Tarea 3 - Programación Orientada a Objetos (POO)

INS125 - Lenguajes de Programación

Universidad Andrés Bello

17 de junio de 2021

1. Gestión de asistencia

Manejar la asistencia de una empresa puede llegar a ser bastante complejo, dado que hay que verificar si las personas fueron a trabajar, cuanto trabajaron, definir turnos, que planificación otorgar a tu personal y ver si están satisfechos o cómodos con sus asignaciones.

Para esto en necesario montar un sistema que permita realizar muchas de las tareas para organizar al personal, donde existirá una clase **Usuario**, la cual debe contener toda su información de una persona, como su nombre, rut, un listado de marcas (entrada y salida) y un listado de planificaciones asociadas. Cada usuario tendrá un rut único.

Día a día un usuario puede tener un Turno que cumplir. La clase **Turno**, tiene un identificador único, una hora de inicio y una hora de termino. A su vez existirá la clase **Turno Día** y **Turno Tarde**, ambas heredan de Turno. **Turno Día** tiene la particularidad de permitir minutos de atrasos. Mientras que la clase **Turno Tarde** permite minutos de retiro anticipado. De esta manera no se descuentan horas trabajadas en caso de que la persona llegue tarde o se retire temprano.

El usuario tiene la capacidad de realizar **Marcas**, cada marca esta asociada a un día y hora, y debe indicar si es de Entrada o Salida. La **Marca** debe estar representada como una clase.

Como se menciono antes, cada usuario tendrá una lista de la clase **Planificador**, cada Planificador contiene:

- Un turno
- Un día (indica a que día pertenece el turno)
- Un estado si la persona se encontraba ausente o presente para aquel día
- Horas trabajadas
- Minutos de atraso
- Minutos de retiro anticipado

Para asignar una planificación a un usuario, lo debe realizar una clase **Administrador** que hereda de usuario, la cual es capaz de asignar una planificación a cualquier usuario.

Por último, se le pide confeccionar una dentro de la clase Usuario un método llamado "Informe", el cual deberá calcular si el usuario cumplió con la planificación asignada generando un archivo `rut.txt` donde rut es el rut del usuario calculado.

Figura 1 presenta el diagrama de clases de la tarea, con sus correspondientes propiedades y métodos.

