Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación
IIC2143 — Ingeniería de Software 2020-I

18 de marzo de 2020

Tarea Ruby

Indicaciones Generales

Esta tarea es de caracter **opcional**. Las personas que realicen esta tarea obtendrán **hasta 5 décimas extra** en su primera evaluación escrita. **Hacerla está muy recomendado**, ya que es una gran oportunidad para aprender Ruby (lenguaje sobre el que se construye el framework Ruby on Rails utilizado durante el curso). **El nombre del archivo principal debe ser main.rb**.

Introducción

El virus COVID-19 llegó al país y se hace más necesario que nunca ser capaces de detectar nuevos casos y prevenir posibles contagios. Por lo mismo, el Ministerio de Salud decide escribir un programa que sea capaz de manejar los datos existentes y sacar conclusiones de ellos basadas en ciertas directrices. De alguna manera, la entidad ministerial se enteró de que estás cursando Ingeniería de Software. Inmediatamente, deciden contactarte y asignarte la tarea de construir un programa escrito en Ruby con las funcionalidades requeridas por los analistas. **Es importante que el programa esté en lenguaje Ruby**, formato **.rb**, ya que luego será incorporado a la plataforma escrita en **Ruby On Rails** del ministerio. A continuación se describen las distintas funcionalidades a desarrollar y lo que se espera de cada una de ellas.

Modelación

Deberás programar las siguientes clases: Persona, Habitante, Pasajero, Sector, Vuelo, Pais. Tanto la clase Habitante como la clase Pasajero deben heredar de la clase Persona. La clase Habitante representa a toda la gente que **al momento del brote** se encuentra dentro del país. La clase Pasajero representa a toda la gente que **desde el momento del brote** llega desde otro país a algún aeropuerto chileno. La clase Sector representa a un sector de Chile arbitrariamente dividido. La clase Vuelo representa algún vuelo internacional. La clase Pais representa algún país distinto de Chile. Los atributos **mínimos** (pueden agregar más) de cada clase son los siguientes:

Clase Persona:

- **Atributos**: `id: int`, `nombre: str`, `edad: int`
- **SubClases**:
  - **Clase Habitante**:
    - **Atributos**: `sector: Sector`
    - **Métodos**: `cuarentena() -> bool`

- **Clase Pasajero**:
  - **Atributos**: `vuelos: List[Vuelo]`, `paises: List[Pais]`
  - **Métodos**: `cuarentena() -> bool`

Clase Sector:

- **Atributos**: `id: int`, `hay_infectados: bool`

Clase Vuelo:

- **Atributos**: `id: int`, `hay_infectados: bool`, `dias_desde_vuelo: int`

Clase Pais:

- **Atributos**: `id: int`, `nombre: str`, `hay_infectados: bool`

Aclaraciones importantes

- Un Habitante solo tiene un sector
- Un Pasajero puede tener varios vuelos
- Un Pasajero puede tener varios paises
- Un Pasajero no puede repetir vuelos ni paises
- Un vuelo puede tener `dias_de_vuelo >= 0` (o sea, que el vuelo puede haber ocurrido hace 0 o más días)
- Un Habitante con edad `>= 65` años debe hacer cuarentena
- Un Habitante de un sector donde hay infectados debe hacer cuarentena
- Un Pasajero con edad `>= 65` años debe hacer cuarentena
- Un Pasajero que estuvo en un país en que hay infectados debe hacer cuarentena.
- Un Pasajero que estuvo en un vuelo donde `dias_de_vuelo <= 14` debe hacer cuarentena.
- Un Pasajero que estuvo en un vuelo en que hay infectados debe hacer cuarentena.

Métodos

Para demostrar que implementaron de manera correcta cada método, **deberán crear un archivo de output** (más sobre este archivo en las secciones finales). En éste, deberán escribir datos según el respectivo método a modo de *logs*. Más detalles a continuación.

