Teh Hao Rui Marcus - Project Portfolio

Project Horo - Overview

Our group of 5 software engineering students were tasked to enhance or morph a basic command line interface Address Book application for our Software Engineering project. We chose to morph it into a personal time management helper app called **Horo**.

Horo enables users to keep tracks of their upcoming tasks and events, reminding users when a deadline or event draws near, and users can also share their data through iCalendar (ICS) files.

This is how the main page of our app looks like:

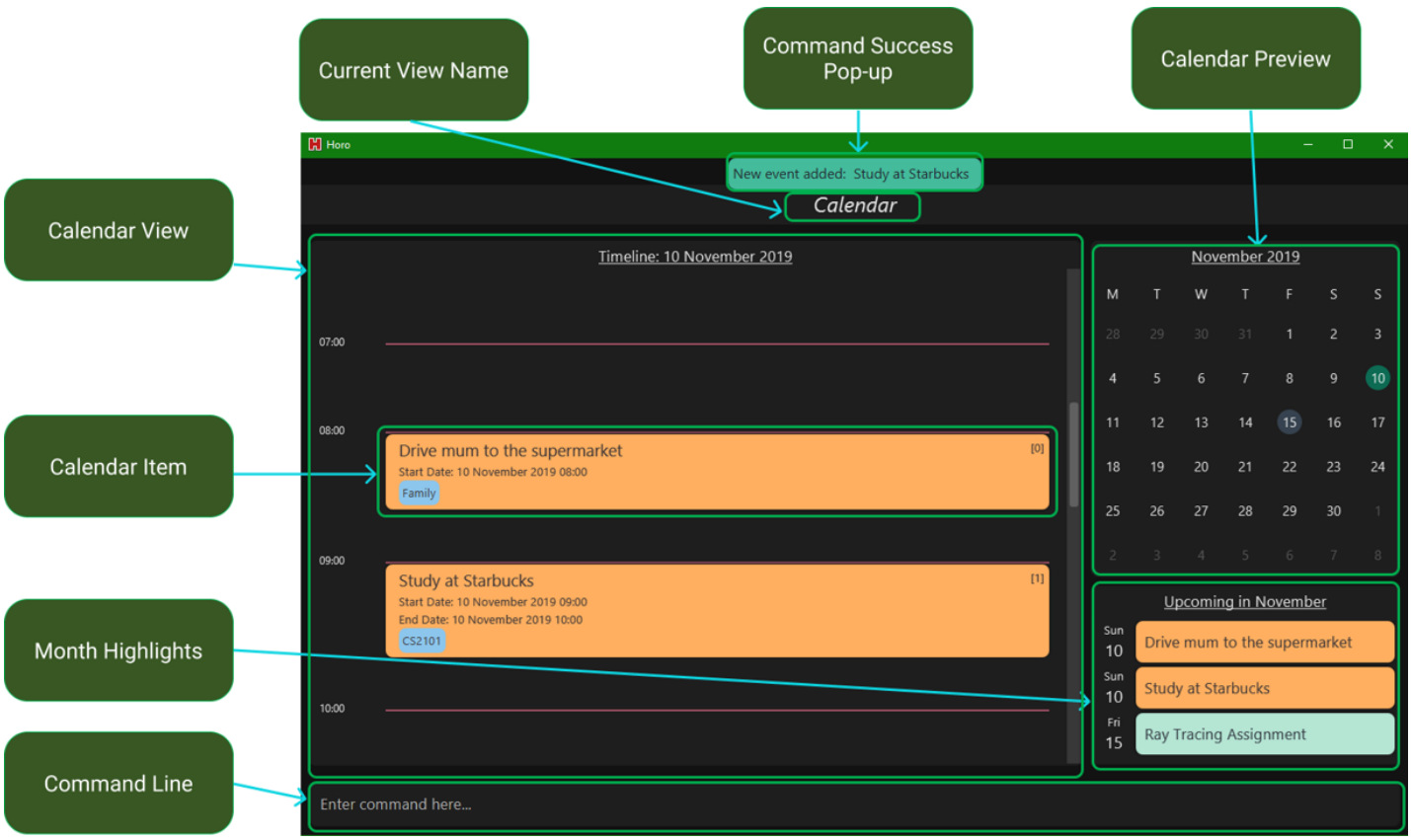

Figure 1. Horo's Graphical User Interface (GUI)

In this project, my role was to design and implement the code for the export and import commands. The following sections illustrate these enhancements in more detail, including some of the relevant documentation I have added to the User Guide and Developer Guide in relation to these enhancements.

