Diseño y Programación Orientado a Objetos

Curso 2020-2021

Práctica 1

LSGallos

	Login	Nombre
Alumnos	albert.clarimon	Albert Clarimón Vilardebó
	marc.valsells	Marc Valsells Niubó

	Proyecto	Memória	Nota
Entrega			

Fecha	15 de enero de 2021

Índice

Breve síntesis del enunciado

Esta práctica consiste en la planificación, desarrollo e implementación de un programa el cual gestione, en base una competición de raperos, los participantes, las regiones donde se efectuará la competición en una determinada fase, los perfiles de cada usuario y finalmente dictaminará al vencedor de esos juegos. La práctica está dividida en 4 fases emparejadas dos a dos, es decir las dos primeras fases corresponderán al mismo bloque y las dos siguientes al segundo bloque.

La primera fase consiste en diseñar previamente el Star UML de la fase 2, en esta deberemos separar las diferentes clases al igual que dictaminar que tipo de relación tienen entre ellas. Cabe decir, que el diagrama efectuado en esta fase es un diagrama de orientación pues en la siguiente fase hubo varios elementos que originalmente no se habían tenido en consideración.

La segunda fase, corresponde a la implementación a nivel de programación del Star UML diseñado con anterioridad. En esta fase, se implementará las diferentes clases, al igual que el control de las batallas simuladas, las batallas del usuario, los temas a rapear, el registro de un nuevo participante, la creación de las batallas entre otras muchas cosas menos la opción de generar un perfil, pues esta corresponde a las otras dos fases restantes.

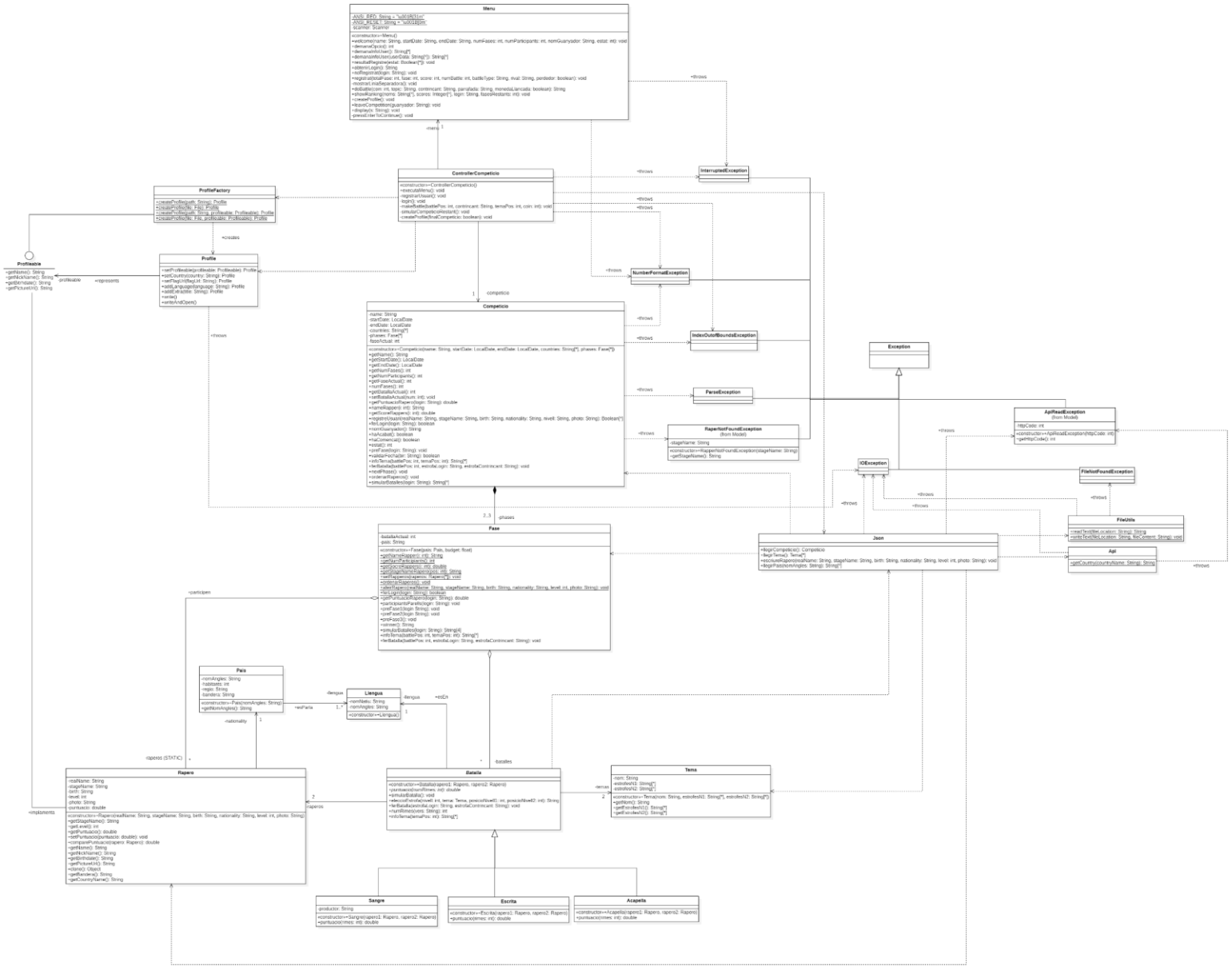
Cabe mencionar también que en esta fase se modifica el Star UML de la fase anterior, pues como ya se había mencionado es muy probable que el original sea relativamente más incompleto que el definitivo.

