FUNDAMENTOS DE PROGRAMACIÓN. Curso 2023/24
SEGUNDA CONVOCATORIA. 11 de julio de 2024. Bloque Java

Disponemos de dos ficheros .csv que contienen información acerca de distintos tipos de carreras populares que se han organizado durante el último año, así como los participantes de cada una de ellas. El primer fichero contiene la información de las carreras en el siguiente formato: código único de carrera, localidad, fecha y hora de comienzo, tipo de carrera, distancia y desnivel (ambos en metros):

```
tr_cobre_23,Gerena,2023-11-26 10:30,TRAIL,15000,750
tr_cazalla_24,Cazalla de la Sierra,2024-01-14 10:30,TRAIL,13500,600
cr_lmcp_23,Castilleja de la Cuesta,2023-09-23 21:45,CROSS,7800,200
…
```

Por otro lado, el fichero de participantes contiene la información de todos los participantes de todas las carreras, con el siguiente formato: identificador de la carrera en la que participa, nombre del participante, apellidos, edad, sexo y el tiempo empleado por dicho participante en finalizar la carrera (en el estándar ISO 8601):

```
tr_cobre_23,Antonio,Ruiz Ramos,50,HOMBRE,PT1H28M6.305983S
imd_sev_marlui_24,Pilar,Gil Álvarez,44,MUJER,PT47M36.180066S
cr_lmcp_23,Carlos,Álvarez Pérez,28,HOMBRE,PT45M27.537071S
…
```

Añada al proyecto proporcionado los paquetes `fp.carreras` y `fp.carreras.test`. Incluya en cada uno de ellos los tipos que se especifican en los siguientes ejercicios:

Ejercicio 1: Tipo Participante (0,5 puntos)
Implemente el tipo Participante mediante un **record**, a partir de la siguiente descripción:

Propiedades:

- **Id carrera**: identificador de la carrera en la que participa, de tipo String.
- **Nombre:** nombre del participante, de tipo String.
- **Apellidos:** apellidos del participante, de tipo String.
- **Edad:** edad del participante en el día de la carrera, de tipo Integer.
- **Sexo:** sexo del participante, de un tipo enumerado **Sexo**, que contiene los valores: HOMBRE, MUJER.
- **Duración:** tiempo empleado por el participante en la carrera correspondiente al identificador, de tipo Duration.
- **Categoría:** categoría del participante, de un tipo enumerado **Categoria**. La categoría se corresponderá con la edad y el sexo del participante, según los siguientes intervalos:
  - 17 a 20 años -> JUNIOR_F o JUNIOR_M
  - 21 a 23 años -> PROMESA_F o PROMESA_M
  - 24 a 39 años -> SENIOR_F o SENIOR_M
  - 40 años en adelante -> VETERANO_F o VETERANO_M

Constructores:

- C1: recibe un parámetro por cada propiedad básica del tipo, en el mismo orden en el que están definidos.

Criterio de igualdad: Dos participantes serán considerados iguales si ambos objetos tienen el mismo valor de las propiedades básicas.
Representación como cadena: una cadena con todas las propiedades básicas del tipo.
Restricciones:

- R1: La edad del participante debe ser igual o superior a 17 años.

Ejercicio 2. Tipo Carrera (1,5 puntos)
Implemente el tipo **Carrera** mediante una **clase**, a partir de la siguiente descripción:

Propiedades:

- **Id**: código identificador de la carrera, de tipo String. Consultable.
- **Localidad**: Nombre de la localidad donde se realiza la carrera, de tipo String. Consultable.
- **Fecha y hora**: fecha y hora de comienzo de la carrera, de tipo LocalDateTime. Consultable.
- **Modalidad**: tipo de la carrera, que puede ser: TRAIL, CROSS, ASFALTO, Consultable.
- **Distancia**: distancia a recorrer, en metros, de tipo Integer. Consultable.
- **Desnivel**: desnivel positivo en el recorrido de la carrera, en metros, de tipo Integer, Consultable.
- **Participantes:** lista de participantes en la carrera, de tipo List<Participante>, Consultable.

