FUNDAMENTOS DE PROGRAMACIÓN – PROGRAMACIÓN ORIENTADA A OBJETOS

Proyecto: Inspecciones de restaurantes de Nueva York (NYC)

Autora: Beatriz Pontes
Revisor: José A. Troyano
Última modificación: 12/04/2019

1 Material

Para la realización de esta práctica se dispone de los siguientes elementos contenidos en el fichero zip:

- **/doc/Enunciado.pdf**: fichero PDF con este enunciado
- **/data/**: carpeta de datos
  - **/data/inspecciones_restaurantes.csv**: fichero CSV con datos de registros de inspecciones de restaurantes de Nueva York (https://www.kaggle.com/new-york-city/nyc-inspections)
- **/src/fp.test**: paquete Java con las clases de test para las distintas clases que habrá que desarrollar en el proyecto
- **/src/fp.utiles**: paquete Java con utilidades de la asignatura

2 Datos disponibles

En este proyecto trabajaremos sobre datos de inspecciones de restaurantes en Nueva York. En estos datos encontramos solo un tipo de entidad:

- **Inspección:** contiene información relativa una inspección realizada en un restaurante. Hay que tener en cuenta que en cada restaurante se han realizado una o varias inspecciones.

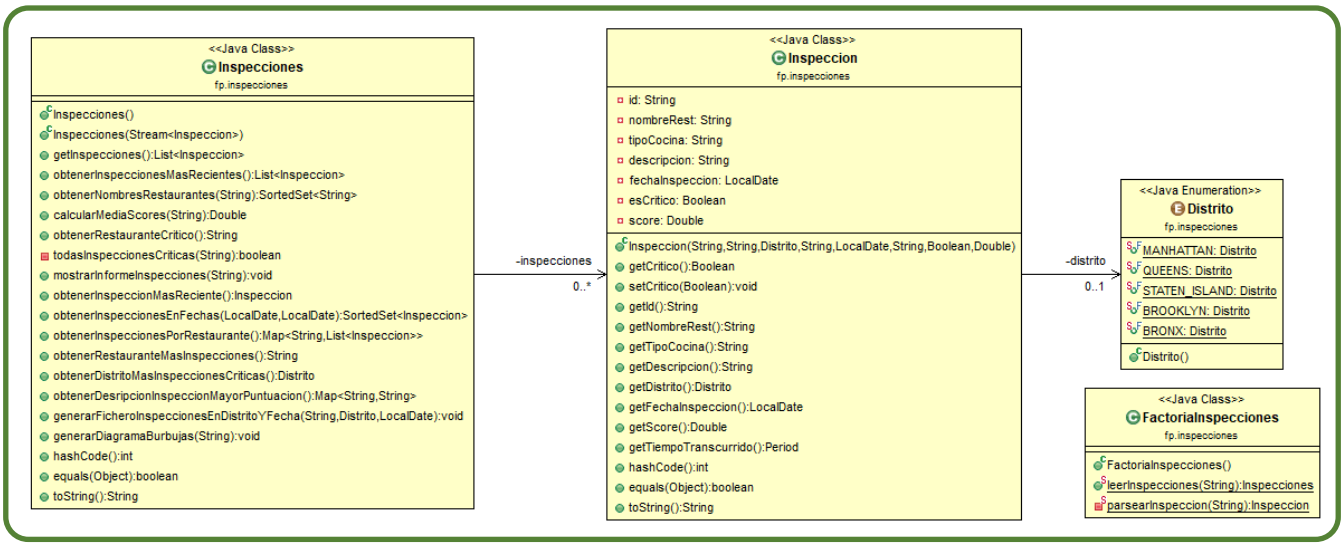
Los datos están disponibles en formato CSV. En la siguiente figura se muestran las primeras líneas del fichero de datos.