`pasajeros_en_cuarentena()`: Método que escribe en el archivo de *output* una sección con todos los pasajeros que deberían hacer cuarentena con la siguiente forma (`paisn` es el enésimo país que visitó el pasajero. `vuelos.length` es la cantidad de vuelos que hizo el pasajero):

```
*** COMIENZO PASAJEROS EN CUARENTENA ***
{nombe1}: {edad1}. {pais1.nombre} - {pais2.nombre} - ... - {paisn.nombre}. {vuelos.length}
{nombe2}: {edad2}. {pais1.nombre} - {pais2.nombre} - ... - {paisn.nombre}. {vuelos.length}
...
{nomben}: {edadn}. {pais1.nombre} - {pais2.nombre} - ... - {paisn.nombre}. {vuelos.length}
*** FIN PASAJEROS EN CUARENTENA ***
```

`habitantes_en_cuarentena()`: Método que escribe en el archivo de *output* una sección con todos los habitantes que deberían hacer cuarentena con la siguiente forma:

```
*** COMIENZO HABITANTES EN CUARENTENA ***
{nombe1}: {edad1}. Sector {sector1.id}
{nombe2}: {edad2}. Sector {sector2.id}
...
{nomben}: {edadn}. Sector {sectorn.id}
*** FIN HABITANTES EN CUARENTENA ***
```

`personas_en_cuarentena()`: Método que escribe en el archivo de *output* una sección con todas las personas que deberían hacer cuarentena con la siguiente forma (los habitantes deben ir antes que los pasajeros):

```
*** COMIENZO PERSONAS EN CUARENTENA ***
{nombe1}: {edad1}. {Habitante/Pasajero}
{nombe2}: {edad2}. {Habitante/Pasajero}
...
{nomben}: {edadn}. {Habitante/Pasajero}
*** FIN PERSONAS EN CUARENTENA ***
```

`personas_en_edad_de_riesgo()`: Método que escribe en el archivo de *output* una sección con todas las personas en edad de riesgo con la siguiente forma (los habitantes deben ir antes que los pasajeros):

```
*** COMIENZO PERSONAS EN EDAD DE RIESGO ***
{nombe1}: {edad1}. {Habitante/Pasajero}
{nombe2}: {edad2}. {Habitante/Pasajero}
...
{nomben}: {edadn}. {Habitante/Pasajero}
*** FIN PERSONAS EN EDAD DE RIESGO ***
```

`pasajeros_de_paises_infectados()`: Método que escribe en el archivo de *output* una sección con todos los pasajeros que estuvieron en países infectados con la siguiente forma (`pais_infn` es el enésimo país infectado que visitó el pasajero):

```
*** COMIENZO PASAJEROS DE PAISES INFECTADOS ***
{nombe1}: {edad1}. {pais_inf1.nombre} - {pais_inf2.nombre} - ... - {pais_infn.nombre}
{nombe2}: {edad2}. {pais_inf1.nombre} - {pais_inf2.nombre} - ... - {pais_infn.nombre}
...
{nomben}: {edadn}. {pais_inf1.nombre} - {pais_inf2.nombre} - ... - {pais_infn.nombre}
*** FIN PASAJEROS DE PAISES INFECTADOS ***
```

`pasajeros_de_vuelos_infectados()`: Método que escribe en el archivo de *output* una sección con todos los pasajeros que estuvieron en vuelos infectados con la siguiente forma (`vuelo_infn` es el enésimo vuelo infectado que hizo el pasajero):

```
*** COMIENZO PASAJEROS DE VUELOS INFECTADOS ***
{nombe1}: {edad1}. {vuelo_inf1.id} - {vuelo_inf2.id} - ... - {vuelo_infn.id}
{nombe2}: {edad2}. {vuelo_inf1.id} - {vuelo_inf2.id} - ... - {vuelo_infn.id}
...
{nomben}: {edadn}. {vuelo_inf1.id} - {vuelo_inf2.id} - ... - {vuelo_infn.id}
*** FIN PASAJEROS DE VUELOS INFECTADOS ***
```