Constructores:
- C1: recibe un parámetro por cada propiedad básica del tipo, salvo la lista de participantes, en el mismo orden en el que están definidos. Permitirá crear una carrera sin participantes.

Restricciones:
- R1: La distancia mínima de la carrera debe ser de 7km.
- R2: El desnivel debe ser mayor o igual a 0 y menor o igual a 1km.

Criterio de igualdad: Dos carreras son iguales si tienen la misma localidad, fecha y hora, distancia e id.
Representación como cadena: Una cadena con todas las propiedades básicas del tipo, salvo la lista de participantes.
Orden natural: Las carreras se compararán por su fecha y hora de celebración. A igualdad de fecha y hora por localidad, y a igualdad de localidad por distancia, y a igualdad de distancia por id.
Otras operaciones:
- *void añadeParticipantes(List<Participante> participantes):* Dada una lista de participantes, los añade a la carrera, siempre que no existan. Es decir, los elementos de la lista de participantes que existan no deberán aparecer duplicados.

Ejercicio 3. Factoría (2 puntos)
Implemente la clase **FactoriaCarreras**, que permita leer ambos ficheros y crear un objeto de tipo **Carreras** (descrito en el ejercicio 4). Se recomienda incluir (al menos) los siguientes métodos:

- *Carreras leeCarreras(String nomfichCarreras, String nomfichParticipantes)*. Este es el método público y principal de la factoría. Recibe los nombres de los dos ficheros y devuelve un objeto de tipo Carreras con la información de todas las carreras y de sus participantes.
- *Carrera parseaCarrera(String lineaCarrera)*: devuelve un objeto de tipo Carrera a partir de una cadena de caracteres, correspondiente a una línea del fichero de carreras.
- *Map<String, List<Participante>> leeParticipantes(String nomfichParticipantes)*: recibe el nombre del fichero de participantes y realiza la lectura del mismo. Devuelve un diccionario donde los participantes se agrupan por cada carrera, en función de su identificador.
- *Participante parseaParticipante(String lineaParticipante)*: devuelve un objeto de tipo Participante a partir de una cadena de caracteres, correspondiente a una línea del fichero de participantes.

Ejercicio 4: El tipo Carreras (6 puntos)
Implemente el tipo **Carreras** a partir de la siguiente descripción:

Propiedades:
- **Carreras**: conjunto ordenado con las carreras realizadas, de tipo SortedSet<Carrera>. Este conjunto estará ordenado según la fecha de realización de la carrera. Use el orden natural para desempatar. Consultable.
- **Número de carreras**: número de carreras realizadas, de tipo Integer. Consultable.

Constructores:
- C1: recibe un parámetro de tipo Stream<Carrera>.

Criterio de igualdad: dos objetos serán iguales si lo son sus conjuntos de carreras.

Representación como cadena: una cadena con todas las propiedades básicas del tipo.

Otras operaciones:

- *List<Participante> participantesUltimaCarrera():* Devuelve los participantes de la última carrera realizada.

Añada además los siguientes métodos al tipo Carreras. Debe implementar todos los métodos usando **streams**, a menos que se indique lo contrario:

