AD-2. TAREA EN EQUIPO JAVADOC Y JUNIT

ENTORNOS DE DESARROLLO

1º DAM/DAW - UNIR

Laura Vinseiro Gutiérrez (Lauu92)
Patricia Victoria Sanz López (pvicSL)
Manuel Alejandro López Ortega (Alejandrounir)
Gregory López Gómez (GregorLopez)

URL del repositorio:
https://github.com/pvicSL/REPOACT2JavadocJUnitGLMP.git

Introducción: breve descripción de la actividad y su propósito.

La actividad propuesta consiste en realizar un pequeño proyecto de desarrollo en grupo. De acuerdo con los requerimientos, los integrantes del equipo deben coordinarse para desarrollar una serie de clases, implementando el código con IntelliJ. Además del desarrollo del código, se debe hacer uso de GIT y GitHub, de modo que cada integrante trabajará en una rama propia durante la implementación de su clase, e irá realizando los *commits* necesarios a la rama máster en un repositorio compartido del proyecto.

Otro de los requerimientos es la realización de pruebas, mediante JUnit, de los métodos de las clases creadas.

Además del desarrollo, implementación y realización de pruebas de las clases y sus métodos, el equipo tiene la tarea de documentar las clases y métodos creados mediante Javadoc, para lo cual deberán incluir los comentarios pertinentes y generar la documentación correspondiente en formato html, que se debe adjuntar en la entrega del proyecto.

Metodología: pasos realizados, uso de GIT y herramientas empleadas.

En primer lugar, hemos repartido las tareas decidiendo qué clase del trabajo desarrollaría cada compañero. Una vez decidido esto, se ha procedido a programar la clase Main, con un menú para elegir qué clases de la calculadora se desea elegir en una rama de la rama *master*. Tras completar la clase Main, hemos unido la rama en la que se ha desarrollado este trabajo con la rama *master*.

A la hora de trabajar de forma individual, cada participante ha sacado una rama propia de la rama *master* para poder trabajar sin conflictos en nuestra clase asignada y, cuando hacíamos un cambio significativo, hemos comprobado que no había ningún tipo de conflicto con la rama *master* para posteriormente hacer un *merge* a esta. Cuando terminamos las distintas clases de la calculadora, utilizamos la misma metodología con las ramas para realizar las pruebas unitarias.

El sistema de trabajo acordado entre los miembros del grupo ha establecido dos canales de comunicación, un grupo de WhatsApp y el Discord de DAW, para organizar el reparto de tareas, poner dudas en común, indicar a los compañeros cuándo se ha realizado o se va a realizar un *commit* o un *push* y ajustar plazos de entrega de cada tarea.

Además de apoyarnos en los materiales facilitados en la materia, los integrantes del grupo también hemos utilizado diversos tutoriales y webs de consulta para tratar de comprender mejor el proceso de documentación y realización de pruebas. Esto nos ha ayudado a profundizar en los puntos más problemáticos del trabajo, es decir, aquellos en los que quizás teníamos menos conocimientos.

Hemos utilizado Git para conectarnos al repositorio y actualizar el repositorio de GitHub, aunque en ciertas ocasiones hemos realizado el mismo proceso mediante IntelliJ.

Tal y como se solicitaba, hemos empleado IntelliJ para desarrollar el código en Java.

Al tener finalizado el grueso del proyecto, hemos dedicado una última reunión a organizar tareas de repaso y corrección. Estas tareas han estado enfocadas a revisar nuevamente todo el proyecto para detectar posibles errores, dotar de coherencia tanto al código como la documentación, maquetar el fichero de trabajo, etc.

Problemas y soluciones: principales retos encontrados y cómo se abordaron.

- En la clase Producto, en el método *potencia*, si la base es muy grande da como resultado *Infinity*. Se ha optado por indicar en un comentario el resultado de la prueba, de acuerdo con lo indicado en algunas sesiones de la asignatura, ante la imposibilidad de incluir una solución en el código, dado que aún somos alumnos de primer curso y no hemos encontrado esta casuística hasta el momento.

