Trabajo Practico Final

Programación orientada a objetos

Al comenzar el trabajo practico nos pareció correcto utilizar una implementación similar a la utilizada en ruby con pequeños cambios al “front-end” para lograr una interfaz visual. Por esta razón generamos las clases Task, TaskHolder y StringDate que utilizaban los mismos nombres y contenían las mismas funciones que en ruby.

A medida que avanzamos con el proyecto, quisimos crear un conjunto de comandos que siguieran una interfaz Command y que tomen las acciones tomadas en la interfaz y las apliquen a las clases del back-end. Esto resulto complicado debido a la diferencia entre las clases base (Array, list, task, etc) y las Clases observable que requerían de otra inicialización y producían que fuera difícil utilizar los datos en sus estados base.

Al encontrarnos con esto decidimos que lo más conveniente sería tener únicamente la interfaz visual como front-end y en ella incluida toda la lógica de las acciones tomadas en ella y por el lado del back-end las clases Task, TaskHolder y StringDate.

El diseño del archivo PrincipalView.fxml y su controlador PrincipalViewControler resulto complejo al comienzo debido a la falta de información devuelta al intentar crear un FXMLLoader que funcione y la dificultad de entender a que se debía cada error. Fue por esto que para generar el PopUp utilizado para editar tareas decidimos generarlo atravez de un meto dentro del controlador de PrincipalView ya que esto probo ser una tarea mas simple pero igualmente útil.

Mas alla del archivo .fxml creamos la clase ObservableTask la cual es utilizado por PrincipalView para mostrar los contenidos de cada task siguiendo los requisitos de JavaFX. Para esto creamos un método que recibiera nuestras task regulares y transformara sus campos en formatos utilizables mientras que a su vez mantiene una copia interna de la tarea para manejo de datos dentro del Holder.

Para finalizar creamos los métodos de guardado a travez de la aplicación de la interfaz “serializable” que nos permitió guardar todos TaskHolder y todos sus contenidos de manera muy sencilla con el uso de ObjectImputStream y ObjectOutputStream.

En conclusión el trabajo presento algunas complicaciones pero la mayoría fueron fácilmente resueltas repasando temas dados en las clases teóricas o prácticas, o foros en internet y en el caso de que todo lo anterior fallara es porque había una solución más simple que simplificaba el trabajo.