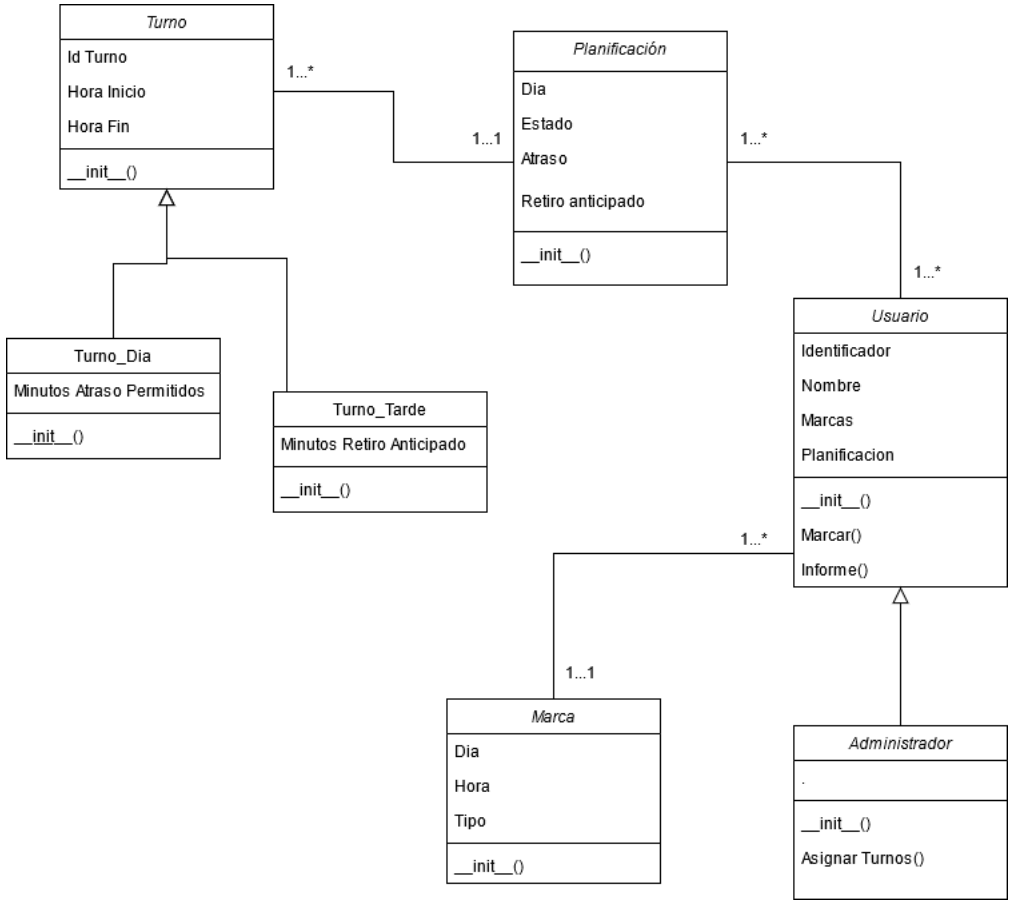

Figura 1: Diagrama UML

2. Tarea

En esta tarea, se le solicita implementar un programa que gestione la asistencia de un grupo de usuario dado una planificación con turnos para varios días. Se deberán crear todas las clases mencionadas anteriormente (es decir, Usuario, Administrador, Marca, Turno, Turno Día, Turno Tarde, Planificación) junto con sus correspondientes propiedades y métodos.

Algunas reglas del sistema son:

- Los turnos solo pueden durar un máximo de 6 hrs y deben ser entre 6 AM y 10 PM.
- La planificación de un usuario debe ser con días consecutivos. En caso de no tener un turno asignado dejar con id turno igual a cero.
- Un usuario solo puede realizar dos marcas por día.

Su programa debe generar como salida el archivo `rut.txt` y recibirá 4 archivos de entrada, donde cada archivo tendrá su propio formato, con datos separados por punto y coma (;) y serán detallados a continuación.

Archivo `Turnos.txt`

```
Id_turno;hora_inicio;hora_termino;minutos_permitidos;tipo_turno
```

Donde `Id_turno` corresponde al identificador del turno, el cual es un numero entero, `hora_inicio` es la hora a la que inicia el turno , `hora_termino` es la hora a la que finaliza el turno, `minutos_permitidos` son los minutos que permite de atraso o retiro anticipado (depende de que tipo sea) y `tipo_turno` indica el tipo de turno, "1" para Turno Día y "2" para Turno Noche.

Archivo `Usuarios.txt`

```
Identificador_usuario;nombre
```

Donde `Identificador_usuario` corresponde al identificador del usuario y `nombre` es todo el nombre del usuario

Archivo `Planificacion.txt`

```
Id_turno;día;estado;atraso;retiro_anticipado;Identificador_usuario
```

Donde `Id_turno` corresponde al identificador del turno para ese día, `día` indica al día que se esta aplicando la planificación, `estado` corresponde si la persona trabajo ese día o no (viene falso por defecto), `atraso` son los minutos de atraso, y `retiro_anticipado` los minutos anticipados que tiene la persona. Finalmente esta `Identificador_usuario` que es el identificador del usuario que se esta asignando la planificación.

