Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación
IIC2143 — Ingeniería de Software 2021-2

Entrega: 8 de septiembre a las 23:59:59 horas

Tarea Ruby

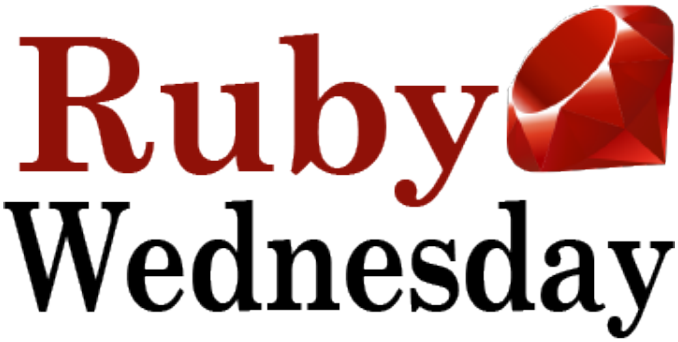
Entrega

- **Fecha y Hora: 8 de septiembre a las 23:59:59 horas**
- **Lugar:** *assignment* de Canvas.

Indicaciones Generales

Las personas que realicen esta tarea obtendrán hasta **5 décimas** extra en su primera evaluación escrita. Si bien la tarea es de carácter **opcional**, **hacerla está muy recomendado**, ya que es una excelente oportunidad para aprender Ruby, lenguaje sobre el cual se construye el framework Ruby on Rails utilizado durante todo el curso.

1. Introducción



Tu sueño siempre ha sido abrir un restaurante, así que, ¡Manos a la obra! Como eres muy organizado, decides primero modelar el sistema del restaurante, por lo que utilizarás Ruby para crear el restaurante *Ruby ~~Tuesday~~ Wednesday.*

Deberás modelar clases y crear métodos que te permitan simular algunas funcionalidades del famoso Ruby Wednesday.

2. Modelación

Deberás modelar las siguientes clases: `Persona, Cliente, Mesero, Platillo, Orden`. Si bien la modelación presentada a continuación es sugerida, pero no obligatoria, **sí se debe tener todas las clases escritas en esta sección.**

Clase Persona

Atributos

- `id`: int
- `nombre`: str

Subclases

- `Clase Cliente`
  - **Atributos**
    - `dinero`: int
    - `mesa`: int
- `Clase Mesero`
  - **Atributos**
    - `mesas`: List
    - `propina`: int

Clase Platillo

Atributos

- `id`: int
- `nombre`: str
- `precio`: int

Clase Orden

Atributos

- `id`: int
- `mesa`: int
- `platillos`: List

3. Aclaraciones y Consideraciones importantes

- En la clase **Cliente**, el atributo **mesa** debe ser igual al **id de la mesa en la cual se encuentra ubicado**.
- En la clase **Mesero**, el atributo **mesas** debe ser igual a **un array con los ids de las mesas que atiende**.
- En la clase **Orden**, el atributo **platillos** debe ser igual a **un array de arrays, donde cada array corresponde al id del platillo junto al id del cliente que lo solicitó**.
- No todos los meseros tienen la misma cantidad de mesas a su cargo, pero todos tienen, al menos, una mesa.
- Las mesas siempre tendrán, por aforo, 3 clientes.

4. Métodos

Para comprobar que las interacciones entre clases se realizaron de manera correcta, se deberá crear un archivo de *output*. En este se deberán escribir datos según el respectivo método a modo de logs. Más detalles a continuación según el método.