Note the following symbols and formatting used in this document:

`export`	A grey highlight (called a mark-up) indicates that this is a command that can be inputted into the command line and executed by the application.
`IcsExporter`	A text with a grey highlight as well as a bold font indicates a component, class or object in Horo.

Summary of Contributions

This section summarizes my contributions to the team project, including coding and documentation.

Enhancement added: The ability to import and export **Horo**'s Event and Task data through ICS files.

- How to use it:
  - The command `export` allows users to export **Horo**'s data into an ICS file. This file will be in the same directory as the **Horo** app.
  - The user may use the command `import <filepath>`, where `<filepath>` is the path to the ICS file, or any other **Horo** save file, in order to import its data to the current app.
- Justification:
  - If users want to share their event details with their friends, they can easily do so by exporting **Horo**'s save data, sharing the ICS file, and having their friends import the save data from the file.
- Highlights:
  - As ICS is a widely used format for exporting and importing calendars into various calendar apps, our app can also import and export data to and from other calendar apps.
  - Although there are existing libraries for ICS files, my code is not dependant on any of them, as I did not want to introduce bugs into our app, thus I wrote a custom parser that better suited our needs.

Code Contributed: I mainly contributed code to the **`ics`** package of our repository. To view a breakdown of my contributions to the repository, click here
(https://nus-cs2103-ay1920s1.github.io/tp-dashboard/#search=marcusteh1238&sort=groupTitle&sortWithin=title&since=2019-09-06&timeframe=commit&mergegroup=false&groupSelect=groupByRepos&breakdown=false)
!

Other contributions:

- Project Management
  - Managed the release of v1.1.
  - Came up with the code integration workflow.
- Documentation:
  - Added information about the ICS package to the User Guide and Developer Guide. (Pull Requests #11 (https://github.com/AY1920S1-CS2103T-F12-1/main/pull/11), #98 (https://github.com/AY1920S1-CS2103T-F12-1/main/pull/98))
- Team:
  - Reviewed Pull Requests with non-trivial review comments: #51 (https://github.com/AY1920S1-CS2103T-F12-1/main/pull/51), #55 (https://github.com/AY1920S1-CS2103T-F12-1/main/pull/55)
  - Reported bugs found on Horo in the Discord server.
- Tools:
  - Integrated Travis Continuous Integration to the team repository.
  - Integrated a Github webhook to the team's Discord server.

Contributions to the User Guide

> As part of the requirements, we had to update the original User Guide with instructions regarding how to use our new features. This section contains an excerpt from the User Guide showing how to use the export command. This shows how I am able to give an understandable guide to new users of Horo.

---

iCalendar (ICS) Integration

Horo stores data in the ICS format. Files saved in this format have the extension `.ics`. This allows for data to be imported from and exported to other calendar applications that also use the `.ics` format.

This means you can export and import your calendar events and todo items straight from Google Calendar into Horo, and vice versa! It also means that you can export and import your save data between different machines running Horo.