	CAMIS	DBA	BORO	CUISINE DESCRIPTION	INSPECTION DATE	VIOLATION DESCRIPTION	CRITICAL FLAG	SCORE
0	40511702	NOTARO RESTAURANT	MANHATTAN	Italian	06/15/2015	Hot food item not held at or above 140Â° F.	Critical	30.0
2	50046354	VITE BAR	QUEENS	Italian	10/03/2016	Non-food contact surface improperly constructe...	Not Critical	2.0
3	50061389	TACK'S CHINESE TAKE OUT	STATEN_ISLAND	Chinese	05/17/2017	Cold food item held above 41Â° F (smoked fish ...	Critical	46.0
4	41516263	NO QUARTER	BROOKLYN	American	03/30/2017	Live roaches present in facility's food and/or...	Critical	18.0
5	50015855	KABAB HOUSE NYC	QUEENS	Pakistani	03/03/2015	Non-food contact surface improperly constructe...	Not Critical	19.0
6	50058069	HENRI'S BACKYARD	BROOKLYN	American	06/22/2017	Evidence of rats or live rats present in facil...	Critical	39.0
7	40807238	RICHMOND COUNTY COUNTRY CLUB	STATEN_ISLAND	American	06/14/2017	Cold food item held above 41Â° F (smoked fish ...	Critical	12.0

3 Modelo

En el siguiente diagrama se muestran todos los elementos que habrá que implementar en este proyecto. Todos ellos se incluirán en el paquete **inspeccionesNYC**. Los aspectos más destacables del modelo son:

- **Inspeccion**: clase para implementar el tipo base.
- **Inspecciones**: tipo contenedor que incluye, además, algunos métodos de consulta basados en tratamientos secuenciales.
- **FactoriaInspecciones**: clase para dar soporte a la creación de objetos **Inspeccion** e **Inspecciones** a partir de datos en un fichero CSV.
- **Distrito**: tipo enumerado con los distintos distritos de Nueva York que consideraremos en el análisis.



Este diagrama ha sido generado con el plugin de Eclipse ObjectAid
URL de instalación: http://www.objectaid.com/update/current

4 Ejercicios

EJERCICIO 1

Crear el tipo enumerado **Distrito** con los siguientes valores posibles

- **MANHATTAN, QUEENS, STATEN_ISLAND, BROOKLYN, BRONX**

EJERCICIO 2
Crear la clase **Inspeccion** con los siguientes atributos

- **id:** atributo *String* que almacenará el identificador de la inspección
- **nombre:** atributo *String* que almacenará el nombre del restaurante
- **tipoCocina:** atributo *String* que almacenará el tipo de cocina
- **descripcion**: atributo *String* que almacenará la descripción del resultado de la inspección
- **fecha:** atributo *LocalDate* que almacena la fecha de la inspección.
- **distrito:** atributo del tipo enumerado **Distrito**.
- **esCritico:** atributo de tipo *Boolean* que almacena si la inspección ha sido o no crítica.
- **score:** atributo de tipo *Double* que almacena una puntuación numérica de la inspección.

EJERCICIO 3
Crear los siguientes métodos de la clase **Inspeccion** comprobando las restricciones de los atributos en los casos en los que sea necesario

- **Inspeccion:** constructor de la clase a partir de los atributos, en el orden en el que aparecen en las columnas del fichero
- Métodos **getters**: para todos los atributos de la clase
- **Inspeccion::getTiempoTranscurrido:** devuelve el periodo (tipo *Period*) transcurrido entre la fecha de la inspección y la fecha actual
- **Inspeccion::setCritico**.
- **Inspeccion::toString:** mostrando los atributos: *nombre, fecha, descripción, esCritico* y *score.*
- **Inspeccion::equals:** usando los atributos **fecha** e **id** para determinar la igualdad
- **Inspeccion::hashCode:** usando la misma selección de atributos que el método **equals**

EJERCICIO 4
Crear la clase **Inspecciones** con los siguientes atributos y métodos

- **Inspecciones:** atributo con una lista de objetos de tipo **Inspeccion**

- **Inspecciones:** constructor vacío
- **Inspecciones:** constructor a partir de un *Stream* de **Inspeccion**
- **Inspecciones:: getInspecciones:** método consultor que devuelve una copia del atributo
- **Inspecciones::toString:** mostrando todos los atributos
- **Inspecciones::equals:** usando el atributo para determinar la igualdad
- **Inspecciones::hashCode:** usando la misma selección de atributos que el método **equals**

