Trabajo Práctico Especial: to-do list

Objetivo

Diseñar e implementar una aplicación de línea de comando para la gestión de tareas en listas aplicando los conceptos sobre diseño orientado a objetos adquiridos en la materia.

Descripción funcional

Para ejecutar la aplicación se debe invocar el siguiente comando:

```
$> ruby main.rb
```

La aplicación se mantendrá activa en primer plano hasta que se ingrese el comando **`exit`**.

1. ***Agregar tareas***

```
add Completar la autoevaluación
Todo [1: Completar la autoevaluación] added.
add Estudiar para el parcial
Todo [2: Estudiar para el parcial] added.
add
Invalid task
```

Se genera un número entero identificador autoincremental empezando en uno para cada una de las tareas. Si no se recibe un nombre, la tarea no se agrega y se informa en pantalla “Invalid Task”.

2. ***Listar tareas***

```
list
 all
 1    [ ]        Completar la autoevaluación
 2    [ ]        Estudiar para el parcial
```

La tarea siempre empieza en estado pendiente.

3. ***Completar una tarea***

```
complete 1
Todo [1: Completar la autoevaluación] completed.
```

```
list
 all
 2    [ ]        Estudiar para el parcial
 1    [x]        Completar la autoevaluación
complete 1
Task 1 already completed
complete 5
Invalid task
```

Las tareas completadas se listan a menos que hayan sido archivadas.

4. *Archivar tareas completadas*

```
ac
All completed todos have been archived.
list
 all
 2    [ ]        Estudiar para el parcial
```

5. *Uso de fechas*

Las tareas pueden tener una **fecha de vencimiento**. De este modo se pueden listar sólo las tareas que ya vencieron, las que vencen en el día de hoy o las que están por vencer.

Las tareas siempre se listan en orden cronológico.

Para las fechas usar el formato **DD/MM/YYYY**. En caso de que la fecha sea ayer, hoy o mañana, utilizar los términos "**yesterday**", "**today**" y "**tomorrow**" en lugar de la fecha completa (para facilidad de uso).

```
add Resolver TP1 due tomorrow
Todo [3: Resolver TP1] added.
add Resolver TP2 due 24/03/2018
Todo [4: Resolver TP2] added.
list
 all
 2    [ ]             Estudiar para el parcial
 3    [ ]  tomorrow   Resolver TP1
 4    [ ]  24/03/2018 Resolver TP2
list due tomorrow
 all
 3    [ ]  tomorrow   Resolver TP1
list due this-week
 all
 3    [ ]  tomorrow   Resolver TP1
 5    [ ]  24/03/2018 Resolver TP2
```

De la misma forma, **`list overdue`** lista todas las tareas vencidas y **`list due today`** lista las tareas que vencen el día de hoy.

6. Listas de tareas

Para agrupar a las tareas, se las pueden ubicar en listas con nombre. Para agregar una tarea que pertenece a una lista, agregar **`+nombreLista`** antes del nombre de la tarea, donde el nombre no puede contener espacios.

```
add +POO Resolver TP1
Todo [6: Resolver TP1] added.
add +POO Resolver TP2
Todo [7: Resolver TP2] added.
add +Personal Pagar tarjeta de crédito due tomorrow
Todo [8: Pagar tarjeta de crédito] added
list
 all
 6    [ ]             +POO Resolver TP1
 7    [ ]             +POO Resolver TP2
 8    [ ]  tomorrow   +Personal Pagar tarjeta de crédito
list group
 POO
 6    [ ]        Resolver TP1
 7    [ ]        Resolver TP2

 Personal
 8    [ ]  tomorrow   Pagar tarjeta de crédito
list +POO
 POO
 6    [ ]        Resolver TP1
 7    [ ]        Resolver TP2
```

7. Serialización de tareas

```
save file_name
```

Guarda todas las tareas en el archivo `file_name`, usando YAML, de forma tal que luego puedan ser recuperadas. Si el archivo existe, lo pisa. Si no puede escribir el archivo, informar el error.

```
open file_name
```

Recupera todas las tareas desde el archivo `file_name`. Al cargar el archivo elimina las tareas actuales. luego de que el usuario acepte el mensaje de confirmación. Si el archivo no existe informa el error y no hace nada.