Archivo `Marcas.txt`

```
identificador_usuario;día;hora;tipo
```

Donde `identificador_usuario` corresponde al identificador del usuario a quien pertenece la marca, `día` indica el día en que se efectuó la marca, `hora` la hora que se efectuó la marca y el `tipo` si fue entrada "1" o salida "2"

Salida Archivo `rut.txt`

Por cada usuario ingresado se debe generar una archivo de salida con la Gestión de Asistencia del usuario. Debe tener un total de lineas igual a la cantidad de planificaciones por usuario. En cada linea debe tener la fecha, la hora de inicio del turno, la hora finalización del turno, si la persona se encuentra presente o ausente y los minutos trabajadas para ese día. Todas separas por ";".

El calculo debe provenir de la planificación del usuario, por lo que el estado y los valores del usuario deben cambiar.

```
fecha;hora_inicio;hora_termino;estado;minutos_trabajadas
```

Para el calculo de los minutos trabajados solo debe considerar las horas del turno, no considerar los minutos si la persona entro antes o se retira después. En caso de que la persona tenga un atraso dentro del rango permitido por un Turno día, el calculo se considera desde inicio del turno. La misma lógica se aplica para el Turno Tarde.

Si el usuario se pasa del rango de atraso o retiro adelantado, esta bonificación no se considera para el calculo de los minutos trabajados.

2.1. Entrada y salida del programa

2.1.1. Ejemplos de entrada

```
1001;15:00;20:00;5;2
1002;9:00;15:00;10;1
1003;10:30;14:00;0;1
1004;10:15;15:00;5;1
1005;16:00;20:00;15;2
1006;16:00;22:30;10;2
1007;12:00;15:45;30;1
```

Figura 2: Ejemplo de archivo de `Turnos.txt`

```
1565514F;Vicente
1678721F;Alvaro
1592221D;Patricio
```

