RETO 3 - PROGRAMACIÓN BÁSICA

CONTEXTO

El partido final de un evento deportivo se jugará en los próximos días y el consorcio de patrocinadores ha pedido al equipo encargado de planear el marketing del evento que, para mercadear la transmisión televisada de este, defina tres clases de tiquetes generadores de credenciales para acceder a la transmisión del juego: Uno llamado "Asistente" que hará las veces de acceso general, otro llamado "Preferencial" que ofrece un beneficio adicional y otro llamado "VIP" que ofrece otros beneficios diferentes.

En la venta de cualquiera de las tres clases de tiquetes se les asigna un identificador, el nombre completo del comprador, su dirección de residencia y unas credenciales que por defecto están desactivadas (en blanco). Si el comprador adquiere un tiquete "Preferencial" a este tiquete se le permite elegir si desea o no desea participar en el sorteo de un premio especial. Si el comprador adquiere un tiquete "VIP" a esta clase de tiquete se le da dos beneficios: El de generar otras credenciales para acceder a la transmisión de la previa del juego y el beneficio de compartir/dejar de compartir la transmisión del partido con otras tres pantallas.

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 las clases correspondientes a `Asistente` (clase padre), `VIP` y `Preferencial` (clases hijas)

Las clases `VIP` y `Preferencial` tienen un comportamiento muy similar, de hecho, lo extienden para lograr diferenciar los niveles de beneficios correspondientes.

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 `nombreCompleto`, sus métodos correspondientes a get y set serían `getNombreCompleto` y `setNombreCompleto`, para el caso de los atributos de tipo `boolean`, el get se cambia por un is, por ejemplo, si el atributo se llama `participaSorteo` y es de tipo `boolean`, su getter será `isParticipaSorteo`.

En la Figura 1 se muestra de forma gráfica el modelo que debe tener en cuenta para su implementación. Recuerde que desde NetBeans puede generar automáticamente

los getter y setters con la opción `Insert Code` dentro de la clase que se está creando, tal como se muestra en la Figura 2.

