Vamos a realizar un programa que simula la vida en un hormiguero. En el hormiguero hay una serie de hormigas, de inicio, que van a buscar alimentos. Transcurrido un tiempo encuentran algo de alimento lo recogen y lo depositan en el hormiguero y vuelven a por otro. No obstante los alimentos que encontramos pueden no ser buenos. Cada alimento tiene dos propiedades: salubridad y poder, de tal forma que, cuando la hormiga se encuentra el alimento, debe compararlo con su historial de recogida de alimentos. Si, cada hormiga recuerda todos los alimentos que ha recogido durante su vida. El valor para comparar lo obtiene mediante el índice de salubridad/poder medio de toda su vida, si el alimento no supera este nivel medio la hormiga lo dejará y esperará otro alimento. Los alimentos son aleatorios, nunca sabe la hormiga cual va a encontrar.

Al final de su vida, la hormiga, va a entregar al hormiguero la idoneidad de su ADN. Esto abre un apasionante capitulo, por un lado el hormiguero va a criar una serie de hormigas (esto ocurre cuando todas las hormigas de la generación anterior han muerto) que dependerán de la despensa del hormiguero. Todas las hormigas llevarán el ADN de la hormiga más idónea de la generación anterior más el suyo propio.

Antes de criar hay que eliminar del hormiguero a las hormigas anteriores.

**Que se te da**

Como es un examen sobre hilos en java se te da mucho código ya hecho, que debes usar y para eso es obligatorio comprenderlo.

**App.java** es el punto de arranque. Observa que es el arranque de un hilo del Runnable Hormiguero.

**Hormiguero.java**

Es la clase que controla todo. Contiene una list de Hormigas, un map de despensa y una list de idoneidades y un criadero. Usa el constructor que se te da

Solo se te da un método **public void descargar(*Alimento* alimento)** que debe ser usado por cada hormiga para llenar la despensa con el alimento que encuentra.

Cuentas con algunos ObjectMother para facilitar tu tarea. **HormigasOM, AdnOM y AlimentoOM.**

El **Alimento.java** es una enumeración. A la misma le puedes preguntar por el índice de Salubridad/poder de cada alimento, así como el poder de cada uno. Solo hay cinco alimentos posibles

El **criadero.java** es otro hilo. Su misión es la de crear hormigas basándose en la despensa. También hay que vaciar la lista de hormigas muertas del hormiguero. Verás que para crear una hormiga hace falta una cantidad de poder (constante static en hormiga.java) el criadero cuenta la cantidad de poder que acumula la despensa y crea tantas hormigas como pueda. Retornando la nueva lista de hormigas al Hormiguero (suponemos que en el hormiguero se coge este retorno para incluir estas hormigas en la población del hormiguero). También necesita de las idoneidades para saber que ADN seleccionar. Otra misión es la de vaciar la despensa, es decir, ponerlo todos los alimentos a cero.

Cuentas con los métodos

**private IdoneidadADN obtenerMasIdoneo(List<IdoneidadADN> idoneidades)** para saber cuál es el mejor adn.

**private void vaciarDespensa()**

**private void eliminar()** que quita de la lista de hormigas a las muertas

y **private List<Hormiga> criar()** que te entrega una lista con las hormigas que ha criado

**“public Criadero(Hormiguero hormiguero)”** es el constructor que debes usar

**Hormiga.java**

Las hormigas son hilos. Su misión es la de deambular hasta que encuentran un alimento (pasa un tiempo aleatorio hasta que encuentra un nuevo alimento). Si el alimento merece la pena, se recoge y se deposita en la despensa del hormiguero. En cada deambular su edad crece y cuando llega a su edad límite, muere, entregando al hormiguero la idoneidad de su ADN. Cuando toda la generación de hormigas muere se usará el mejor ADN para criar una nueva generación (En Criadero.java)

La hormiga cuenta con **”private List<Alimento> recogidos”** que son los alimentos que ha recogido durante su vida. El método **“private float getIndiceSalubridadPoderMedio()”** te ayuda a conocer su historial.

Cuentas con el método **“private void acarrear(*Alimento* alimento)”** que lleva el alimento al hormiguero.

**public boolean isAlive()** nos dice si la hormiga esta viva

**“public Hormiga(int id,String adnProgenitor, Hormiguero hormiguero)”** es el constructor que se pasa.

**IdoneidadADN.java** es la clase que, a la muerte de la hormiga, recoge su nivel de trabajo mediante el indiceSalubridadPoder y su adn. Para que el criadero pueda basarse en el mejor adn de la generación actual de hormigas.

**Que se te pide**

Se te dan muchos métodos ya implementados, en java 6, y eso no nos gusta. Pásalos a java 8. Encontrarás la etiqueta “//java 8” para encontrarlos

En Criadero.java

private IdoneidadADN obtenerMasIdoneo(List<IdoneidadADN> idoneidades)

En Hormiga.java

private float getIndiceSalubridadPoderMedio()

Hay un momento en que la hormiga va tener que decidir si recoge o no el alimento que se acaba de encontrar. Usa Optional para ese retorno.

private *Alimento* recoger()

**Los hilos.**

Tienes que describir el tipo de hilos de Criadero, Hormiguero y Hormiga y hacer que todo funcione. El programa funciona si acaba porque la condición de salida del hilo del Hormiguero es llegar a una población mayor que **“private int poblacionMaxima= 10000;”**

En concreto **Hormiga.java** es un hilo donde la hormiga deambula, encuentra, recoge y deposita en la despensa, mientras está viva. Al final del hilo retorna la idoneidad de su adn.

En **Criadero.java** hemos decidido que sea un hilo que debe eliminar las hormigas muertas, criar las hormigas posibles y vaciar la despensa.

En **Hormiguero.java** debes arrancar la generación inicial que se crea en el constructor. Al final de esta primera generación ya contarás con una despensa llena para poder criar muchas generaciones de hormigas hasta alcanzar la población máxima permitida.

La puntuación

Si consigues que funcione el programa son siete puntos. Divididos de la siguiente forma:

3 puntos para la hormiga

3 puntos para el hormiguero

1 punto para el criadero

Las dos funciones a java 8 2 puntos

El uso de Optional 1 punto

Hay dos test que acompañan al examen. De tal forma que puedes entregar, por ejemplo, la clase Hormiga sin terminar el Criadero.

Hay un test para la hormiga y otro para el criadero. La única prueba posible para el Hormiguero es que alcance la población máxima.