RETO 2 – PROGRAMACIÓN BÁSICA

CONTEXTO

La casa desarrolladora UdeASoft desea desarrollar una aplicación para un turno virtual para la EPS SARU ya que esto presenta una gran ventaja reduciendo tiempos de desplazamiento y mejorando la atención al usuario.

Usted ha sido contratado como Desarrollador Experto en Java, porque ha logrado demostrar habilidades de desarrollo en este lenguaje de programación y se le ha concedido implementar la clase correspondiente al Turno Virtual.

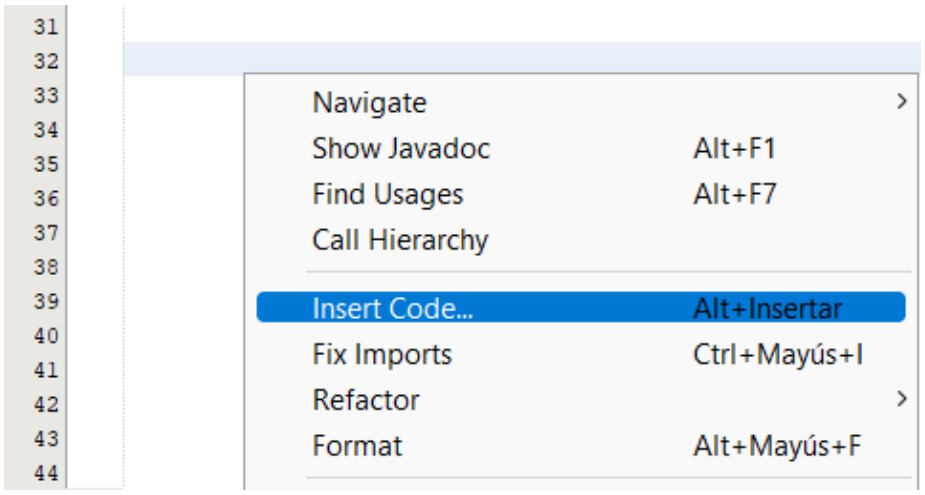
Esta clase contará con las siguientes funcionalidades:

- Atender siguiente turno: Se deja de atender al turno actual y se continúa con el que sigue si el turno virtual está abierto.
- Agregar turno perdido: Si el usuario que tenía el turno actual no responde, se cuenta dicho turno como perdido.
- Cambiar el estado del turno: Abrir o cerrar el turno; si el turno está cerrado, entonces lo abre; si está abierto, lo cierra.
- Se contará la cantidad de turnos que se llaman.

Para facilitar la implementación de la clase `TurnoVirtual`, el equipo de Ingeniería de software le hace entrega del diagrama de clase (Figura 1).

Recuerde que los métodos relacionados al constructor, getters y setters son obviados en el diagrama de clases, pero deberán ser incluidos en el código; estos métodos deben ser creados con el estándar camel case, por ejemplo, si el atributo se llama `turnoEnAtencion`, sus métodos correspondientes a get y set serían `getTurnoEnAtencion` y `setTurnoEnAtencion`, para el caso de los atributos de tipo `boolean`, el `get` se cambia por un `is`, por ejemplo, si el atributo se llama `estadoTurnoVirtual` y es de tipo `boolean`, su getter será `isEstadoTurnoVirtual`.

Nota: Recuerde que desde NetBeans puede generar automáticamente los getter y setters con la opción `Insert Code` dentro de la clase que se está creando.