La tercera fase, al igual que la primera, consiste en diseñar el Star UML, que será necesario para la implementación de la cuarta fase, el cual corresponde a la opción de generar un perfil.

La cuarta y última fase, es el resultado de la aplicación de las fases anteriores juntamente con la agregación de la funcionalidad de generar un perfil el cual contiene entre otras cosas el nombre del país, su bandera y las lenguas habladas. Para realizar esta última modificación es necesario leer de la API que es proporcionada. Finalmente, en esta fase es necesario, como ya se ha vito con anterioridad, la corrección del Star UML incluido en la tercera fase y por consiguiente la finitud de el Star UML, así como del programa.

Asimismo, en esta fase, es necesario realizar una memoria, que debe ser adjuntada junto las dos partes mencionadas anteriormente, la cual debe contener una serie de apartados mencionados en la normativa.

Diagrama de clases UML



Explicación del diagrama de clases diseñado

En el diagrama UML anterior se puede ver que gran parte del código funciona a partir de la clase ControllerCompetició, esta clase solo tiene un método público que es executaMenu el cual es el método que se llamara des del Main para ejecutar, el resto

de los métodos de esta clase son privados dado que solo los llama el método executaMenu. Ese método es encargado de instanciar y llamar métodos de las otras clases para ejecutar el programa según los datos que introduce el usuario por la consola.

La clase ControllerCompetició tiene dos asociaciones, una con la clase Menu, que se encargar de mostrar el texto por consola y pedir los datos al usuario, y la clase Competicio en la cual des de ella se gestiona toda la competición. Además, dado que los datos iniciales de la competición es necesario leerlos de un fichero JSON el ControllerCompeticio tiene una dependencia con la clase Json. Por otra banda ese controller también tiene dependencia con las clases ProvileFactory y Profile dado que son las encargadas de generar el archivo HTML con el perfil de un rapero.

Siguiendo en orden en el diagrama se encuentra la clase Competicio, esta clase tiene información básica sobre la competición en si y además contiene un numero largo de métodos. Esto es dado a la implementación de GRASP, cualquiera información de los raperos, batallas, etc. que queramos mostrar por pantalla tiene que pasar por la clase Competicio por tal de no crear relaciones innecesarias. Además, hay varios métodos para controlar las fases y cuando debemos cambiar de ella.

A continuación, nos encontramos con la clase Fase es se encarga de gestionar los raperos los cuales son de tipo static por hacer más fácil su manejo entre las diferentes fases. Esta clase se encarga de generar y almacenar la información sobre las batallas las cuales pueden ser de 3 tipologías diferentes, según esa tipología se obtendrá una puntuación u otra, esto se ha implementado con herencia y un método extracto.

Para finalizar, tenemos las clases Json, FileUtils y Api para obtener la información de terceros en formato JSON y interpretarla para el programa y viceversa. Además, nos encontramos con las clases de la librería ProfileHelper para poder generar los perfiles ha sido necesario implementar los métodos de Profilable en la clase Rapero.