8. Opcionales

Además de la funcionalidad obligatoria ya enunciada, se ofrece también funcionalidad adicional, de implementación opcional, que en caso de realizarla otorgará puntos extras en la calificación del trabajo.

8.1 Búsqueda de tareas por nombre (+1.00)

```
find resol
 6    [ ]            +POO Resolver TP1
 7    [ ]            +POO Resolver TP2
```

El comando **`find`** ***`text`*** busca todas las tareas que en su descripción contengan el texto ***`text`***. No debe ser *case sensitive*. No se deben contemplar expresiones regulares o "*wildcards*".

8.2 Valores por defecto (+0.50)

```
set date_task date
```

Indica que, a partir de la ejecución del comando, todas las tareas que se agreguen tendrán una fecha de vencimiento por defecto, salvo que al agregarla se indique una.

Los posibles valores para ***`date`*** son: una fecha con forma **`DD/MM/YYYY`**, **`today`**, **`tomorrow`** o en blanco. En caso de estar en blanco las tareas no tendrán una fecha de vencimiento por defecto.

```
set group group_name
```

Indica que, a partir de la ejecución del comando, todas las tareas que se agreguen tendrán un grupo por defecto. El grupo puede existir o no, y si está en blanco indica que ya no hay grupo por defecto.

Armado de grupos y entrega

Los alumnos deben informar la conformación del grupo y la fecha de final elegida en el Foro de Discusión de Campus ITBA hasta el **29/03/2018 17:00 ART.**

Cada grupo deberá realizar la entrega mediante la actividad correspondiente en Campus ITBA **antes del 05/04/2018 17:00 ART** entregando un archivo comprimido .zip conteniendo:

- **El código fuente** (main.rb y otros archivos fuentes)
- **El diagrama de clases correspondiente** usando la herramienta de diseño de la cátedra PlantUML o similar
- **Un archivo de respaldo de ejemplo** (.YAML) que ya cuente con tareas pendientes, completadas y archivadas, con diferentes fechas de vencimiento y pertenecientes a distintas listas
- **Un informe** de no más de dos páginas que indique todas las decisiones de diseño que fueron tomadas durante la implementación, justificando el diseño de clases y métodos elegidos.

Criterios de evaluación y calificación

No se aceptará el uso de bibliotecas de terceros, a excepción de la biblioteca estándar de Ruby. El código debe ser íntegramente de autoría propia. El uso de bibliotecas no autorizadas implicará la desaprobación.

Si un trabajo presenta errores de ejecución al invocar alguno de los comandos arriba listados, el mismo será reprobado.

El programa no debe abortar por ningún motivo y ante cualquier error se debe mostrar un mensaje adecuado.

Para la evaluación y calificación del trabajo especial se considerarán:

- el correcto funcionamiento del programa (recordar el uso de métodos de prueba de software)
- la modularización realizada
- el correcto diseño de las clases
- la separación del frontend y el backend
- la claridad del código
- el cumplimiento de las reglas de estilo de programación dadas en clase

Todas las pruebas serán realizadas en Pampero. Verificar que la versión de Ruby que utilizan para el desarrollo coincide con la de Pampero.

Dudas sobre el TPE

Si bien el enunciado contempla la funcionalidad completa a desarrollar es normal que surjan dudas acerca de cómo interpretar ciertos casos. O que una consigna genere más de una posible solución, por lo que es importante que analicen bien el enunciado, y ante cualquier duda pregunten. Las dudas sobre enunciado deben volcarse al **Foro de Discusión del Campus ITBA**.