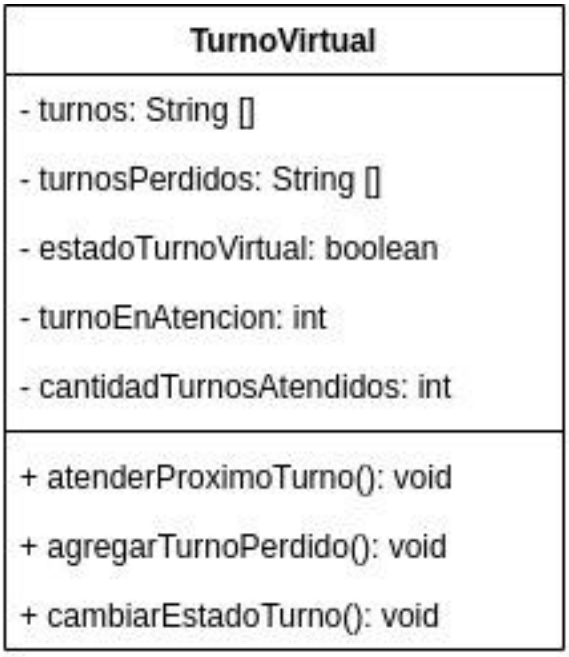


Figura 1

Además del diagrama, el equipo de Ingeniería entrega esta documentación para comprender mejor los elementos del diagrama:

Clase TurnoVirtual

Atributos

NOMBRE	TIPO DATO	CONCEPTO	INICIALIZACIÓN
`turnos`	`String[]`	Contiene los turnos que esperan ser atendidos.	En el método constructor.
`turnosPerdidos`	`String[]`	Contiene los turnos en los que los usuarios no respondieron al llamado.	En el método constructor se inicializa con un espacio en blanco por turno, el array debe tener longitud igual a `turnos`.
`estadoTurnoVirtual`	`boolean`	Guarda `true` si el `TurnoVirtual` está abierto, y `false` si no lo está.	Debe estar inicializado en `true`.
`turnoEnAtencion`	`int`	Guarda la posición del array `turnos` en el que se encuentra el turno en atención.	Debe estar inicializado en 0.
`cantidadTurnosAtendidos`	`int`	Cantidad de turnos que se han llamado.	Debe estar inicializada en 1.

Métodos (Ningún método recibe parámetros, y todos retornan `void`)

NOMBRE	CONCEPTO
`atenderProximoTurno`	Si `estadoTurnoVirtual` es `true`, se actualiza `turnoEnAtencion` por el turno que sigue en la lista y se incrementa la `cantidadTurnosAtendidos`.
`agregarTurnoPerdido`	Se agrega el `turnoEnAtencion` al array `turnosPerdidos`.

`cambiarEstadoTurno`	Si `estadoTurnoVirtual` es `true`, entonces la cambia a `false`; si `estadoTurnoVirtual` es `false` entonces la cambia a `true`

NOTA

El método **agregarTurnoPerdido YA** se le entrega **implementado** en la plantilla del VPL, no se debe preocupar por desarrollarlo.

TAREAS

- En el archivo preconstruido en la plataforma Moodle, implementar la clase especificada en el diagrama de clases, teniendo en cuenta la documentación dada por el equipo de Ingeniería de software.
- Los nombres de los métodos y atributos **DEBEN** ser nombrados tal y como aparecen en el diagrama de clases.
- Usted **NO** debe solicitar datos por teclado, ni programar un método `main`, tampoco debe especificar el paquete al que pertenece la clase, usted está solamente encargado de la construcción de la clase.

EJEMPLO

El calificador hará veces de usuario y será quien evalúe el funcionamiento del `Turno` que usted desarrolló:

1. El calificador añade doce (12) turnos en la cola:

```
String[] cola = {"A0", "A1", "A2", "A3", "A4", "A5",
                 "A6", "A7", "A8", "A9", "A10", "A11"};
TurnoVirtual turnoVirtual1 = new TurnoVirtual(cola);
```

Note que usted **NO** añade los turnos, lo hace quien use su clase (En este caso el calificador).