Archivos CSV + TXT

Para cargar los datos, se les entregarán los siguientes archivos en formato csv: *habitantes.csv*, *pasajeros.csv*, *sectores.csv*, *vuelos.csv* y *paises.csv*. Adicionalmente, habrá un sexto archivo en formato txt que contendrá las instruciones que serán ejecutadas con el programa separadas por líneas. Este archivo se entregará en la línea de comandos como un `argv` (más sobre este archivo en las últimas secciones). Los archivos tienen la siguiente estructura:

habitantes.csv

id,nombre,edad,sector
0,Alexander Rovint,21,3
1,Elliot Alderson,34,6
2,Steven Spielberg,73,0

pasajeros.csv

id,nombre,edad,vuelos,paises
0,Daniel Leal,21,0:142:25:36:26:13,4:1:5
1,Darlene Alderson,30,0:140:7:41:69:420,4
2,Dwayne Johnson,47,1313:546:15,0:43

sectores.csv

id,hay_infectados
0,true
1,true
2,false
3,true
4,false
5,false
6,false

vuelos.csv

id,hay_infectados,dias_desde_vuelo
0,false,3
1,false,23
2,true,45
3,true,1
4,false,14
5,true,14

paises.csv

id,nombre,hay_infectados
0,wirtland,false
1,italia,true
2,korea,true
3,estonia,false

El archivo con las instrucciones tendrá la siguiente forma:

pasajeros_en_cuarentena
habitantes_en_cuarentena
personas_en_cuarentena
personas_en_edad_de_riesgo
pasajeros_de_paises_infectados
pasajeros_de_vuelos_infectados

En otras palabras, será una serie de líneas, cada una conteniendo una instrucción. **Pueden haber instrucciones repetidas. Pueden faltar instrucciones. El orden es totalmente arbitrario**. Por temas de espacio, **en los ejemplos entregados arriba puede que hayan referencias a ids de países/vuelos que no existen**, pero en la tarea eso **no** ocurrirá.

Ejecución

Para corregir la tarea, los ayudantes ejecutarán el siguiente comando:

```
ruby main.rb {datasets_folder} {instructions_file} {output_file}
```

Los archivos `.csv` con los datos siempre se llamarán según lo especificado en la sección **Archivos CSV + TXT**, pero irán dentro de la carpeta **{datasets_folder}**. El archivo de instrucciones tendrá el nombre especificado en **{instructions_file}** y el archivo al cual escribir el output tendrá el nombre especificado en **{output_file}**.

Para dejar esto más claro, si nuestro repositorio tuviera la siguiente forma:

```
repo/
--> data/
-----> habitantes.csv
-----> paises.csv
-----> pasajeros.csv
-----> sectores.csv
-----> vuelos.csv
--> instr.txt
--> main.rb
```

El comando a correr sería:

```
ruby main.rb data instr.txt output.txt
```

Esto debería generar un archivo `output.txt` con los outputs esperados, dejando el repositorio con la siguiente forma:

```
repo/
--> data/
-----> habitantes.csv
-----> paises.csv
-----> pasajeros.csv
-----> sectores.csv
-----> vuelos.csv
--> instr.txt
--> main.rb
--> output.txt
```

Foro

Cualquier duda que tengan sobre la tarea pueden preguntarla en las https://github.com/IIC2143-2020-1/proyecto/issues del repo del curso escribiendo en el título "[Tarea] - Pregunta...".

Entrega

La entrega de esta tarea se realizará mediante un `assignment` en Canvas, donde deberán subir un archivo `.zip` **sin** archivos `.csv` no `txt`, dentro del cual deben incluir el o los archivos que utilicen en su programa. Tienen hasta el miércoles 1 de Abril a las 23:59 para entregar su tarea.