Opcionales

En el transcurso de la práctica se han usado una variedad de funcionalidades pues eran de vital importancia para el correcto funcionamiento de ella. No obstante, cabe mencionar que nuestro equipo ha implementado una serie de variantes opcionales al fin de poder representar de una forma más realista la práctica.

Una de esas variantes es la posibilidad de parar la ejecución del programa durante una serie de segundos, de esa manera se ha podido representar la cuenta regresiva mostrando de forma pausada la cuenta atrás. Esto se ha usado en el momento de poder dictaminar a un ganador.

Otra de las mejoras que se ha considerado es en el momento de mostrar el ranking, pues se ha diferenciado los participantes que en ese momento pasan de

ronda y los que quedan desclasificados, estos últimos se les puede ver, pues su nombre aparece en rojo como si fuesen eliminados.

Métodos de prueba

Para asegurar el bueno funcionamiento de la practica a medida que se iba picando y elaborando el código ese mismo se iba probando mediante la consola para verificar su correcto funcionamiento. Además, también se ha usado la herramienta debugger para ver el contenido de las clases y que los métodos funcionaran correctamente mientras estos se van ejecutando con la opción de ir visualizando el estado del código paso a paso.

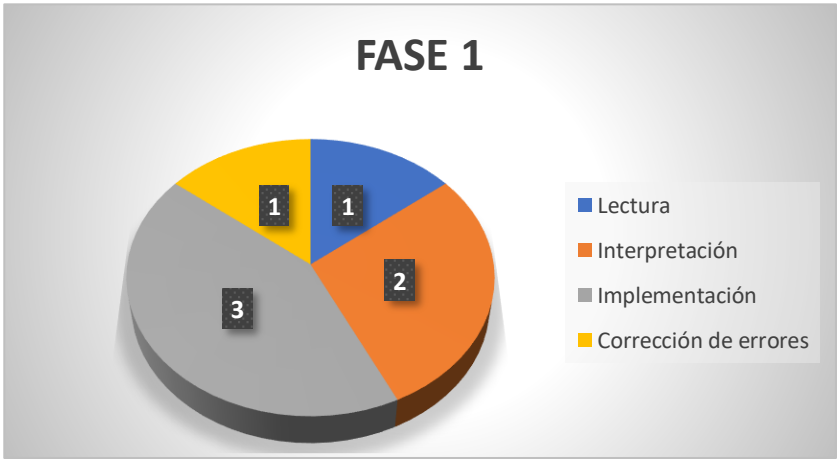
En el caso de leer los archivos JSON una vez se había obtenido la información se ha analizado con el debugger y con prints para mostrar el contenido mediante la consola para comprobar que la información sea la correcta. En el caso de la API, antes de implementarla en el código se izo una investigación sobre como funcionaba con el navegador Mozilla Firefox dado que te muestra el contenido JSON de una API REST en una formato amigable y entendible.

Otra herramienta que nos ha beneficiado en hacer los métodos de prueba ha sido el hecho de realizar el código con git dado que si una cosa la cual ya funcionaba deja de hacerlo teníamos la facilidad de poder volver atrás al punto que funcionaba sin ningún problema.

Dedicación en horas

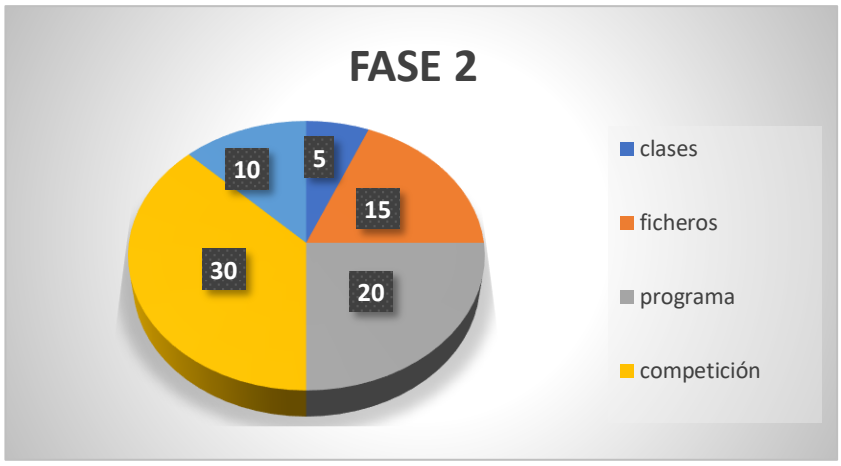
Fase 1

En la primera fase se dedicaron aproximadamente 7 horas en poder realizar todo el trabajo, pues estas fueron distribuidas en la interpretación de la práctica; el reconocimiento de las clases, atributos y métodos; la implementación de ellos en el Star UML y finalmente la comprobación de errores.