EJERCICIO 5
Crear la clase **FactoriaInspecciones** con los siguientes métodos estáticos

- **FactoriaInspecciones::parsearInspeccion:** método privado para construir un objeto **Inspeccion** a partir de una línea CSV del fichero de entrada
- **FactoriaInspecciones::leerInspecciones:** método que devuelve un objeto **Inspecciones** a partir de la ruta del fichero en el que se encuentran los datos de las inspecciones

EJERCICIO 6

Completar la clase **Inspecciones** con los siguientes métodos

- **Inspecciones::obtenerInspeccionesMasRecientes:** filtra el conjunto inspecciones devolviendo una lista con aquellas de los últimos 2 años.
- **Inspecciones::obtenerNombresRestaurantes:** devuelve un conjunto ordenado con el nombre de aquellos restaurantes que sirvan un determinado tipo de comida (que se recibe como parámetro)
- **Inspecciones::calcularMediaScores:** devuelve el valor medio de las puntuaciones de las inspecciones de un determinado restaurante, cuyo nombre se recibe como parámetro.
- **Inspecciones::obtenerRestauranteCritico**: devuelve el nombre de algún restaurante que tenga todas sus inspecciones críticas (o **null** si no lo hubiera).
- **Inspecciones::mostrarInformeInspecciones:** dado el nombre de un restaurante, imprime por pantalla un informe con todas sus inspecciones.
- **Inspecciones::obtenerInspeccionMasReciente:** devuelve la inspección realizada más recientemente.
- **Inspecciones::obtenerInspeccionesEnFechas:** devuelve un conjunto ordenado (por distrito) con las inspecciones realizadas entre dos fechas recibidas como parámetros.
- **Inspecciones::obtenerInspeccionesPorRestaurante:** calcula un diccionario cuyas claves son los nombre de los restaurantes, y el valor es una lista con todas las inspecciones que se han llevado a cabo en él.
- **Inspecciones::obtenerRestauranteMasInspecciones:** devuelve el nombre del restaurante que más inspecciones haya tenido.
- **Inspecciones::obtenerDistritoMasInspeccionesCriticas:** devuelve el distrito con mayor número de inspecciones críticas.
- **Inspecciones::obtenerDesripcionInspeccionMayorPuntuacion:** devuelve un diccionario cuyas claves son los nombre de los restaurantes, y el valor es la descripción de la inspección que se haya llevado a cabo en dicho restaurante y haya tenido mayor puntuación.
- **Inspecciones::generarFicheroInspeccionesEnDistritoYFecha:** recibe el nombre de un fichero, un distrito y una fecha y genera un fichero con las inspecciones llevadas a cabo en dicho distrito y fecha.

EJERCICIO 7

Implementar el siguiente método de la clase **Inspecciones** que genera un informe gráfico usando la API de **Google Charts**

- **Inspecciones:: generarDiagramaBurbujas:** genera un diagrama de burbujas que muestre datos sobre las inspecciones que han resultado críticas según los distritos.
  El método recibirá un único parámetro:
  - ficheroSalida: un *String* con el nombre del fichero **HTML** en el que se generará la salida

  El proceso será el siguiente:
  - Construir el diccionario **inspeccionesPorDistrito** con las inspecciones por cada distrito.
  - Construir el diccionario **numeroInspeccionesPorDistrito** con el número de inspecciones por cada distrito.
  - Construir el diccionario **mediaScoresPorDistrito** con la media del atributo **score** para los restaurantes de cada distrito.
  - Construir el diccionario **porcentajesCriticasPorDistrito** con el porcentaje de inspecciones críticas para cada distrito.
  - Generar la salida **HTML** en un fichero de la carpeta **/out** con la siguiente instrucción:

```
GraphTools.bubbleChart("out/"+ficheroSalida,
   "Volumen de inspecciones",  numeroInspeccionesPorDistrito,
   "Porcentaje de inspecciones críticas",  porcentajesCriticasPorDistrito,
   "Media de scores", mediaScoresPorDistrito);
```