- En la clase Cociente, los métodos *división real* y *división enteros* no pueden tener como divisor el cero. Para solucionarlo, se ha implementado un bucle *while* en ambos métodos, cuya función es evitar precisamente que el usuario introduzca un cero por consola. El bucle no se detiene hasta que el usuario introduce un número diferente.

- En varias clases se piden métodos que únicamente contemplan la posibilidad de introducir números enteros. Como ejemplo de solución en caso de que el usuario no introduzca la información correcta, hemos introducido un bloque *Try/Catch* que captura la excepción *InputMismatchException* en algunos casos. Mientras que en otros se captura la excepción *NoSuchElementException* lanzada por el método si se introduce un número real. También es una excepción válida si se introduce cualquier otro tipo de dato que no sea un número entero. Aunque nos funciona, debemos recalcar que lo hemos hecho como trabajo de investigación y que, al no haber visto aún en profundidad el tema de las excepciones, no sabemos si es del todo correcta la solución que hemos planteado.

- Como hemos dicho, este método también se ha utilizado en otros casos para evitar que el programa aborte los procesos si el usuario introduce un carácter no esperado (letras en lugar de números al elegir una opción de un menú, por ejemplo, o letras en lugar de enteros al realizar una operación).

- En todo el proyecto, cualquier texto impreso por consola sale sin acentos o ciertos símbolos especiales. Después de revisar todas las opciones básicas y de buscar información hasta la saciedad en foros y similares, no hemos conseguido dar con una respuesta que nos permita solucionar este problema. Finalmente, y para evitar la mala legibilidad de las instrucciones y menús que aparecen por consola, hemos decidido eliminar todos aquellos símbolos que salieran impresos con error. Esto supone que los textos no contienen acentos, por ejemplo.

- Hemos experimentado la opción que nos ofrece IntelliJ para manejar Git desde la propia aplicación, pero nos han surgido varios problemas, como encontrarnos con que no se realizaban las operaciones indicadas. Puesto que no era un requisito necesario para la práctica hacerlo desde IntelliJ, hemos optado por trabajar únicamente con Git cuando surgía este problema. Intentar solucionar los problemas de IntelliJ en este apartado era algo que nos quitaba un tiempo del que no podíamos prescindir y, por tanto, optamos por no indagar más.

Desglose de tareas: descripción de las labores realizadas por cada integrante.

Patricia V. Sanz López: implementación inicial de la clase Main, desarrollo de la clase Producto y testing de la clase Resta. Comentarios para documentación correspondientes. Maquetación final del documento en formato pdf para la entrega de la actividad.

Laura Vinseiro Gutiérrez: desarrollo de la clase Cociente y testing de la clase Producto. Comentarios para la documentación correspondiente. Trabajo para dotar de coherencia y uniformidad a todos los menús de las clases y la información propuesta para el Javadoc.

Manuel Alejandro López Ortega: desarrollo de la clase Resta y testing de la clase Suma. Comentarios para la documentación correspondiente.

Gregory López: desarrollo de la clase Suma (con ayuda de Alejandro al acabar de retocar el menú) y testing de la clase Cociente. Comentarios para la documentación correspondiente.

Conclusiones: reflexión sobre el trabajo en equipo y el aprendizaje.

Durante la realización del trabajo nos hemos encontrado con varios desafíos como es el trato de excepciones a la hora de tanto de realizar las clases como de implementar un test para comprobar las excepciones, pero gracias al trabajo en equipo y a la buena comunicación entre nosotros hemos podido superarlo satisfactoriamente.

Hemos desarrollado conocimientos en la documentación con Javadoc, en la creación de pruebas unitarias con Junit y en la implementación de excepciones para nuestras clases.