Por lo tanto, se espera que los atributos del objeto `turnoVirtual1` tengan los siguientes atributos:

```
Turnos: [A0, A1, A2, A3, A4, A5, A6, A7, A8, A9, A10, A11]
Turnos Perdidos: [ ,  ,  ,  ,  ,  ,  ,  ,  ,  ,  ,  ]
Estado del Turno Virtual: true
Turno en Atención: 0
Cantidad de turnos atendidos: 1
```

2. El calificador usa el método `atenderProximoTurno` cuatro (4) veces:

```
turnoVirtual1.atenderProximoTurno();
turnoVirtual1.atenderProximoTurno();
turnoVirtual1.atenderProximoTurno();
turnoVirtual1.atenderProximoTurno();
```

   Por lo tanto, se espera que los atributos del objeto `turnoVirtual1` se actualicen de la siguiente manera:

```
Turnos: [A0, A1, A2, A3, A4, A5, A6, A7, A8, A9, A10, A11]
Turnos Perdidos: [ ,  ,  ,  ,  ,  ,  ,  ,  ,  ,  ,  ]
Estado del Turno Virtual: true
Turno en Atención: 4
Cantidad de turnos atendidos: 5
```

   Como se esperaba, el turno en atención es el quinto en la cola, cuando el calificador terminó con el primer turno y atendió al siguiente, `turnoEnAtención` quedó en 1; en la segunda vez que atendió al siguiente, quedó en 2; en la tercera vez que atendió al siguiente, quedó en 3; y en la última vez que atiende al siguiente, queda en 4.

3. El calificador realiza las siguientes operaciones en ese orden:

```
turnoVirtual1.agregarTurnoPerdido();
turnoVirtual1.atenderProximoTurno();
turnoVirtual1.agregarTurnoPerdido();
turnoVirtual1.atenderProximoTurno();
turnoVirtual1.agregarTurnoPerdido();
```

   Por lo tanto, se espera que los atributos del objeto `turnoVirtual1` se actualicen de la siguiente manera:

```
Turnos: [A0, A1, A2, A3, A4, A5, A6, A7, A8, A9, A10, A11]
Turnos Perdidos: [A4, A5, A6,  ,  ,  ,  ,  ,  ,  ,  ]
Estado del Turno Virtual: true
Turno en Atención: 6
Cantidad de turnos atendidos: 7
```

   En el inciso anterior, el turno en atención era 4, refiriendose a que era el quinto en la cola, luego el calificador lo agrega a los turnos perdidos; el calificador atiende al siguiente turno (sexto en la cola) y también lo agrega a turnos perdidos; luego se atiende al siguiente turno (septimo en la cola) y lo agrega a turnos perdidos.

4. El calificador realiza las siguientes operaciones en ese orden:

```
turnoVirtual1.cambiarEstadoTurno();
turnoVirtual1.atenderProximoTurno();
turnoVirtual1.cambiarEstadoTurno();
```

En estas líneas de código, el calificador cambia el estado del turno, es decir, inicialmente está en `estadoTurno = true`, la primera línea lo cambia `false`, luego se intenta atender al siguiente turno, pero como el `estadoTurno` es `false`, dicha línea no tiene efecto, por último se cambia el estado del turno nuevamente quedando en `true`.

Se puede apreciar que el objeto queda igual al objeto mostrado en la captura del incizo anterior:

```
Turnos: [A0, A1, A2, A3, A4, A5, A6, A7, A8, A9, A10, A11]
Turnos Perdidos: [A4, A5, A6,  ,  ,  ,  ,  ,  ,  ,  ,  ]
Estado del Turno Virtual: true
Turno en Atención: 6
Cantidad de turnos atendidos: 7
```

Si desea realizar el ejemplo en Netbeans, copia y pega el siguiente código dentro de tu método `main`:

