FUNDAMENTOS DE PROGRAMACIÓN. Curso 2018/19

EXAMEN DE LA SEGUNDA CONVOCATORIA. 3 de septiembre de 2019 (Java)

1. Material

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

- **/doc/enunciado.pdf**: fichero PDF con este enunciado.
- **/data/**: carpeta de datos.
  - **/data/titanic.csv**: fichero CSV con datos sobre el naufragio del Titanic.
- **/src/fp.accidentes.navales**: paquete Java donde debe incluirse la implementación del modelo.
- **/src/fp.accidentes.navales.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

El hundimiento del RMS Titanic es uno de los naufragios más infames de la historia. El 15 de abril de 1912, durante su viaje inaugural, el Titanic se hundió después de chocar con un iceberg, matando a 1502 de sus 2224 pasajeros y tripulantes. Esta tragedia sensacional conmocionó a la comunidad internacional y condujo a mejores regulaciones de seguridad para los buques. En este examen, nos proponemos desarrollar el software Java necesario para analizar datos de accidentes navales. Para ello, usaremos el formato provisto por Kaggle para el estudio del naufragio del Titanic (https://www.kaggle.com/c/titanic).

Id	Supervivien	Clase	Nombre	Sexo	Edad	NumHermanosOPa	NumeroPadre	PrecioTicke	Cabina	CiudadEmbarqu
1	0	3	Braund, Mr. (	hombre	22	1	0	7.25		Southampton
2	1	1	Cumings, Mrs	mujer	38	1	0	71.2833	C85	Cherbourg
3	1	3	Heikkinen, M	mujer	26	0	0	7.925		Southampton
4	1	1	Futrelle, Mrs.	mujer	35	1	0	53.1	C123	Southampton
5	0	3	Allen, Mr. Wi	hombre	35	0	0	8.05		Southampton
6	0	3	Moran, Mr. J	hombre	1001	0	0	8.4583		Queenstown
7	0	1	McCarthy, M	hombre	54	0	0	51.8625	E46	Southampton
8	0	3	Palsson, Mast	hombre	2	3	1	21.075		Southampton
9	1	3	Johnson, Mrs.	mujer	27	0	2	11.1333		Southampton
10	1	2	Nasser, Mrs.	mujer	14	1	0	30.0708		Cherbourg

Figura 1. Datos del naufragio del Titanic

3. Modelo

En el diagrama de la Figura 2 se muestran todos los elementos que habrá que implementar en este proyecto. Todos ellos se incluirán en el paquete **fp.accidentes.navales**. Los aspectos más destacables del modelo son:

- **PasajeroBarco**: clase para implementar el tipo básico.
- **Sexo**: tipo enumerado con los valores posibles en los que se clasifican los pasajeros.
- **AccidenteNaval**: tipo contenedor que incluye algunos métodos de consulta basados en tratamientos secuenciales.
- **FactoriaAccidentesNaval**: clase para dar soporte a la creación de objetos **PasajeroBarco** y **AccidenteNaval** a partir de datos en un fichero CSV.

4. Ejercicios

Para cada ejercicio se muestra su puntuación. No se puntuarán aquellos fragmentos de código que puedan ser generados de forma automática con Eclipse.