Export Current Data

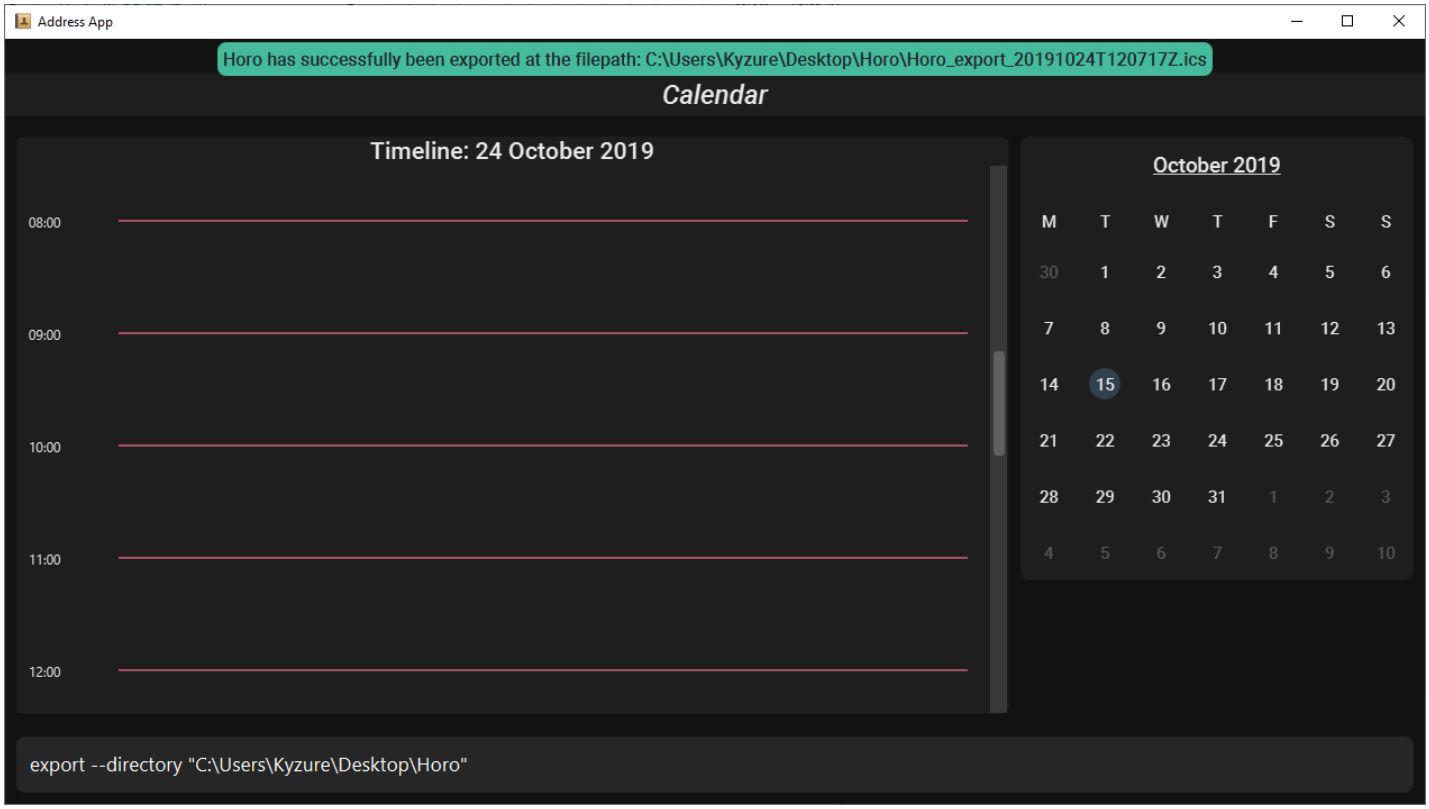

Figure 2. Export Command

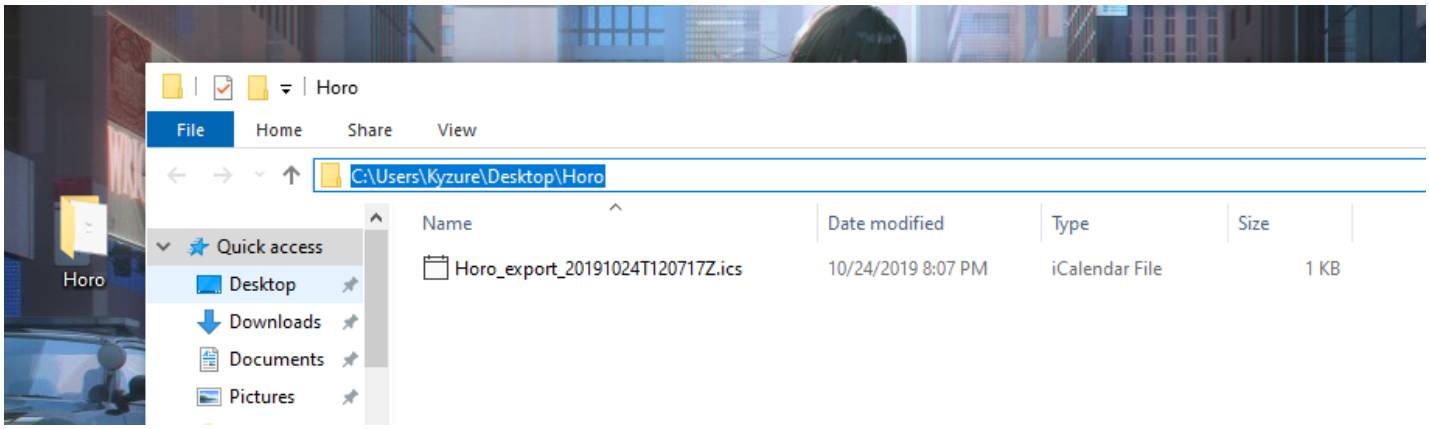

Figure 3. Exported File

The `export` command exports your current calendar as an ICS file to the specified directory. If no directory is specified, the file will be created in the same directory as Horo.

If you want to export the ICS file to a certain directory like your desktop, all you have to do is specify it with the **`--directory`** parameter.