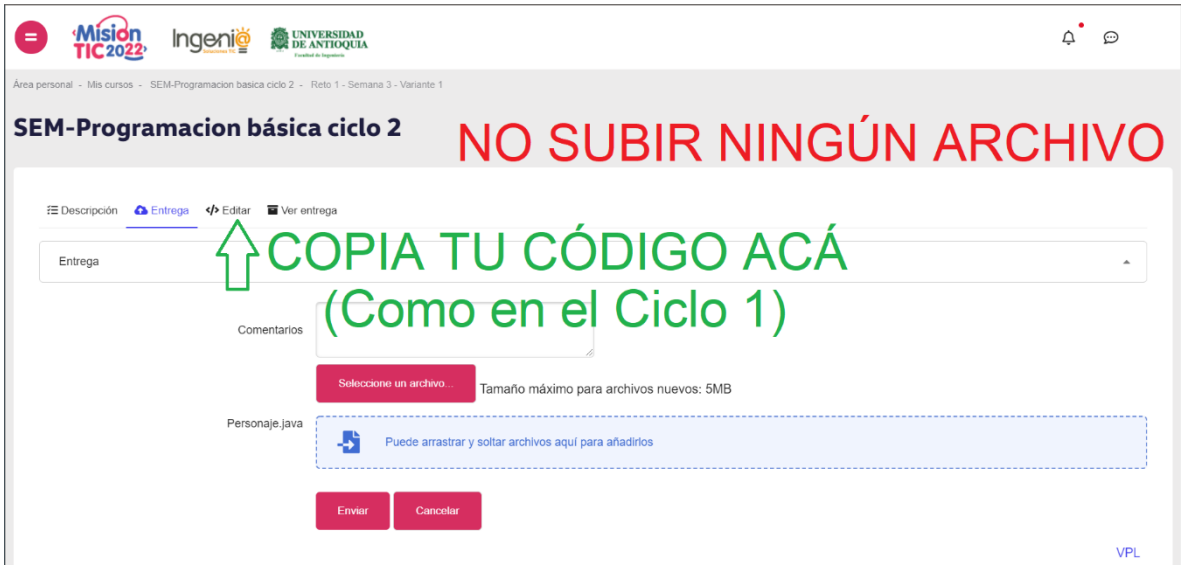
```
import java.util.Arrays;


public class R2vTurno {


    public static void main(String[] args) {
        // Pantallazo 1
          String[] cola = {"A0", "A1", "A2", "A3", "A4",
"A5",
                           "A6", "A7", "A8", "A9", "A10",
"A11"};
                    TurnoVirtual turnoVirtual1 = new
TurnoVirtual(cola);
        // Pantallazo 2
        turnoVirtual1.atenderProximoTurno();
        turnoVirtual1.atenderProximoTurno();
        turnoVirtual1.atenderProximoTurno();
        turnoVirtual1.atenderProximoTurno();
```

```
            // Pantallazo 3
            turnoVirtual1.agregarTurnoPerdido();
            turnoVirtual1.atenderProximoTurno();
            turnoVirtual1.agregarTurnoPerdido();
            turnoVirtual1.atenderProximoTurno();
            turnoVirtual1.agregarTurnoPerdido();
            // Pantallazo 4
            turnoVirtual1.cambiarEstadoTurno();
            turnoVirtual1.atenderProximoTurno();
            turnoVirtual1.cambiarEstadoTurno();
            System.out.print("Turnos: ");

System.out.println(Arrays.toString(turnoVirtual1.getTurn
os()));
            System.out.print("Turnos Perdidos: ");

System.out.println(Arrays.toString(turnoVirtual1.getTurn
osPerdidos()));
            System.out.print("Estado del Turno Virtual: ");

System.out.println(turnoVirtual1.isEstadoTurnoVirtual())
;
            System.out.print("Turno en Atención: ");

System.out.println(turnoVirtual1.getTurnoEnAtencion());
             System.out.print("Cantidad de turnos atendidos:
");

System.out.println(turnoVirtual1.getCantidadTurnosAtendi
dos());
        }


}
```