- `informacion_mesas()`: Este método se encarga de escribir en el archivo de *output* un listado de todas las mesas que se encuentran en el restaurante seguido del nombre del mesero que atiende esa mesa y el nombre de los clientes en la mesa, ordenados de menor a mayor según el id de la mesa. El *output* debe tener el siguiente formato:

```
COMIENZA INFORMACION MESAS
{id mesa 1} {nombre mesero mesa 1} {nombre cliente 1} {nombre cliente 2} {nombre cliente 3}
{id mesa 2} {nombre mesero mesa 2} {nombre cliente 4} {nombre cliente 5} {nombre cliente 6}
.
.
.
{id mesa n} {nombre mesero mesa n} {nombre cliente m-2} {nombre cliente m-1} {nombre cliente m}
TERMINA INFORMACION MESAS
```

- `generar_boletas()`: Este método se encarga de escribir en el archivo de *output* un listado de la boleta de cada mesa, separando los platos por cliente, ordenados de menor a mayor según el id de la mesa. El *output* debe tener el siguiente formato:

```
COMIENZA GENERAR BOLETAS
{id mesa 1} {nombre cliente 1} {total cliente 1} {nombre cliente 2} {total cliente 2} {nombre cliente 3} {total cliente 3} {total mesa 1}
{id mesa 2} {nombre cliente 4} {total cliente 4} {nombre cliente 5} {total cliente 5} {nombre cliente 6} {total cliente 6} {total mesa 2}
.
.
.
{id mesa n} {nombre cliente m-2} {total cliente m-2} {nombre cliente m-1} {total cliente m-1} {nombre cliente m} {total cliente m} {total mesa n}
```

```
TERMINA GENERAR BOLETAS
```

- `top_mesas()`: Este método se encarga de escribir en el archivo de *output* un listado de las 3 mesas que más gastaron, ordenadas de menor a mayor según el id de la mesa. El *output* debe tener el siguiente formato:

```
COMIENZA TOP MESAS
{id mesa 1} {total mesa 1}
{id mesa 2} {total mesa 2}
{id mesa 3} {total mesa 3}
TERMINA TOP MESAS
```

- `top_meseros()`: Este método se encarga de escribir en el archivo de *output* un listado de los 3 meseros que más propina tienen, ordenados de menor a mayor según el id del mesero. El *output* debe tener el siguiente formato:

```
COMIENZA TOP MESEROS
{id mesero 1} {propina mesero 1}
{id mesero 2} {propina mesero 2}
{id mesero 3} {propina mesero 3}
TERMINA TOP MESEROS
```

5. Calcular total cliente

Para calcular el total gastado por un cliente se debe:

1. Obtener el costo de los platillos pedidos por el cliente.
2. Luego de obtener esta suma total de platillos se le deberá añadir el 10 % de éste[1], correspondiente a la propina.

6. Calcular total mesa

El total gastado por la mesa m corresponderá a la suma de lo gastado por cada cliente que se encuentre ubicado en esa mesa.

7. Calcular propina

La propina del mesero n corresponde a la suma de la propina de cada mesa que atiende. Por ejemplo, si el mesero con id 1 atiende a las mesas 3, 6 y 7, con propinas $2.940, $4.100 y $1.000, respectivamente, la propina asociada al mesero con id 1 será de $8.040.

8. Archivos CSV + TXT

Para cargar los datos se subirán al repositorio oficial del curso los siguientes archivos en formato csv: `clientes.csv`, `meseros.csv`, `pedidos.csv`, `platos.csv`. Adicionalmente, se te entregará un archivo adicional en formato txt que contendrá las instrucciones que serán ejecutadas con el programa separadas

[1] Redondear por exceso, por ejemplo $156.5 sería $157.

por líneas. Este archivo se entregará en la línea de comandos como un **ARGV** (más sobre este archivo en la sección 9. Ejecución).

Los archivos tienen la siguiente estructura:

`clientes.csv`

Este archivo contiene la información de los clientes que asisten al restaurante Ruby Wednesday. Cada fila contiene el id de la persona, el nombre, el dinero que posee en su billetera y la mesa donde se sienta, separados por una coma (`","`). Cabe destacar que el id es único y no se repetirá por cliente. Un ejemplo del archivo `clientes.csv` es el siguiente:

```
id,nombre,dinero,mesa
1,Felipe,15000,2
2,Josefa,8000,1
3,Moises,12345,2
4,Catalina,9500,2
5,Nicole,4500,4
```