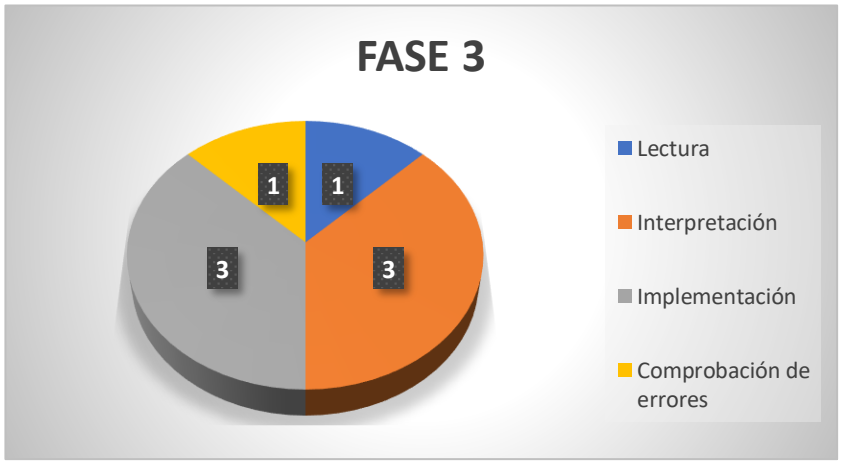
Fase 2

Al principio de esta fase se pronosticaron unas aproximadamente 80 horas de trabajo total con periodos de tiempo muy bien definidos, pero como era de esperar, las cosas no salieron como en un principio se esperaba, pues varios factores como la carga de trabajo de otras asignaturas o simplemente la complicación de ese punto de la fase, hicieron que fuera bastante difícil seguir el plan previsto.