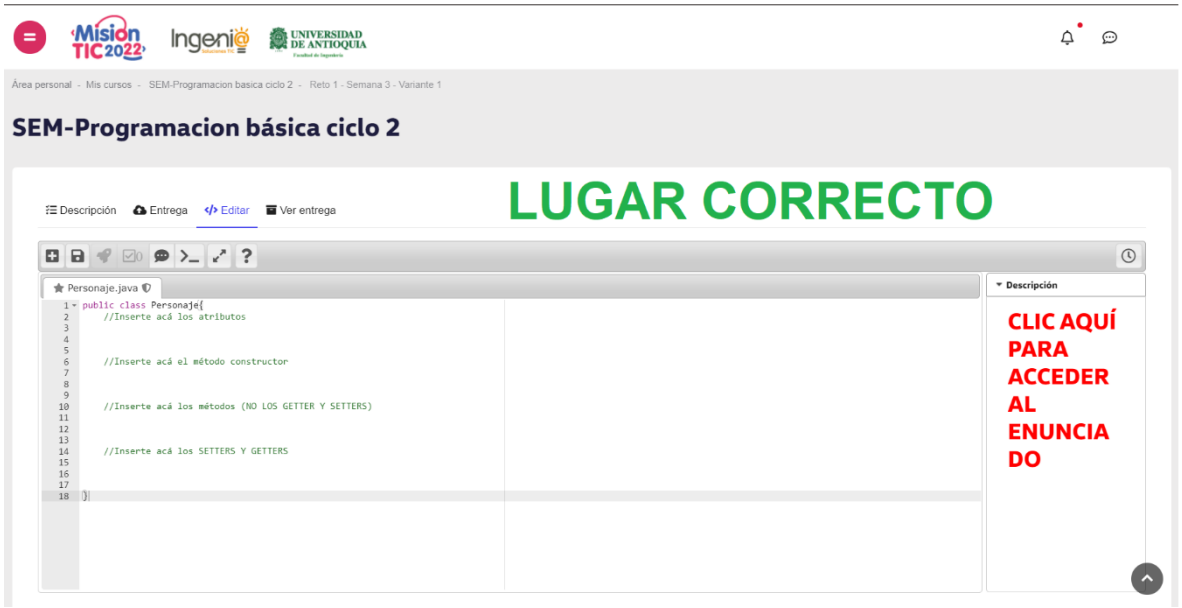
NOTA ACLARATORIA

Usted podrá desarrollar la clase requerida en un IDE como NetBeans, y al final copiar y pegar el código en la herramienta VPL, pero **NO** deberá subir archivos, es decir:

Modo incorrecto:



Modo correcto:



¡MUCHOS ÉXITOS EN EL DESARROLLO DEL RETO 2 TRIPULANTE!

ACLARACIÓN DE PLAGIO

El objetivo es que los tripulantes cuenten con una oportunidad de aprendizaje relacionada con la programación. La colaboración académica es buena mientras no se lleve a un engaño académico, ya que el engaño académico inflige las buenas conductas del saber y del aprendizaje. El engaño académico hace referencia al plagio o envío de ideas que no son propias.

Colaborar implica compartir ideas, explicar a alguien cómo podría hacer su trabajo (más no hacer el trabajo por el otro) y ayudar al otro si tienes problemas a la hora de ejecutar o encontrar errores en el código.

En aras de evitar el plagio se recomienda colaborar pero no compartir su código o proyecto, no compartir sus soluciones, no usar un código encontrado en internet u otras fuentes que las propias. (Mason, Gavrilovska, y Joyner, 2019)

Los ejercicios enviados a verificación deben cumplir con la política antiplagio. Es decir, cualquier envío que sea una copia textual de otro trabajo puede ser suspendido o no aprobado por parte del equipo evaluador. El acto de copiar material de otro estudiante es un comportamiento inaceptable para el desarrollo de las competencias individuales y su progreso en este curso.

Referencia.

Mason, T., Gavrilovska, A., y Joyner, D.A. (2019). *Collaboration vesus cheating. 50th ACM Technical Symposium on Computer Science Education SIGCSE 2019*, Mineapolis, MN. DOI: 10.1145/3287324.3287443