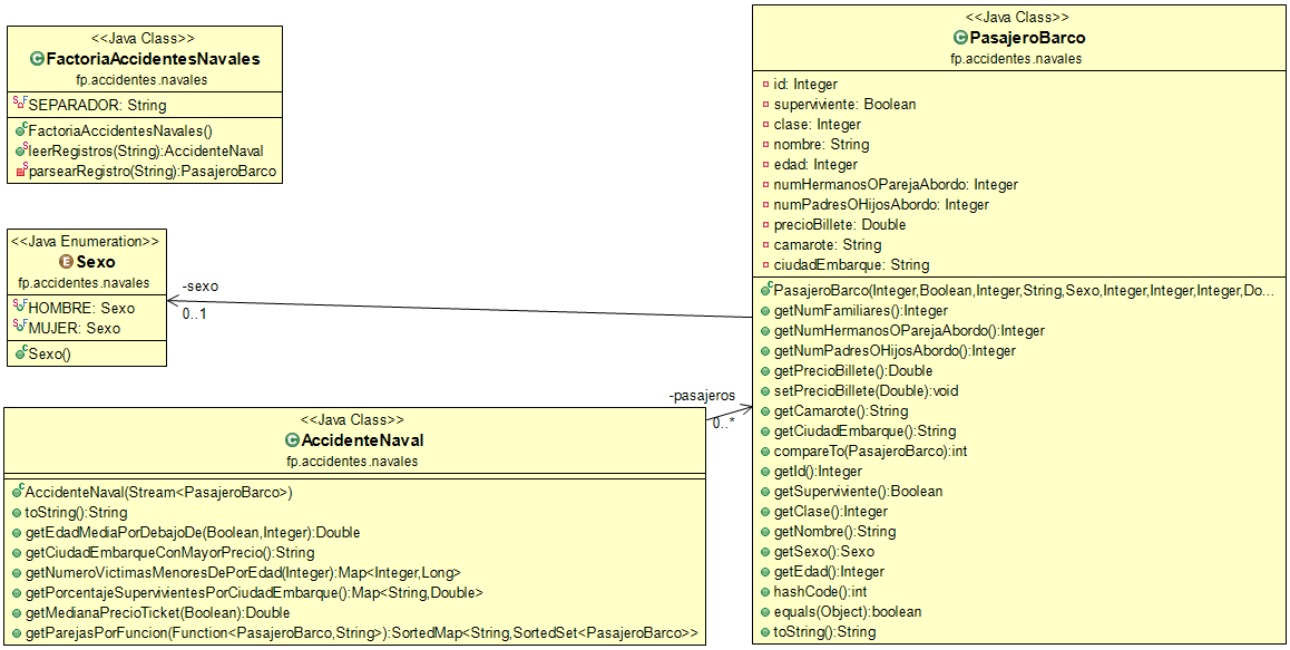

Figura 2. Diagrama de clases UML

EJERCICIO 1 – TIPO BASE (1,5 puntos)

Implemente la clase PasajeroBarco ajustándose a la siguiente descripción del tipo.

Tipo PasajeroBarco

Propiedades:

- id, de tipo Integer. Consultable. Identificador de un pasajero.
- superviviente, de tipo Boolean. Consultable. Indica si el pasajero sobrevivió al naufragio.
- clase, de tipo Integer. Consultable. Clase del alojamiento del pasajero en el barco.
- nombre, de tipo String. Consultable. Nombre del pasajero (o más información).
- sexo, de tipo Sexo. Consultable. Puede tomar los valores HOMBRE, MUJER.
- edad, de tipo Integer. Consultable. Edad del pasajero en el momento del accidente.
- numHermanosOParejaAbordo, de tipo Integer. Consultable. Representa el número de hermanos más marido/esposa incluidos en el pasaje.
- numPadresOHijosAbordo, de tipo Integer. Consultable. Representa el número de padres más hijos incluidos en el pasaje.
- precioTicket, de tipo Double. Consultable. Precio del ticket pagado por el pasajero.
- cabina, de tipo String. Consultable. Contiene la referencia al camarote/cabina que el pasajero ocupó en el viaje.
- ciudadEmbarque: de tipo String. Consultable. Nombre de la ciudad donde embarcó el pasajero.
- numFamiliares: de tipo Integer. Consultable. Es el número total de familiares (padres, hijos, hermanos más pareja) del pasajero incluidos en el pasaje.

Constructores: un solo constructor que tiene como parámetros las propiedades básicas del tipo, y en el mismo orden en el que se han definido en la descripción de las propiedades anteriores.

Criterio de igualdad: dos pasajeros son iguales si tienen el mismo id.

Criterio de ordenación: por id.

Representación como cadena: el valor de todas las propiedades, separados por coma.

Restricciones:

- Las propiedades id y superviviente no pueden ser nulas.
- Las propiedades numéricas no pueden ser nulas y deben tener un valor mayor o igual que cero.

