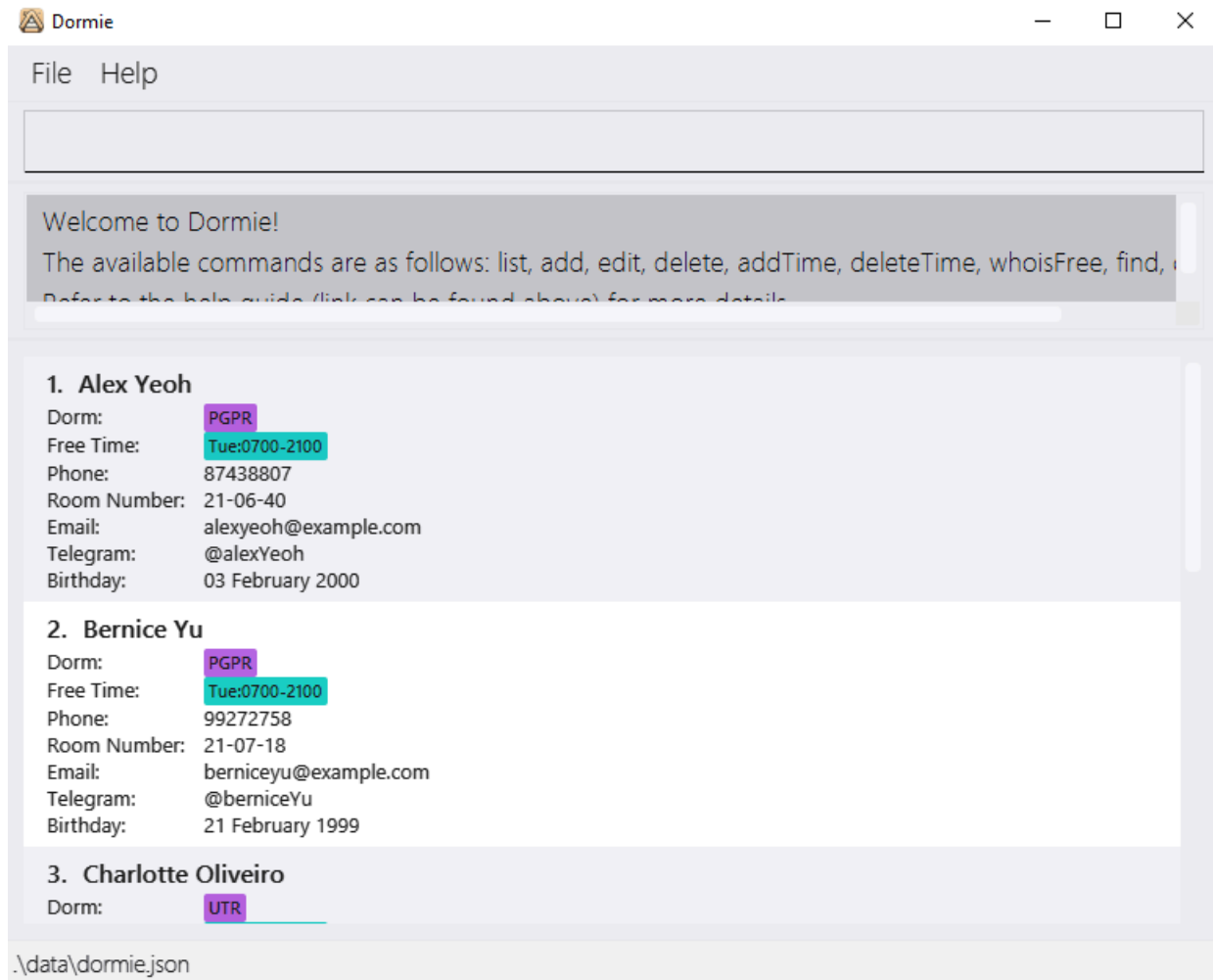
1. Press the Ctrl + F keys on your keyboard.
2. A search bar or dialog box should appear on your screen.
3. Type the keyword or phrase you want to search for in the search bar or dialog box.
4. Press Enter or click on the "Find" button to start the search.
5. The document viewer or web browser will highlight all instances of the keyword or phrase found within the document.
6. To navigate through the search results, you can use the arrow buttons or options provided by the search feature.
7. Once you have finished reviewing the search results, you can close the search bar or dialog box to return to your document.

Quick Start

1. Ensure you have Java `11` or above installed in your Computer.
 - o MacOS: [Java 11](#)
 - o Windows: [Java 11](#)
2. Download the latest `dormie.jar` from [here](#).
3. Create a new *Home Folder* you want Dormie to permanently reside in.
 - o Move Dormie into the *Home Folder*
 - o This is where Dormie and all it's supporting data will be stored
4. Open a command terminal (`Terminal` for MacOS, or `Windows Terminal` for Windows)
 - o MacOS:
 1. Right click the *Home Folder*
 2. Left click `New Terminal at Folder`
 - o Windows:
 1. Navigate into the *Home Folder*
 2. Right click anywhere inside the *Home Folder*
 3. Left click `Open in Windows Terminal`

5. Past this Command into the new terminal window `java -jar dormie.jar` and press enter.

A GUI similar to the below should appear in a few seconds. Note how the app contains some sample data.