1. *carreraMayorDesnivelParticipante*: Recibe el nombre y apellidos de un participante y devuelve la carrera con mayor desnivel en la que haya participado. En caso de haber participado en varias carreras con el mismo desnivel, se devolverá la de menor distancia. Asuma que no habrá dos corredores con el mismo nombre y apellidos. Si no se puede calcular, se elverá la excepción NoSuchElementException. (1 punto)
   - Para resolver este ejercicio defina un método *buscaParticipante* en la clase **Carrera**, que reciba el nombre y apellidos del participante y lo devuelva, o null si no se encuentra.
2. *tiempoMedioCarrera*: Recibe el identificador de una carrera y devuelve el tiempo medio por kilómetro (minutos por kilómetro) de todos los participantes de dicha carrera. Si no se puede calcular, devuelve NoSuchElementException (1,25 puntos)
   - Para resolver este ejercicio defina un método *tiempoMedioPorKm* en la clase **Carrera**, que devuelva el tiempo medio por km de todos los participantes de la carrera.
3. *ganadoresPorCategoria*: Recibe el identificador de una carrera y devuelve un SortedMap donde las claves son las distintas categorías (según edad y sexo), y los valores los apellidos y el nombre del ganador de esa categoría. Si no se puede calcular, eleva NoSuchElementException (1,25 puntos)
4. *posicionesParticipante:* Recibe el nombre y apellidos de un participante y devuelve un Map con la clasificación de dicho participante para cada carrera en la que haya participado (las claves del Map serán los identificadores de cada carrera y los valores una lista de enteros que representa el puesto que obtuvo en la carrera el participante). Tenga en cuenta que la clasificación se calcula según la duración en la categoría general (sin diferenciar por edad), pero teniendo en cuenta el sexo del participante. **Este ejercicio debe resolverse utilizando bucles** (no con streams) (1,5 puntos)
   - Para resolver este ejercicio defina un método *posicionParticipante* en la clase Carrera, que reciba el nombre, apellidos y sexo de un participante y devuelva su posición en la clasificación general teniendo en cuenta su sexo.
5. *categoriaMasParticipantes*: Devuelve la categoría de las carreras en la que haya habido más participantes. (1 punto)

Escriba una clase **TestCarreras**. En la clase se leerán los datos del fichero y se probarán todos los tratamientos secuenciales, definiendo un método de test con los parámetros adecuados por cada tratamiento secuencial a probar. No se obtendrá la puntuación máxima del ejercicio si no se realiza el test, éste no ejecuta, o no se obtienen los resultados esperados. Los resultados esperados para los archivos csv proporcionados, con los valores indicados en los tests, son:

FACTORÍA:

```
Se ha leído información de 11 carreras
-----
En la última carrera han participado 85 participantes.
```

EJERCICIO 4.1:

```
----- Carrera con mayor desnivel de participante: --------

Elena Blanco Vázquez: Carrera [id=tr_cobre_23, localidad=Gerena, fechahora=2023-11-26T10:30, tipo=TRAIL, distancia=15000, desnivel=750]

Alejandro Ruiz Blanco: Carrera [id=tr_rutagu_24, localidad=Guillena, fechahora=2024-04-14T10:00, tipo=TRAIL, distancia=27500, desnivel=800]
```

EJERCICIO 4.2:

```
----- Tiempo medio de carrera: --------
```

medmar_sev_24: 5.3825451749327184

tr_cobre_23: 5.54375

EJERCICIO 4.3:

----- Ganadores por categoría: --------

tr_lima_pedr_24: {JUNIOR_F=Martínez Moreno, Martina, JUNIOR_M=Ramírez Molina, Juan, PROMESA_F=Romero Domínguez, Carmen, PROMESA_M=Torres Romero, Miguel, SENIOR_F=García Serrano, Martina, SENIOR_M=García Muñoz, Álvaro, VETERANO_F=Gutiérrez Fernández, Lucía, VETERANO_M=Pérez Fernández, Javier}

EJERCICIO 4.4:

----- Posiciones participante: --------

Sara Navarro Díaz:

{imd_sev_tam_24=16, cr_lmcp_23=16, tr_cobre_23=10, imd_sev_noct_23=13}

Luis Blanco Pérez:

{medmar_sev_24=43, cr_esqui_24=38, cr_lmcp_23=58, tr_cobre_23=33}

EJERCICIO 4.5:

----- Categoría con mayor número de participantes: --------
VETERANO_M