EJERCICIO 2 – TIPO BASE (8 puntos, de los cuales 7,5 son las operaciones)

Implemente el tipo contenedor **AccidenteNaval,** con la siguiente descripción:

Propiedades:

- pasajeros, de tipo SortedSet<PasajeroBarco>. Consultable. Conjunto ordenado con los pasajeros del barco.

Constructores:

- Un constructor sin parámetros, que crea un objeto de tipo AccidenteNaval sin datos de pasajeros.
- Un constructor que tiene como parámetro un Stream<PasajeroBarco> y crea un objeto de tipo AccidenteNaval partir del Stream.

Criterio de igualdad: dos objetos de tipo AccidenteNaval son iguales si sus conjuntos de pasajeros son iguales.

Representación como cadena: cadena con todos los pasajeros separados por un salto de línea.

Otras operaciones:

a) **Double getEdadMediaPorDebajoDe(Boolean superviviente, Integer limite)**. Devuelve la edad media de los pasajeros cuyo valor de supervivencia coincida con el parámetro superviviente y su edad esté por debajo de una edad límite proporcionada por el parámetro limite. Si no se puede calcular (por ejemplo, si no hay datos procesables), se devuelve el valor 0. **(0,5 puntos)**

b) **String getCiudadEmbarqueConMayorPrecio().** Devuelve la ciudad de embarque con un ticket más caro (los nulos no deben tratarse). Si no se puede calcular (por ejemplo, si no hay datos procesables), se lanzará la excepción *NoSuchElementException* **(1 punto)**

c) **Map<Integer, Long> getNumeroVictimasMenoresDePorEdad(Integer edad).** Devuelve un Map que asocia a las edades de víctimas con edad menor o igual a un límite dado por el parámetro edad, el número de víctimas de dicha edad **(1 punto)**

d) **Map<String, Double> getPorcentajeSupervivientesPorCiudadEmbarque()**. Devuelve un Map con las ciudades de embarque como clave y el porcentaje de supervivientes que embarcaron en esa ciudad (en base al número de pasajeros de la ciudad en cuestión) como valor asociado. No trate los datos con valores nulos en la ciudad **(1,5 puntos)**

e) **Double getMedianaPrecioTicket(Boolean superviviente)**. Devuelve la mediana del precio de los pasajeros supervivientes o fallecidos, dependiendo del valor del parámetro superviviente. Tenga en cuenta que la mediana coincide con el percentil 50 de la distribución por lo que si num_items es el total de tickets vendidos, la mediana ocupará la posición "num_items/2" en una secuencia ordenada de precios. Si no se puede calcular (por ejemplo, si no hay datos procesables), se lanzará la excepción *NoSuchElementException* **(1,5 puntos)**

f) **SortedMap<String, SortedSet<PasajeroBarco>> getParejasPorFuncion(Function<PasajeroBarco, String> extractorIdPareja)**. Suponiendo que todas las parejas formadas por marido-esposa del barco tienen asociado un identificador, el método devuelve un SortedMap cuya clave es el identificador de una pareja y el valor es un conjunto ordenado de PasajeroBarco con los componentes de dicha pareja. Las claves del SortedMap estarán ordenadas por id, y los elementos del conjunto de los valores estará ordenado por Sexo. La función *extractorIdPareja* permite obtener, dado un pasajero, el identificador asociado a la pareja a la que pertenece. Tenga en cuenta que deben excluirse los solteros en la construcción del SortedMap y que, por lo tanto, los conjuntos ordenados que forman parte de los valores solo tendrán dos elementos el marido y la esposa **(2 puntos)**

EJERCICIO 3 – FACTORÍA (0,5 ptos)

Dado el tipo **FactoriaAccidenteNaval,** que tiene parcialmente implementado, añádale el siguiente método estático:

- **PasajeroBarco parsearRegistro(String linea)**. Crea un objeto de tipo PasajeroBarco a partir de una cadena de caracteres. La cadena de caracteres debe tener el mismo formato que una de las líneas del fichero CSV especificado en la Figura 1. Tenga en cuenta que un valor de supervivencia 0 equivale a false y un valor 1 a true.