Figura 3: Ejemplo de archivo de `Usuarios.txt`

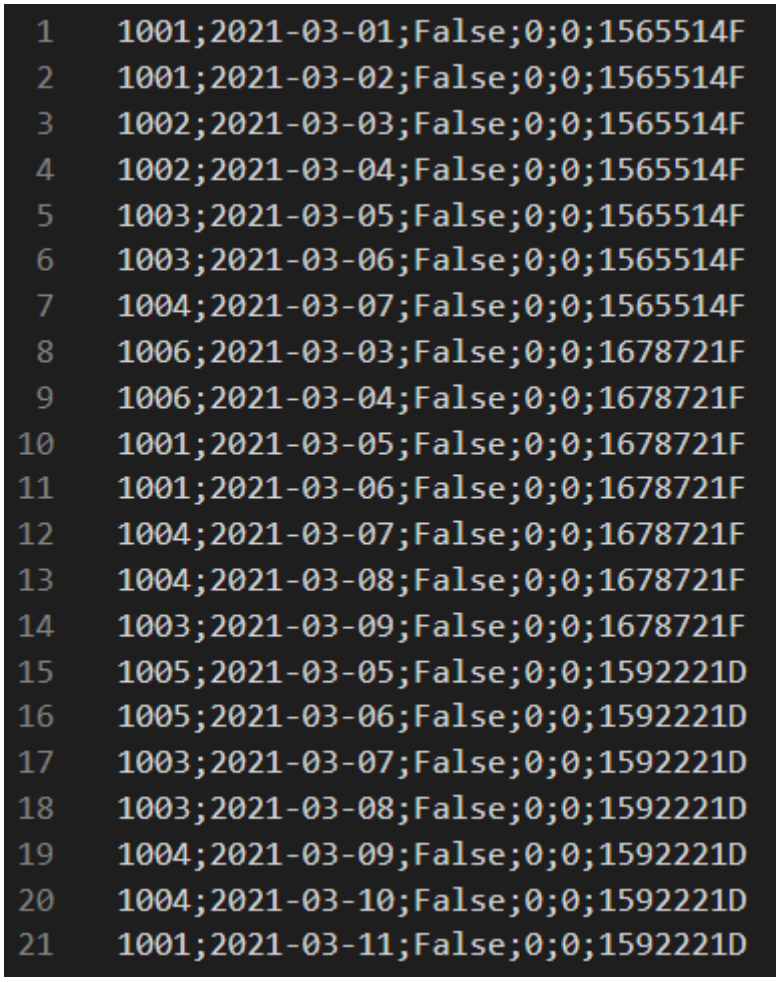

Figura 4: Ejemplo de archivo de `Planificacion.txt`

```
1565514F;2020-03-01;15:05;1
1565514F;2020-03-01;20:00;2
1565514F;2020-03-02;15:10;1
1565514F;2020-03-02;19:50;2
1565514F;2020-03-04;9:10;1
1565514F;2020-03-04;15:00;2
1565514F;2020-03-07;10:15;1
1565514F;2020-03-07;15:00;2
1678721F;2020-03-05;15:00;1
1678721F;2020-03-05;20:00;2
1678721F;2020-03-06;16:30;1
1678721F;2020-03-06;20:30;2
1678721F;2020-03-07;10:20;1
1678721F;2020-03-07;15:00;2
1592221D;2020-03-05;16:00;1
1592221D;2020-03-05;19:45;2
1592221D;2020-03-06;17:30;1
1592221D;2020-03-06;20:00;2
1592221D;2020-03-07;11:30;1
1592221D;2020-03-07;14:00;2
1592221D;2020-03-08;10:30;1
1592221D;2020-03-08;14:00;2
1592221D;2020-03-09;10:00;1
1592221D;2020-03-09;15:00;2
```

Figura 5: Ejemplo de archivo de `Marcas.txt`

2.1.2. Ejemplos de salida

```
2021-03-01;15:00;20:00;True;295
2021-03-02;15:00;20:00;True;280
2021-03-03;9:00;15:00;False;0
2021-03-04;9:00;15:00;True;360
2021-03-05;10:30;14:00;False;0
2021-03-06;10:30;14:00;False;0
2021-03-07;10:15;15:00;True;285
```

Figura 6: Ejemplo de archivo de `1565514F.txt`

3. Instrucciones

- Fecha de entrega: Jueves 24 de Junio a las 23:59 hrs.
- Método de entrega: su repositorio privado creado a través del link de la tarea. No se aceptarán entregas en repositorios no creados a través del link de la tarea.
- Trabajo personal hecho en lenguaje Python3.
- Su programa debe recibir y generar los archivos entrada y salida de acuerdo a los formatos dados.
- Sólo puede utilizar bibliotecas estándar (que no requieran instalación).
- El proceso de revisión será automatizado. Es importante respetar el formato establecido para los archivos de entrada y salida. Este formato no es modificable. Por lo tanto, si su archivo de salida no corresponde al formato preestablecido, su calificación será deficiente.
- Este enunciado podría sufrir modificaciones, las cuales serán anunciadas en el repositorio del laboratorio.
- Las preguntas sobre la tarea deben ser formuladas como un Issue en el repositorio del laboratorio ubicado en el siguiente link:
  https://github.com/INS125/Laboratorio/issues

4. Código de Honor

Toda persona inscrita en este curso se compromete a:

- Actuar con honestidad, rectitud y buena fe frente a sus profesores y compañeros.
- No presentar trabajos o citas de otras personas como propias o sin su correspondiente citación, ya sea de algún compañero, libro o extraídos de internet como también a no reutilizar trabajos presentados en semestres anteriores como trabajos originales.
- No copiar a compañeros ni hacer uso de ayudas o comunicaciones fuera de lo permitido durante las evaluaciones.
- Cualquier alumno o alumna que no respete el código de honor durante una evaluación (sea este la entrega de una tarea o el desarrollo de una prueba o control tanto durante la cátedra como el laboratorio) será evaluado con la nota mínima y será virtud de profesor, de acuerdo con la gravedad de la falta, las acciones siguientes a tomar.