Command Format:

```
export
export --directory DIRECTORY
```

Example:

```
export --directory "C:\Users\USER_NAME\Desktop\Horo"
```

Import other ICS Data

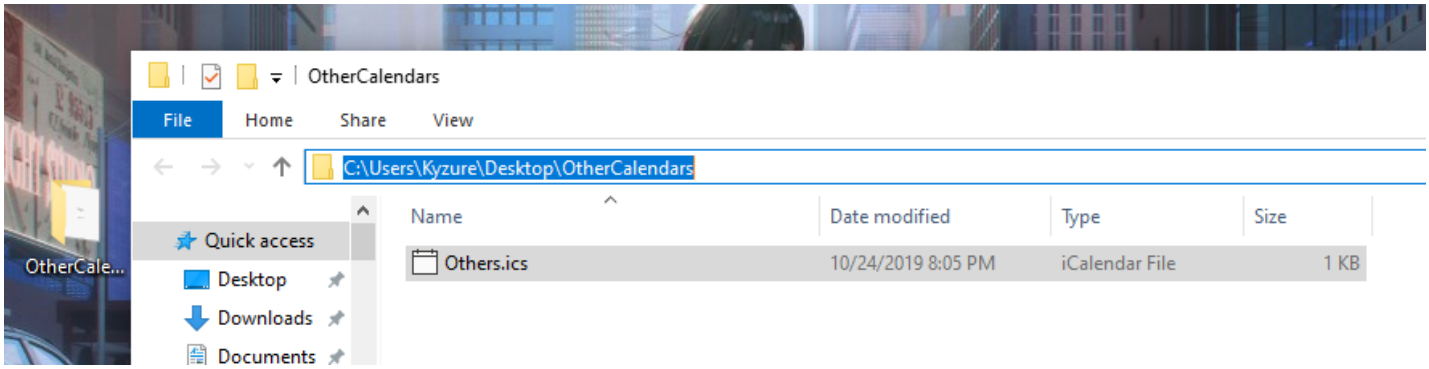

Figure 4. Imported File

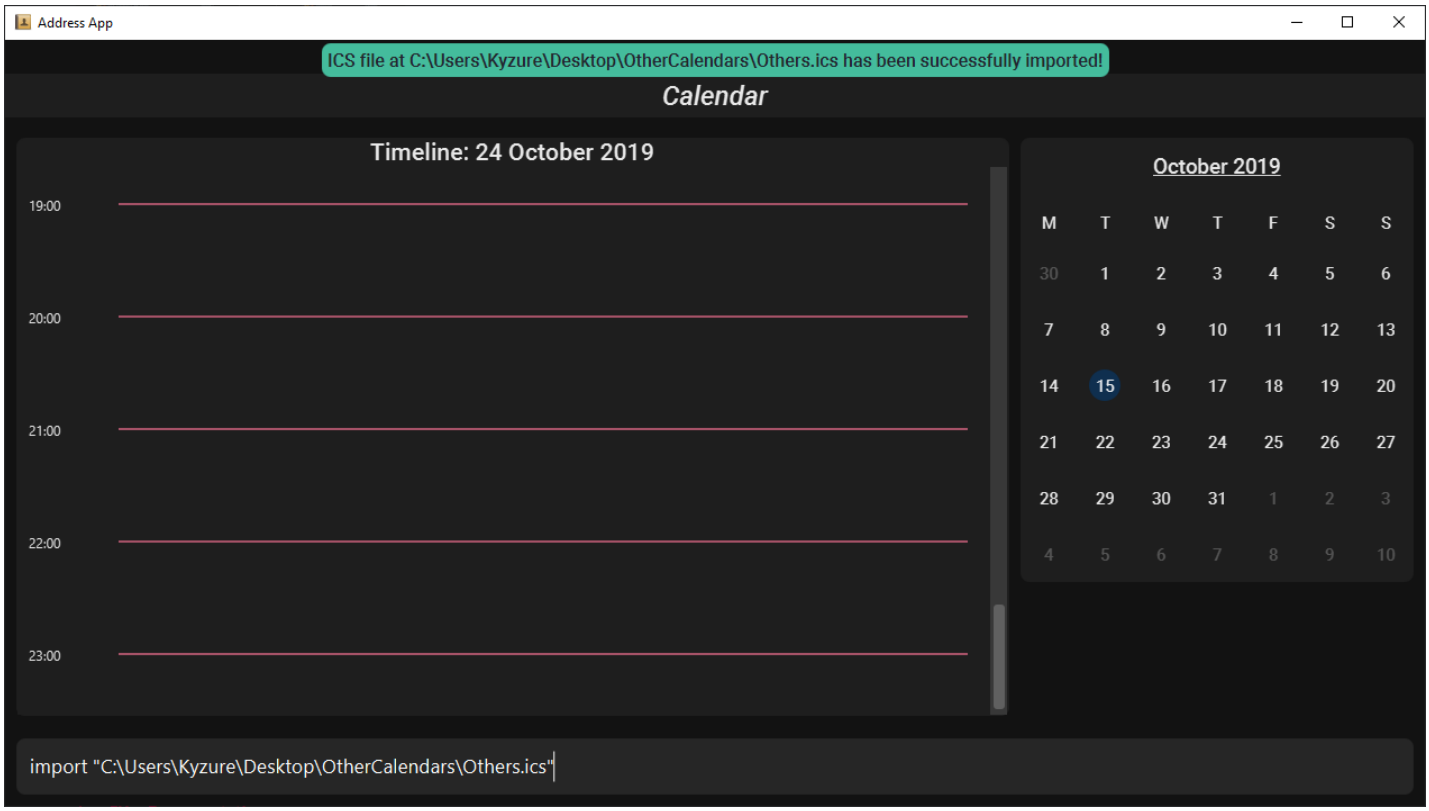

Figure 5. Import Command

The `import` command imports an ICS file from the specified filepath.

Command Format:
`import FILEPATH`

Example:
`import "C:\Users\USER_NAME\Desktop\OtherCalendars\Others.ics"`

Contributions to the Developer Guide

> The following section shows my additions to the Horo Developer Guide for the import and export features, showing how I can write documentation that can be useful to other developers.

Ics Component

API : `IcsParser.java`
(https://github.com/AY1920S1-CS2103T-F12-1/main/src/main/java/seedu/address/ics/IcsParser.java)

The ICS component is made up of 2 main sub-components: ICS file parser, and ICS file exporter.

The file parser makes use of a custom parser that converts files with the `.ics` file extension to `EventSource` and `TaskSource` objects in Horo.

Here is an overview of how the ICS component looks like:

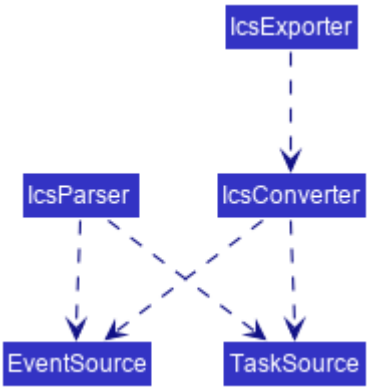

Figure 6. ICS Component Architecture

Right now, this is how the **`IcsExporter`** class exports Horo's **`EventSource`** and **`TaskSource`** data. Notice that the file is created onnly when it is known that the directory provided by the user is valid. This is to prevent extra uncaught errors being thrown.