6. Type the command in the command box and press Enter to execute it.

Quick Tutorial:

- `add n/John Doe n/sw-01-01` : Adds a contact named `John Doe` to Dormie.
- `find John` : Finds a contact with `John` in his name.
- `delete 1` : Deletes the 1st contact shown in the current list.
- `list` : Lists all contacts again.

7. Refer to [Features](#) below for details of each command or to [Command Summary](#) for a summary of the commands.

Contact Fields

Field Name	Valid Format, Examples
Name	Any non-empty String, eg: <code>Dormie</code>
Dorm Tag	Any non-empty String, eg: <code>PGPR</code>
Free Time Tag	DDD:HHmm-HHmm, Start Time must be earlier than End Time, eg: <code>Mon:1300-1400</code>
Phone	Must contain 8 digits, eg: <code>91234567</code>
Room Number	{block}-{floor}-{room number}, where block and room number are at least 2 alphanumeric characters and floor is strictly 2 alphanumeric characters, eg: <code>nw-12-12</code>

Field Name	Valid Format, Examples
Email	Must contain @ and .com, eg: <code>dormie@example.com</code>
Telegram	Can only contain case-insensitive letters A-Z, digits 0-9, and underscores, with a length between 5 and 32 characters, eg: <code>dormie_123</code>
Birthday	dd/MM/yyyy, "dd-MM-yyyy", "yyyy/MM/dd", "yyyy-MM-dd, eg: <code>21/01/2024</code> , <code>21-01-2024</code> , <code>2024/01/21</code> , <code>2024-01-21</code>

Features

i

Notes about the command format:

- Words in `UPPER_CASE` are the parameters to be supplied by the user.
e.g. in `add n/NAME` , `NAME` is a parameter which can be used as `add n/John Doe` .
- Items in square brackets are optional.
e.g `n/NAME [t/TAG]` can be used as `n/John Doe t/friend` or as `n/John Doe` .
- Free Time Tags are in the following format: `DDD:HHmm-HHmm`
- `DDD` is from Mon-Sun, `HHmm` is 24 hour time format
e.g. **Mon:1300-1400**
- Parameters can be in any order.
e.g. if the command specifies `n/NAME p/PHONE_NUMBER` , `p/PHONE_NUMBER n/NAME` is also acceptable.
- Extraneous parameters for commands that do not take in parameters (such as `list`) will be ignored.
e.g. if the command specifies `list 123` , it will be interpreted as `list` .
- Fields with the format followed by `...` can have more than one copy in the command.
e.g. `[ft/FREE TIME TAG]...` means there can be more than one free time tag.
- Command keywords for the commands should be exactly the same as demonstration, or else Dormie would not recognize it. e.g. `add` is the command keyword in the add command, where `Add` or `ADD` will not work.
- If you are using a PDF version of this document, be careful when copying and pasting commands that span multiple lines as space characters surrounding line-breaks may be omitted when copied over to the application.

Creates a new contact for a dorm mate : add

Adds a person to Dormie.

Format: `add n/NAME p/PHONE [e/EMAIL] [r/ROOM NUMBER] [t/TELEGRAM] [b/BIRTHDAY] [d/DORM TAG] [ft/FREE TIME TAG]...`

Command usage examples:

- Adding a person with only mandatory fields: `add n/Alice Lim p/91234567`
- Adding a person with all mandatory fields and some optional fields: `add n/Bob p/88998899 r/21-03-01 ft/Tue:1300-1400 ft/Mon:0900-1200 ft/Tue:0800-1000`
- Adding a person with all fields: `add n/John Doe p/98765432 e/johnd@example.com r/sw-01-01 t/johnDoe b/12/12/2000 d/PGPR ft/Mon:1300-1400`

Some common cases that considered as invalid inputs:

- Invalid format, e.g. missing spaces
- Missing mandatory fields (name and phone)
- Duplicate parameters for fields except free time tags: e.g. having two name in the command

Add Free Time Tag : addTime

Adds 1 or multiple specified `freeTimeTags` to the specified person.

Format: `addTime INDEX ft/FREE_TIME_TAG...`

Notes on format:

- The index refers to the index number shown in the displayed person list
- The index must be a positive integer e.g. 1, 2, 3

Command usage examples:

- Single input: `addTime 1 ft/Mon:1300-1400`
- Multiple input: `addTime 1 ft/Mon:1300-1400 ft/Tue:1300-1400`

Future enhancement (not implemented yet):

- The command can be performed on multiple person, e.g. `addTime 1 2 ft/Mon:1300-1400` will add the free time tag to both the first and second person.

Clear the entire contact list : clear

Removes all existing contacts from Dormie.

Format: `clear`

Deleting a person : delete

Deletes the specified person from Dormie.

Format: `delete INDEX`

Notes on format:

- Deletes the person at the specified INDEX
- The index refers to the index number shown in the displayed person list
- The index must be a positive integer e.g. 1, 2, 3

Command usage examples:

- Deletes the 2nd person in Dormie: `delete 2`

Delete Free Time Tag : deleteTime

Deletes 1 or multiple specified `freeTimeTags` from the specified person.