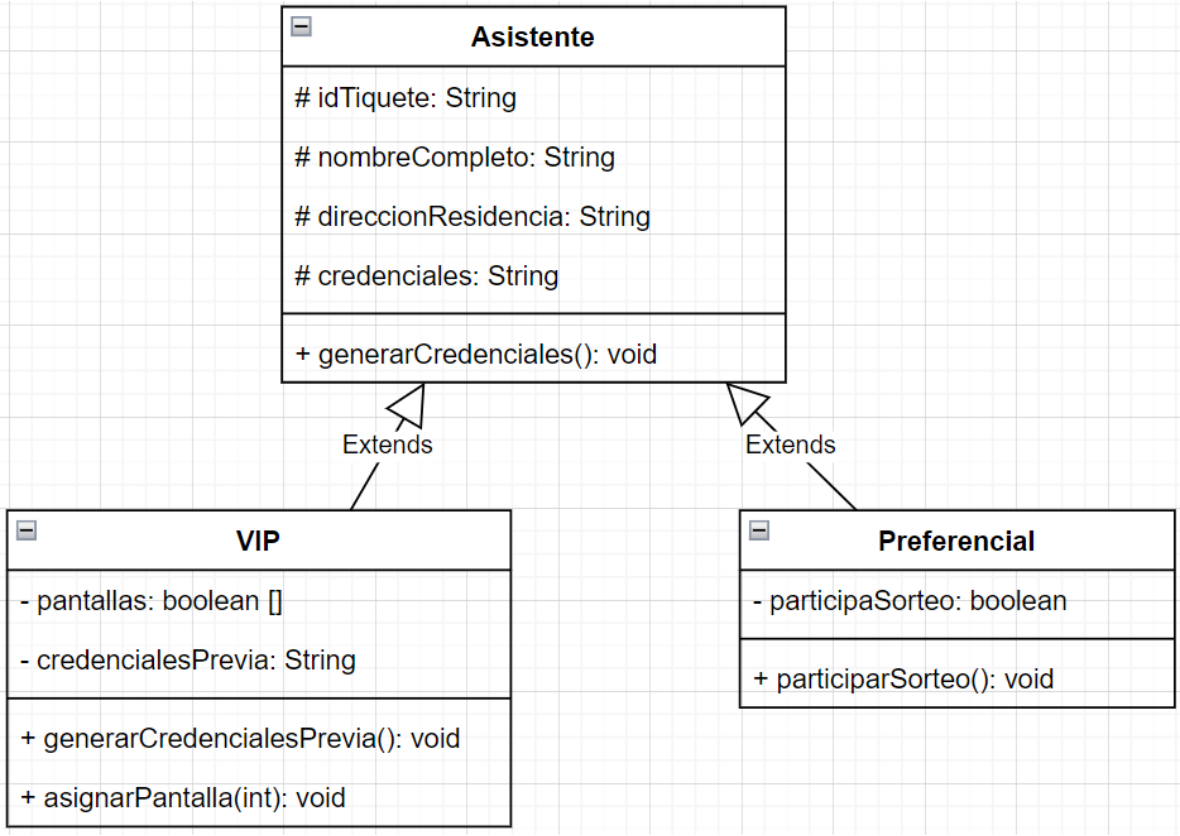


Figura 1

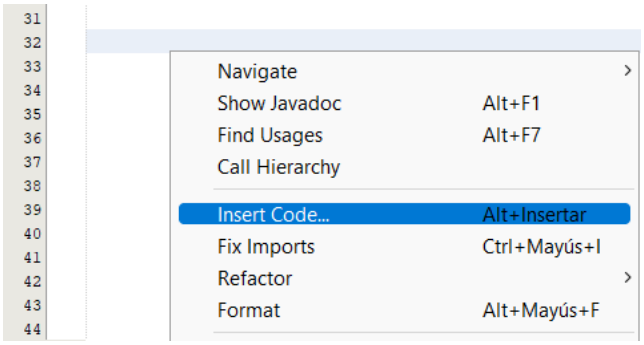


Figura 2

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

Clase Asistente

Atributos

NOMBRE	TIPO DATO	CONCEPTO	INICIALIZACIÓN
`idTiquete`	`String`	Es el número de identificación del tiquete.	En el método constructor.
`nombreCompleto`	`String`	Es el nombre del cliente que compró el tiquete.	En el método constructor.
`direccionResidencia`	`String`	Es la dirección de residencia del cliente.	En el método constructor.
`credenciales`	`String`	Es un número en formato `String` que servirá de credencial para ingresar a la transmisión	Debe ser igual a las dos comillas "" sin encerrar caracter alguno

Métodos

NOMBRE	TIPO RETORNO	PARÁMETROS	CONCEPTO
`generarCredenciales`	`void`	No recibe	Genera de manera aleatoria un número entero menor a 100000 (no puede generarse ninguno otro después de generar el primero) y lo convierte en un `String` que será la credencial de acceso a la transmisión, guardando el resultado en el atributo `credenciales`.

Clase Preferencial (Hereda de Asistente)

Atributo

NOMBRE	TIPO DATO	CONCEPTO	INICIALIZACIÓN
`participaSorteo`	`boolean`	Indica si el comprador de un tiquete preferencial está participando en el sorteo	Se inicializa con valor `false`.

Método

NOMBRE	TIPO RETORNO	PARÁMETROS	CONCEPTO
`participarSorteo`	`void`	No recibe	Se cambia el atributo `participaSorteo`. Aprovechando el getter y el setter de este. Si `participaSorteo` es igual a `true` lo cambia a `false`, y si `participaSorteo` es igual a `false` lo cambia a `true`.

