TimeTracker

Diseño del Software

2020-2021

Daniel Cendagorta-Galarza Friend (NIU: 1459480)
David Martí Zaragoza (NIU:1394933)
Joan Mata Pàrraga (NIU: 1533089)
Ismael Pajuelo Berjano (NIU: 1456941)

Índice:

1.- Introducción:

Durante estas semanas, hemos continuado con el desarrollo del software del TimeTracker. El principal desafío que hemos tenido en este *milestone* ha sido solucionar los problemas en la compilación y ejecución del código anteriormente funcional, ya que se ha tenido que invertir gran parte del tiempo en solucionarlo.

Modificando el UML, reestructuramos el proyecto añadiendo 2 clases nuevas, necesarias para el *milestone 2*: Tag y TotalTime, además de pequeñas modificaciones en otras clases anteriores.

Para hacer los loggers, fue necesario crear un fichero .xml para poder configurarlos. Una vez configurado el fichero, se tuvo que sustituir todos los System.out.*println* de las clases existentes por los loggers. Al mismo tiempo, se añadían nuevos tipos de logger, tanto en las clases existentes como en las nuevas.

Añadimos asserts en la mayor parte de las clases para asegurar el correcto funcionamiento de éstas y detectar la mínima incongruencia en el funcionamiento del programa. Algunos asserts más generales se usan como base en otras clases para hacer asserts más específicos para su propia función. Aunque en un principio sólo se crearon los asserts para las clases del *milestone 1,* también se incluyeron asserts en la creación de las clases nuevas del *milestone 2.*

A lo largo de la práctica se fueron añadiendo, modificando y perfeccionando los comentarios de todas las clases.

2.- Estilo de codificación:

Uno de los puntos del *milestone 2,* consiste en cambiar el estilo de codificación por el de *checkstyle.*

Mantiene la misma idea, solo que se ha tenido que ajustar en algunos puntos para que no de conflicto con *checkstyle,* quitando las "_" y poniendo los tres primeros caracteres de las variables y las funciones refiriendo a la clase a la que pertenecen, siendo de esta manera más entendible.

3.- Trabajo realizado:

Para plantear el *milestone 2*, comenzamos haciendo el JSON que no realizamos en la entrega anterior. De eso se encargó generalmente Daniel.

Referido a este hito, comenzamos por instalar el plugin de *checkstyle* dentro de la interficie. Joan se vio obligado a modificar el nombre de la gran mayoría de variables. Algunas funciones también, aunque las clases estaban declaradas según las pautas. Nuestro estilo de codificación sigue la misma idea, aunque con modificaciones, los tres primeros caracteres de las variables y las funciones se refieren a que clase forman parte, para que al realizar funciones similares en diferentes clases podamos diferenciarlas con claridad.

Seguidamente nos pusimos a realizar la parte del *logging*, Daniel e Ismael desempeñaron esa tarea y crearon las variables correspondientes, el fichero *.xml* necesario, etc. Para esto, fueron sustituidos todos los *System.out.println* de información del código por *logger.info*, todas las excepciones por *logger.error* y todos los *System.out.println* cuya función era debugar se cambiaron por *logger.debug*. Se añadieron, también, *logger.trace* en las funciones que requerían de ello.

Al mismo tiempo tratamos de codificar el capítulo referido al *Desing by Contract*, Joan creó la función para los invariantes que hemos aplicado al inicio y final de todas las funciones públicas y al final de los constructores, seguido de algunas pautas e ideas de David. También introdujimos algunos asserts en las zonas más delicadas del código.

Hemos tenido ciertos problemas con la compilación y ejecución del código, llevados a tener que centrar nuestro esfuerzo en debugar nuestro programa, para hacer que el resultado fuese el correcto al percibir errores en el cálculo del tiempo total.

Sobre las clases nuevas, David creó la clase *Tag* apoyándose también en algunas ideas de Joan. Ésta los problemas que dio fue que el equipo no estaba seguro de si se podían importar funciones más elegantes que no fueran del propio Java, pero aún así la implementación final, aunque aparentemente más rudimentaria, es completamente funcional y case-insensitive.

La segunda nueva clase *TotalTime* fue creada conjuntamente por Joan y David, ésta abarca todos los posibles casos de intersección de tiempo relevantes para la función. Pero aún no es del todo funcional y todavía nos queda algo de trabajo para debugarla y arreglarla.

4.- Conclusiones:

En esta segunda entrega se ha notado la mejora de compenetración y comunicación del grupo, y los problemas principales que se nos han presentado esta vez han derivado de otros lados, como arreglar el código, cosa que nos llevó bastante tiempo impidiéndonos avanzar el resto de la práctica.