Format: `deleteTime INDEX ft/FREE_TIME_TAG`

Notes on format:

- The index refers to the index number shown in the displayed person list
- The index must be a positive integer e.g. 1, 2, 3

Command usage examples:

- Single input: `deleteTime 1 ft/Mon:1300-1400`
- Multiple input: `deleteTime 1 ft/Mon:1300-1400 ft/Tue:1300-1400`

Future enhancement (not implemented yet):

- The command can be performed on multiple person, e.g. `deleteTime 1 2 ft/Mon:1300-1400` will delete the free time tag from both the first and second person.

Editing a person : edit

Replaces the specified field(s) of an existing person in Dormie with the new input(s).

Format: `edit INDEX [n/NAME] [p/PHONE_NUMBER] [e/EMAIL] [r/ROOM_NUM] [t/TELEGRAM] [d/DORM_TAG] [ft/FREE_TIME_TAG]...`

Notes on format:

- Edits the person at the specified INDEX. The index refers to the index number shown in the displayed person list.
- The index must be a positive integer e.g. 1, 2, 3
- Minimum 1 parameter must be specified

Command usage examples:

- Edits the name and room number of the 1st person to be Alex and 05-11 respectively: `edit 1 n/Alex r/01-05-11`

Important Note

- If `freeTimeTags` are edited, the person's `freeTimeTags` will be replaced with the new set of `freeTimeTags`. Example:
 - Let Joe have a `freeTimeTag : Mon:1300-1400` and have the index 1:
 - `edit 1 ft/` : Will delete the existing `freeTimeTags`
 - `edit 1 ft/Tue:1300-1400 ft/Wed:1300-1400` Will replace the existing *Monday* tag with the *Tuesday* and *Wednesday* tag.
- The edit command can be done on multiple person when the change is only done on dorm tag and / or free time tags.

End the application : exit

Closes the application.

Format: `exit`

Filter the contact list by name : find

View all persons with name containing the given keyword.

Format: `find KEYWORD`

Provide more detail on existing commands : help

Show a link to the User Guide.

Format: `help`

Listing all persons in Dormie : list

List all persons in Dormie and their details.

Format: `list`

Important Note

- This command can be used to "reset" the view after using `find`.

Check who is free : whoisfree

View all persons that are available on the specified day and time.

Format: `whoisfree DAY:TIME`

- `DAY` is from Mon-Sun
- `TIME` is 24-hour time format

Command usage examples:


- `whoisfree Mon:1300`

Saving the data

Dormie data are saved in the hard disk automatically after any command that changes the data. There is no need to save manually.

Editing the data file

Dormie data are saved automatically as a JSON file [JAR file location]/data/dormie.json. It's possible for advanced users to update data directly by editing that data file.

 Warning: If the data file is edited wrongly, there is no guarantee that Dormie will behave as expected.

FAQ

Q: How do I transfer my data to another Computer?

A: Install the app in the other computer and overwrite the empty data file it creates with the file that contains the data of your previous Dormie home folder.

Known Issues

1. **When using multiple screens**, if you move the application to a secondary screen, and later switch to using only the primary screen, the GUI will open off-screen. The remedy is to delete the `preferences.json` file created by the application before running the application again.

Glossary

Term	Definition, Examples
Command	Instruction provided by a user to specify the desired action or change to be performed by an application.
Command Line Interface (CLI)	Text-based interface used to interact with the application by typing commands into a command box.
Graphical User Interface (GUI)	User interface that allows users to interact with graphical icons and visual indicators, use graphical elements such as windows, buttons, menus, and dialog boxes to facilitate user interaction with the application.
JavaScript Object Notation (JSON)	Lightweight data interchange format that is easy for humans to read and write and easy for machines to parse and generate. JSON is based on key-value pairs and data structures, making it a popular format for representing structured data in web development and other programming contexts.
** Web Browser **	Software application used to access information on the World Wide Web. Examples include Google Chrome, Mozilla Firefox, and Microsoft Edge.
Document Viewer	Software application used to view, read, and interact with documents in various formats, such as PDF, Word, Excel, and PowerPoint files. Examples include Adobe Acrobat Reader, Microsoft Word, and Google Docs.
Java	General-purpose, class-based, object-oriented programming language designed to have as few implementation dependencies as possible. Java is widely used for developing applications, including desktop, web, and mobile applications.
MacOS	Operating system developed by Apple Inc. for its Macintosh line of computers. MacOS is known for its user-friendly interface, stability, and security features.
Windows	Operating system developed by Microsoft Corporation for personal computers. Windows is known for its graphical user interface, multitasking capabilities, and compatibility with a wide range of software applications.
Home Folder	Main directory or folder where an application is installed or resides. The home folder typically contains the application's executable files, configuration files, and data files.
Terminal	Command-line interface used to interact with the operating system by typing commands. The terminal allows users to execute commands, run scripts, and perform various system tasks.
Parameter	Value or variable that is passed to a command or function to specify the desired action or behavior. Parameters are used to customize the behavior of commands and functions based on user input.