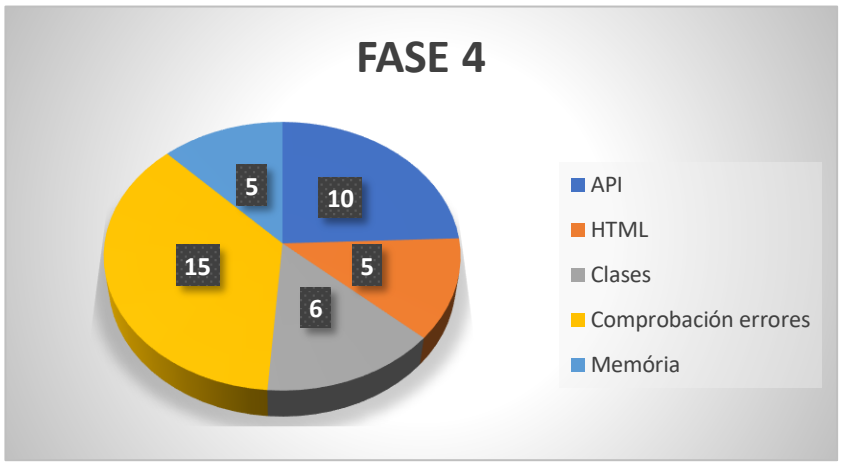
Fase 3

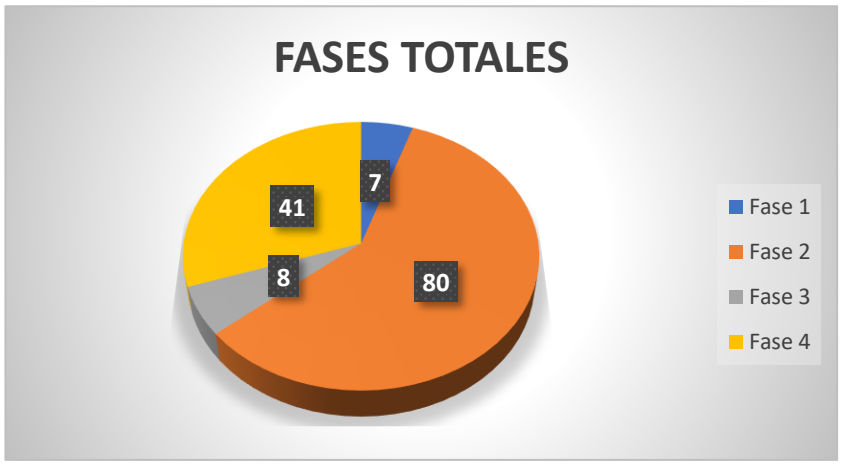
En comparación a la fase anterior, la carga de trabajo en esta fase fue infinitamente más pequeña pues, aunque tuvimos que aplicar los mismos pasos que en la fase primeriza, la dedicación en horas no supero las aproximadamente 8 horas en comparación con las 80 anteriores. Estas son debido a que la interpretación de las clases que debíamos aplicar al igual que su implementación fue algo más dificultoso que en la fase uno.



Fase 4

Finalmente, la cuarta fase, fue la culminación y finalización de la practica con la unión de todas las fases anteriores. En esta fase, el tiempo aproximadamente de dedicación fueron de unas 41 horas, pues aún con parte del código ya otorgado por el profesorado, la investigación de como acceder a la API, tratarla y generar el HTML, así como controlar los errores generados por ellos, hicieron incrementar ese tiempo de dedicación.





Conclusiones

Esta práctica a pesar de requerir un gran tiempo y esfuerzo, nos ha ayudado mucho a poder no solamente aplicar los conocimientos adquirido durante las sesiones efectuadas durante los últimos meses, sino que sin duda nos ha ayudado a entender los distintos funcionamientos del entorno de desarrollo integrado *IntelliJ* al igual que del *Star UML*.

El proceso de investigación sobre cómo aplicar diferentes funciones al igual de como efectuar varias partes del código, ha sido sin duda una de las partes más

costosas de la práctica pues la falta de conocimiento al igual que la complicación para solucionar problemas de los cuales la gran mayoría éramos inconscientes, han sido dos de los grandes causantes de esa dificultad. Esto se debe a que en muchas ocasiones tuvimos que cambiar por completo varias partes del código ya sea porque originalmente teníamos una idea en mente que resultó errónea, o porque a veces intentábamos aplicar algún código que no entendíamos y finalmente teníamos que cambiarlo.

Sin embargo, este proceso de investigación nos ha servido en gran parte para no solamente incrementar nuestras habilidades en diseño y procesado de información, sino que nos ha ayudado a entender cómo se efectuaba nuestro programa en todo momento.

La práctica además nos ha ayudado a entender todas las ventajas que este nuevo entorno de desarrollo nos proporciona a diferencia de lo que se usaba en años anteriores, como la posibilidad de debugar internamente, o de poner los nombrados *break_points.*

Finalmente, nos gustaría comentar que una de las partes más cruciales en el momento de efectuar una práctica es estimar correctamente el tiempo de dedicación necesario a cada parte e intentar mantenerse acorde el plan, pues cabe mencionar que varias veces ese plan no se siguió correctamente y por consecuencia nos vimos apurados de tiempo en un par de situaciones.

Bibliografía

Java: Try-Catch statement within a Do-While loop – StackOberflow [en línea] [fecha de consulta 27 de diciembre de 2020] Disponible en: https://stackoverflow.com/questions/22004921/java-try-catch-statement-within-a-do-while-loop

number formatting - How to set thousands separator in Java? - Stack Overflow [en línea] [fecha de consulta 27 de diciembre de 2020] Disponible en: https://stackoverflow.com/questions/5323502/how-to-set-thousands-separator-in-java

Java String replace(), replaceFirst() and replaceAll() method [en línea] [fecha de consulta 27 de diciembre de 2020 Disponible en: https://beginnersbook.com/2013/12/java-string-replace-replacefirst-replaceall-method-examples/