Clase VIP (Hereda de Asistente)

Atributos

NOMBRE	TIPO DE DATO	CONCEPTO	INICIALIZACIÓN
`pantallas`	`boolean[]`	Es un array que contiene los estados de las tres pantallas en las que el comprador del tiquete VIP desea compartir la transmisión. Dicho array contiene booleanos en los que `true` representa una pantalla habilitada para acceder a la transmisión y `false` representa una pantalla inhabilitada	Debe estar inicializado en `new boolean[] {false, false, false}`
`credencialesPrevia`	`String`	Es un número en formato `String` que servirá de credencial para ingresar a la transmisión de la previa del juego final	Debe ser igual a las dos comillas `""` sin encerrar caracter alguno

Métodos

NOMBRE	TIPO RETORNO	PARÁMETROS	CONCEPTO
`generarCredencialesPrevia`	`void`	No recibe	Genera de manera aleatoria un número entero menor a 100000 (no puede generarse ninguno otro después de generar el primero) y lo convierte en un `String` que será la credencial de acceso a la transmisión de la previa, guardando el resultado en el atributo `credencialesPrevia`
`asignarPantalla`	`void`	`int pantalla`	El método recibe como parámetro un entero de 1 a 3 que representará alguna de las tres pantallas en el array `pantallas` con las que el comprador del tiquete puede compartir la transmisión. Si `pantallas[pantalla]` es igual a `true`, lo cambia a `false` y si `pantallas[pantalla]` es igual a `false` lo cambia a `true`. Si recibe un entero fuera del rango 1-3, mantiene la asignación intacta

TAREAS

- En el archivo preconstruido en la plataforma Moodle, implementar las clases especificadas 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 las clases.

EJEMPLO

El calificador hará veces de usuario y será quien evalúe el funcionamiento de los tiquetes que usted desarrolló:

1. Dos personas, Juan Pérez y María López están interesadas en adquirir tiquetes de acceso a la transmisión. El calificador genera y vende un tiquete Preferencial a Juan y un tiquete VIP a María:

```
Preferencial tiq_pref1 = new Preferencial(
        "0123",
        "Juan Perez",
        "CL 01 CR 02-03"
);

VIP tiq_vip1 = new VIP(
        "5434",
        "María López",
        "CR 29 CL 28-27"
);
```

El objeto `tiq_pref1` tendrá los siguientes atributos:

```
ID del tiquete:
0123
Nombre del cliente:
Juan Perez
Dirección del cliente:
CL 01 CR 02-03
Credenciales del cliente:

¿Participa en el sorteo?
false
```

El objeto `tiq_vip1` tendrá los siguientes atributos:

```
ID del tiquete:
5434
Nombre del cliente:
María López
Dirección del cliente:
CR 29 CL 28-27
Estado de las pantallas del cliente:
[false, false, false]
Credenciales del cliente:

Credenciales del cliente para la previa:
```

Observe que, a pesar de que en la instanciación del objeto `tiq_pref1` no se asignó el atributo `participaSorteo` sino los atributos `idTiquete`, `nombreCompleto` y `direccionResidencia`, las reglas de la implementación deben operar e inicializar el valor de este atributo como `false`, cosa que se hace evidente en el mensaje impreso del tiquete de Juan Pérez.

Observe que en la instanciación del objeto `tiq_vip1` no se asignó el atributo `pantallas` y a pesar de ello, las reglas de implementación definidas en la documentación deben operar y reasignar el valor de este atributo como `{false, false, false}` cosa que se hace evidente en el mensaje impreso del tiquete de María López al consultar este atributo `pantallas` y encontrar el array `{false, false, false}`. Así mismo la regla de instanciación del atributo `credencialesPrevia` para los tiquetes `VIP` indica que este atributo debe inicializarse como "" y se hace evidente al imprimir las "`Credenciales del cliente para la previa`".

Observe también que para los dos tiquetes generados (instanciación del objeto `tiq_pref1` y `tiq_vip1`) las reglas de implementación definidas en la documentación establecen que, en la instanciación de cualquier objeto de la clase `Asistente` (y por lo tanto de sus clases hijas) las credenciales deben inicializarse como un caracter vacío "", cosa que se imprime en los mensajes de los tiquetes de Juan Pérez y María López.