`meseros.csv`

Este archivo contiene la información de los meseros que trabajan en restaurante Ruby Wednesday. Cada fila contiene el id del mesero, el nombre, las mesas que atiende separadas por dos puntos (`":"`) y la propina inicial que tiene, separados por una coma (`","`). Cabe destacar que el id es único y no se repetirá por mesero. Un ejemplo del archivo `meseros.csv` es el siguiente:

```
id,nombre,mesas,propina
1,Javier,1:3,3000
2,Martin,2:5,12000
3,Pablo,4:6,0
```

`pedidos.csv`

Este archivo contiene la información de los pedidos realizados en el restaurante. Cada fila contiene el id del pedido, el id de la mesa que realizó el pedido, y los platillos solicitados separados por dos puntos (`":"`) junto con el id del cliente que lo solicitó separado por un guión (`"-"`), todo separado por una coma (`","`). Cabe destacar que el id es único y no se repetirá por pedido. Un ejemplo del archivo `pedidos.csv` es el siguiente:

```
id,mesa,platillos
1,2,1-3:2-3:1-4
2,1,3-2:1-2:5-2
3,2,2-1
```

Por ejemplo, para la orden con id 1, la mesa que la solicitó fue la mesa 2, solicitando el platillo con id 1 para el cliente con id 3, el platillo con id 2 para el cliente con id 3, y el platillo con id 1 para el cliente con id 4.

`platillos.csv`

Este archivo contiene la información de los platillos disponibles en el restaurante. Cada fila contiene el id del platillos, el nombre, y el precio del platillos separado por una coma (`","`). Cabe destacar que el id es

único y no se repetirá por plato. Un ejemplo del archivo `platillos.csv` es el siguiente:

```
id,nombre,precio
1,backaid,2000
2,accounting JS Rails,1500
3,Regex Concat,1900
4,Callisto,2500
5,Easycrumbs,1200
```

El archivo de instrucciones tendrá la siguiente forma:

```
informacion_mesas
generar_boletas
top_mesas
top_meseros
```

En otras palabras, el archivo de instrucciones será una serie de líneas, cada una conteniendo una instrucción. **Pueden haber instrucciones repetidas**. **Pueden faltar instrucciones**. **El orden es totalmente arbitrario**. Además, por temas de espacio, **en los ejemplos entregados en esta sección puede que hayan referencias a ids de mesas que no existen, pero en la tarea eso no ocurrirá**.

9. Ejecución

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

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

Los archivos .csv con los datos siempre se llamarán según lo especificado en la sección 8. Archivos CSV + TXT, pero irán dentro de la carpeta `{datasets_folder}`. El archivo instrucciones tendrá el nombre especificado en `{instructions_file}`, y el archivo al cual escribir el output tendrá el nombre especificado en `{output_file}`.

De manera visual, si nuestro repositorio tuviera la siguiente forma:

```
repo/
----|data/
--------|clientes.csv
--------|meseros.csv
--------|pedidos.csv
--------|platillos.csv
----|instr.txt
----|main.rb
```

El comando a correr sería:

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

Generando un archivo `output.txt` con los outputs esperados, dejando el repositorio de la siguiente forma:

```
repo/
----|data/
--------|clientes.csv
--------|meseros.csv
--------|pedidos.csv
--------|platillos.csv
----|instr.txt
----|main.rb
----|output.txt
```

IMPORTANTE: La corrección se realizará de manera automatizada, por lo que se espera que el formato que utilicen para el archivo de *output* sea exactamente el que se pide en el enunciado. Para facilitar su desarrollo, se les entregará un archivo de prueba que podrán utilizar para verificar que, efectivamente, están cumpliendo con esto.

10. Foro

Cualquier duda que surja respecto a la tarea pueden preguntarla en el foro oficial del curso, **escribiendo en el título de la issue "[Tarea] - Contenido pregunta"**.

11. Entrega

La entrega de la tarea se realiza mediante un *assignment* en Canvas, donde se deberá subir un archivo .zip **sin contener archivos .csv ni .txt**, dentro del cual deben incluir el o los archivos que utilicen en su programa. El plazo es hasta el 8 de septiembre a las 23:59:59 horas.