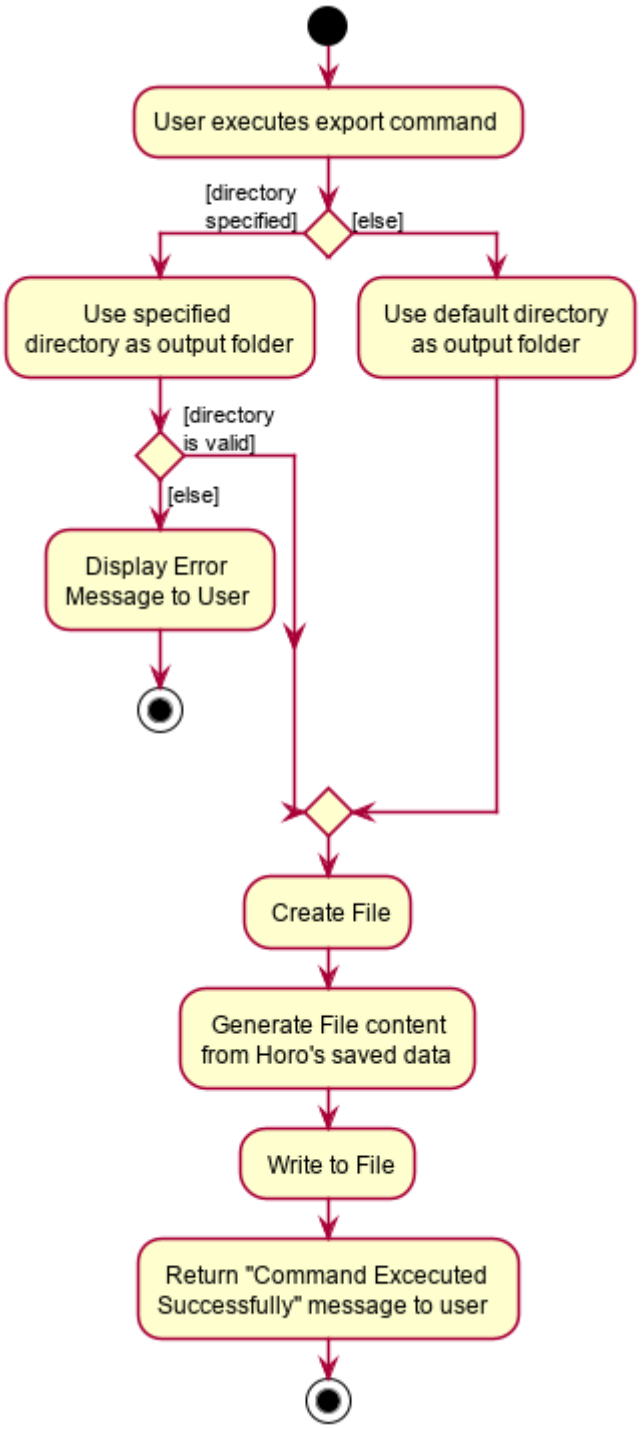

Figure 7. Activity Diagram of an Export Command

In order to generate the file content from Horo's saved data, the file exporter uses the **`IcsConverter`** class to convert **`EventSource`** and **`TaskSource`** objects stored in the **`ModelManager`** singleton object into their ICS String representations.

They will then be concatenated together using a **`StringBuilder`** object. Boilerplate information will be added at the start and end of the save file to make the file valid to be imported to other Calendar applications.

Check out the iCalendar Wiki Page (https://en.wikipedia.org/wiki/ICalendar) for more information on the specifications.

- Can export Horo's save data as a file The `ICS` Component,with a `.ics` extension.

- Can import other Horo's save data from a `.ics` file.

Design Considerations

Aspect: Handling of Horo TaskSource and EventSource conversion to ICS Strings

- **Alternative 1 (Current Choice):** Use of a separate class `IcsConverter` to convert `TaskSource` and `EventSource` objects their ICS string representations.
  - Pros: Adherence to Single Responsibility Principle, decouples `IcsExporter` from the `TaskSource` and `EventSource` classes, and keeps code reusable and scalable.
  - Cons: Not consistent with Object-Oriented Programming structure.
- **Alternative 2:** Create a common `IcsConvertible` Interface for `TaskSource` and `EventSource` to implement a `toIcsString()` function.
  - Pros: Adheres to Object-Oriented Programming structure.
  - Cons: Hard to reuse functions and modify code.

Alternative 1 was chosen eventually, as I felt that it is more important to adhere to the Single Responsibility Principle and keep all code relevant to converting objects to ICS Strings in the same class.

This further makes it easier for future debugging, and makes adding new exportable objects a lot easier as there are common functions that can be used.