2. Juan Perez desea participar del sorteo, por lo tanto, debe activar su participación y el calificador lo hará por él ejecutando el método

`participarSorteo()`. El objeto `tiq_pref1` tendrá los siguientes atributos:

```
tiq_pref1.participarSorteo();
```

```
ID del tiquete:
0123
Nombre del cliente:
Juan Perez
Dirección del cliente:
CL 01 CR 02-03
Credenciales del cliente:

¿Participa en el sorteo?
true
```

3. Para acceder a la transmisión del partido, Juan debe contar con las credenciales que se deben generar con la compra del tiquete. Él pide que se le indiquen y la ejecución del método `generarCredenciales()` lo hará:

```
tiq_pref1.generarCredenciales();
```

Con ello, los atributos del tiquete preferencial de Juan y sus credenciales quedan:

```
ID del tiquete:
0123
Nombre del cliente:
Juan Perez
Dirección del cliente:
CL 01 CR 02-03
Credenciales del cliente:
73594
¿Participa en el sorteo?
true
```

4. María López quiere que, en la casa de sus padres, en la casa de una de sus hermanas y en la casa de un amigo, también se pueda ver la transmisión. Pide que habiliten sus credenciales (las de la transmisión de la previa y las de la transmisión del juego final) y pide que se habiliten las tres pantallas a las que tiene derecho con su tiquete clase VIP. Los atributos de su tiquete quedan así:

```
tiq_vip1.generarCredenciales();
tiq_vip1.generarCredencialesPrevia();
tiq_vip1.asignarPantalla(1);
tiq_vip1.asignarPantalla(2);
tiq_vip1.asignarPantalla(3);
tiq_vip1.asignarPantalla(3);
```

Obteniéndose la siguiente información de los atributos de `tiq_vip1`:

```
ID del tiquete:
5434
Nombre del cliente:
María López
Dirección del cliente:
CR 29 CL 28-27
Estado de las pantallas del cliente:
[true, true, true]
Credenciales del cliente:
71121
Credenciales del cliente para la previa:
72557
```

Observar que con la ejecución de los métodos `generarCredenciales()`, `generarCredencialesPrevia()` y `asignarPantalla(int)` se modifican los atributos del objeto `tiq_vip1` si se compara la pantalla anterior con la primera que se generó en la instanciación del objeto. María López ya cuenta con credenciales para ver la previa del juego, el juego como tal y para compartir estas transmisiones con las tres pantallas que ella eligió.

5. María López decide no compartir la transmisión del juego con su amigo y revoca el permiso que le tenía asignado en el array `pantallas`; esta pantalla fue la tercera que ella pidió habilitar en el paso anterior. Ejecutando el método `asignarpantalla(3)` se logra lo requerido y los atributos de su tiquete tiq_vip1 quedan así:

```
tiq_vip1.asignarPantalla(3);
```

```
ID del tiquete:
5434
Nombre del cliente:
María López
Dirección del cliente:
CR 29 CL 28-27
Estado de las pantallas del cliente:
[true, true, false]
Credenciales del cliente:
71121
Credenciales del cliente para la previa:
72557
```

6. Juan Pérez solicita cambiar su dirección de residencia y el calificador lo hará por él recurriendo al método setter `setDireccionResidencia`:

```
tiq_pref1.setDireccionResidencia("CL 04 CR 05-06");
```

Los atributos de su tiquete `tiq_pref1` se encontrarán así:

```
ID del tiquete:
0123
Nombre del cliente:
Juan Perez
Dirección del cliente:
CL 04 CR 05-06
Credenciales del cliente:
73594
¿Participa en el sorteo?
true
```