Hemos profundizado en el trabajo con Git, ya que es la primera vez que trabajábamos en un proyecto “real” en el que cada miembro del equipo trabajaba en su clase de forma individual para después añadirlo a un todo superior, lo cual nos ha requerido de comunicación continua. Hemos cometido algunos errores con Git, pero estos nos han proporcionado aprendizajes a los no habríamos podido llegar trabajando solo de forma teórica.

La tarea nos ha servido para profundizar en la materia y, sobre todo, para aprender a solucionar problemas buscando documentación adicional y usando referencias de blogs o tutoriales para solventar dificultades. Aunque también nos han surgido dudas a lo largo del trabajo, es cierto que tenemos más claros determinados conceptos y hemos automatizado ciertas partes del flujo de trabajo.

Una de las partes más satisfactorias de esta tarea ha sido, al igual que en el trimestre anterior, el apoyo mutuo del equipo. Darnos consejos, buscar información para los compañeros, explicar lo que uno ha ido haciendo para ayudar a un compañero a entender mejor algún aspecto en el que no tiene conceptos muy claros, o simplemente dedicar tiempo a la tarea y la asignatura, han sido algo habitual durante el trimestre y la clave para que hayamos podido sacar el trabajo adelante.

Notas:

Hemos acordado incluir esta sección adicional en el documento para poder trasladar ciertos comentarios generales, dudas y otros asuntos relacionados con la tarea y su desarrollo.

Duda acerca del formato: durante el desarrollo de la tarea, nos ha surgido el debate sobre si deberíamos igualar las variables de toda la aplicación para dar uniformidad o dejar que la parte de cada programador tenga ligeras diferencias. Hemos decidido unificar el tratamiento del usuario, porque se trata de algo básico, y el uso de acentos por un tema de problemas de configuración de IntelliJ. Entendemos que en los proyectos profesionales habrá una serie de directrices a este respecto, en las que se indiquen unas buenas prácticas de redacción de comentarios, así como los aspectos relacionados con el formato.

Duda sobre presencia de las ramas individuales en GitHub: a medida que avanzábamos en el trabajo, hemos observados que podemos ver las ramas individuales de cada miembro del grupo en los *commits*, aunque no en el repositorio. No tenemos claro a qué se debe esto, o si tiene que ver con la metodología empleada para actualizar el repositorio (trabajo en rama individual, merge con rama *master* y *push*, de acuerdo con lo indicado en el enunciado).

Pruebas con IntelliJ: en este apartado, nos han surgido muchas dudas, al principio no teníamos claro qué funcionalidades debíamos usar para ciertas pruebas, y hemos tenido que acudir a documentación online para entender bien qué hace cada una. Por ejemplo, hemos tenido que buscar información sobre qué funcionalidad usar para las excepciones que lanzaba el programa cuando el usuario introduce un carácter no esperado en alguno de los métodos, pero tenemos la sensación de que la comprensión de lo que estábamos usando, y por qué funciona o no, es más bien superficial. Seguramente se deba a que no hemos abordado aún las excepciones en la asignatura de programación y carecemos de la base teórica necesaria para entender totalmente lo que estamos haciendo. Nos ha parecido relevante comentarlo, ya que esas “brechas” de conocimiento entorpecen el trabajo y provocan que tengamos que ir salvando los obstáculos con cierta inseguridad en lo que estamos haciendo.

Para finalizar, me gustaría incluir algo. Soy Gregory López y, como ya hablé contigo, Luis, por problemas familiares, en la primera parte de mi trabajo (clase Suma) faltaba por retocar alguna cosa y tuvo que retocarla Alejandro y hacer el *push* desde su PC, ya que yo no estuve en mi casa y no pude hacerlo. Quiero agradecer a los tres compañeros que han trabajado conmigo ya que tuvieron que asumir una parte del trabajo que no les tocaba y han sido muy comprensivos conmigo.