Difference between Abstract class and Interface – Javatpoint [en línea] [fecha de consulta 20 de diciembre 2020] Disponible en https://www.javatpoint.com/difference-between-abstract-class-and-interface
API REST con Java (JAX-RS) - Oscar Blancarte - Software Architecture [en línea] [fecha de consulta 18 de diciembre 2020] Disponible en https://www.oscarblancarteblog.com/api-rest-java-jax-rs/
Java Sort Arrays Examples (with Comparable and Comparator) [en línea] [fecha de consulta 13 de diciembre 2020] Disponible en https://www.codejava.net/java-core/collections/sorting-arrays-examples-with-comparable-and-comparator
java - warning - @author: is an unknown tag - Stack Overflow [en línea] [fecha de consulta 12 de diciembre 2020] Disponible en https://stackoverflow.com/questions/18717854/warning-author-is-an-unknown-tag
Javadocs - IntelliJ IDEA [en línea] [fecha de consulta 12 de diciembre 2020] Disponible en https://www.jetbrains.com/help/idea/working-with-code-documentation.html
Java - Write to File | Baeldung [en línea] [fecha de consulta 9 de diciembre 2020] Disponible en https://www.baeldung.com/java-write-to-file
How to write JSON object to File in Java? – Crunchify [en línea] [fecha de consulta 9 de diciembre 2020] Disponible en https://crunchify.com/how-to-write-json-object-to-file-in-java/
Java - Access is denied java.io.FileNotFoundException - Stack Overflow [en línea] [fecha de consulta 9 de diciembre 2020] Disponible en https://stackoverflow.com/questions/19561386/java-access-is-denied-java-io-filenotfoundexception
How to add an element to an Array in Java? – GeeksforGeeks [en línea] [fecha de consulta 6 de diciembre 2020] Disponible en https://www.geeksforgeeks.org/how-to-add-an-element-to-an-array-in-java/
Collections (Java Platform SE 8) [en línea] [fecha de consulta 6 de diciembre 2020] Disponible en https://docs.oracle.com/javase/8/docs/api/java/util/Collections.html
java - returning a private static variable - Stack Overflow [en línea] [fecha de consulta 4 de diciembre 2020] Disponible en https://stackoverflow.com/questions/25374168/returning-a-private-static-variable
Java ArrayList copy - Stack Overflow [en línea] [fecha de consulta 1 de diciembre 2020] Disponible en https://stackoverflow.com/questions/6536094/java-arraylist-copy
How to sort a list of things (string, float)? (Beginning Java forum at Coderanch) [en línea] [fecha de consulta 1 de diciembre 2020] Disponible en https://coderanch.com/t/488038/java/sort-list-string-float
Ways how to remove last character from String in Java | XENOVATION [en línea] [fecha de consulta 22 de noviembre 2020] Disponible en https://xenovation.com/blog/development/java/remove-last-ch

How to remove duplicate elements in ArrayList – HowToDoInJava [en línea] [fecha de consulta 21 de noviembre 2020] Disponible en https://howtodoinjava.com/java/collections/arraylist/remove-duplicate-elements/
Java.String.split() | Baeldung [en línea] [fecha de consulta 21 de noviembre 2020] Disponible en https://www.baeldung.com/string/split
Shuffle or Randomize a list in Java – GeeksforGeeks [en línea] [fecha de consulta 20 de noviembre 2020] Disponible en https://www.geeksforgeeks.org/shuffle-or-randomize-a-list-in-java/
How to remove an element from ArrayList in Java? – GeeksforGeeks [en línea] [fecha de consulta 20 de diciembre 2020] Disponible en https://www.geeksforgeeks.org/remove-element-arraylist-java/
How to Shuffle an Array in Java – JournalDev [en línea] [fecha de consulta 20 noviembre 2020] Disponible en https://www.journaldev.com/32661/shuffle-array-java
Gson - How to convert Java object to / from JSON - Mkyong.com [en línea] [fecha de consulta 16 de noviembre 2020] Disponible en https://mkyong.com/java/how-do-convert-java-object-to-from-json-format-gson-api/
java - How to format LocalDate to string? - Stack Overflow [en línea] [fecha de consulta 16 de noviembre 2020] Disponible en https://stackoverflow.com/questions/28177370/how-to-format-localdate-to-string
How to write a JSON string to file using the Gson library in Java? [en línea] [fecha de consulta 14 de noviembre 2020] Disponible en https://www.tutorialspoint.com/how-to-write-a-json-string-to-file-using-the-gson-library-in-java
java - How can I clear or empty a StringBuilder? - Stack Overflow [en línea] [fecha de consulta 14 de noviembre 2020] Disponible en https://stackoverflow.com/questions/5192512/how-can-i-clear-or-empty-a-stringbuilder