Análisis y Diseño de Software

Práctica 5



Tabla de contenido

[Apartado 1. Modelo de datos 3](#_Toc5551603)

[Apartado 2. Recomendador de vecinos 3](#_Toc5551604)

[Apartado 3. Otros recomendadores 3](#_Toc5551605)

[Apartado 4. Evaluar recomendaciones 4](#_Toc5551606)

[Apartado 5. Comprobar que todo funciona correctamente 4](#_Toc5551607)

[Diagrama de clases 5](#_Toc5551608)

[Nota 5](#_Toc5551609)

# Apartado 1. Genericidad y Colecciones

En este primer apartado, hemos realizado la implementación de las dos interfaces que nos daban. La primera decisión de diseño importante que hemos llevado a cabo es usar una lista enlazada para almacenar los elementos de la matriz. Evidentemente de esta manera evitamos que en matrices grandes las búsquedas sean muy largas, dado que no almacenamos los elementos vacíos. Además, en el elemento almacenamos su posición para saber dónde se encuentra. Hemos implementado MatrixElement de manera que sea genérico, y pueda almacenar cualquier objeto en su interior, en este caso serán celdas, en el siguiente apartado.

# Apartado 2. Comparadores

En el segundo apartado, hemos implementado un Comparator de IMatrixElement, que vamos a usar en un nuevo método (asListSortedBy), que se encarga de ordenar nuestra lista de la matriz, y retornar la misma. En nuestro método compare hemos ordenado primero por columnas, y si se encontraban en la misma por filas.

# Apartado 3. Simulación básica

En este ejercicio, hemos comenzado a usar Simuladores, usando en este, BasicSimulator, el más básico de los que vamos a usar y que tan solo muestra el número de elementos que hay en cada IMatrixElement, pero permanecerán impasibles dado que aún no tenemos ningún comportamiento que pueda modificarlos. También hemos creado una nueva clase BasicAgent, que implementa la interfaz que nos dan y una nueva clase, Cell; que básicamente contendrá una lista de agentes. Para hacer las copias, en un principio usamos clone, pero definitivamente decidimos sencillamente crear un nuevo BasicAgent.

# Apartado 4. Lambdas e Interfaces Funcionales

En el cuarto apartado hemos desarrollado los agentes, añadiéndoles comportamiento. Ahora tenemos una nueva interfaz (IAgent) la cual extiende de IBasicAgent de manera, que la clase que implementa la primera (Agent). Por comodidad también hemos creado la clase Comportamiento para almacenar la tupla de comportamiento y su condición necesaria para ocurrir. Si a un comportamiento no le damos condición, esta será siempre verdadera.

# Apartado 5. Patrones de diseño

# Diagrama de clases

El .png de la imagen se encuentra en el .zip si se quiere ver más grande y a mejor calidad.

# Nota

Documentación en ./doc, código fuente en ./src y ficheros utilizados en Moodle.

Compilar y ejecutar desde el directorio ./src