A continuación, se proporcionan las ejecuciones hechas durante estos ejemplos. Puede usar estas líneas a modo de referencia para corroborar de manera local el funcionamiento de su propuesta de solución. La ejecución conjunta de todas estas líneas debería dar como resultado la información final de atributos que se mostró para el tiquete de Juan Pérez y para el tiquete de María López:

```
Preferencial tiq_pref1 = new Preferencial(
                "0123",
                "Juan Perez",
                "CL 01 CR 02-03"
        );
```

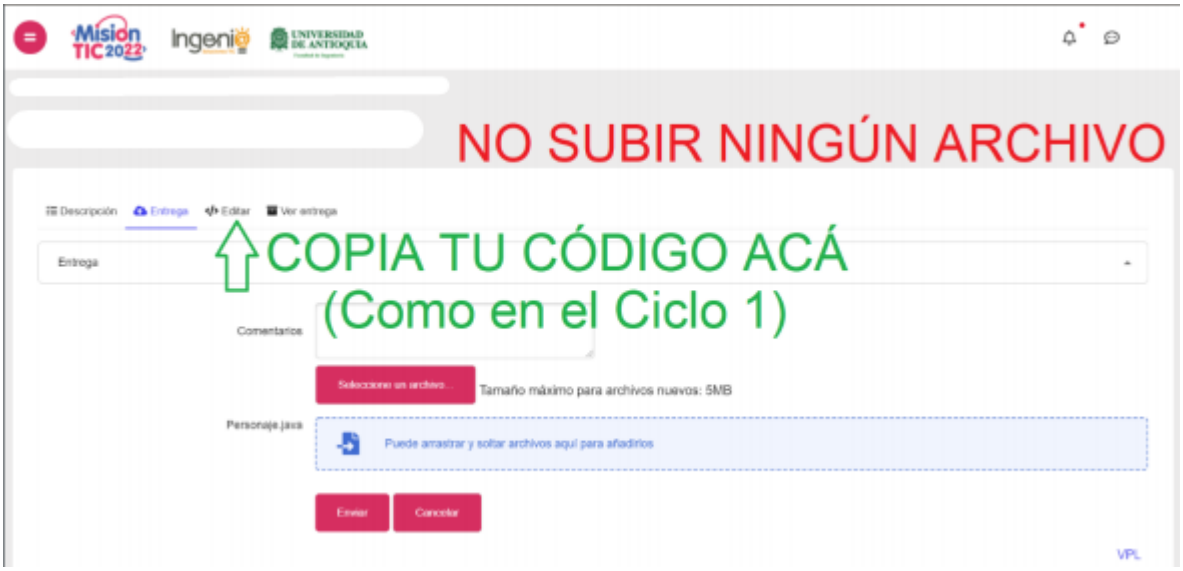
```
VIP tiq_vip1 = new VIP(
                "5434",
                "María López",
                "CR 29 CL 28-27"
        );
tiq_pref1.participarSorteo();
tiq_pref1.generarCredenciales();
tiq_pref1.setDireccionResidencia("CL 04 CR 05-06");
tiq_vip1.generarCredenciales();
tiq_vip1.generarCredencialesPrevia();
tiq_vip1.asignarPantalla(1);
tiq_vip1.asignarPantalla(2);
tiq_vip1.asignarPantalla(3);
tiq_vip1.asignarPantalla(3);

System.out.println("ID del tiquete:");
System.out.println(tiq_pref1.getIdTiquete());
System.out.println("Nombre del cliente:");
System.out.println(tiq_pref1.getNombreCompleto());
System.out.println("Dirección del cliente:");
System.out.println(tiq_pref1.getDireccionResidencia());
System.out.println("Credenciales del cliente:");
System.out.println(tiq_pref1.getCredenciales());
System.out.println("¿Participa en el sorteo?");
System.out.println(tiq_pref1.isParticipaSorteo());

System.out.println("ID del tiquete:");
System.out.println(tiq_vip1.getIdTiquete());
System.out.println("Nombre del cliente:");
System.out.println(tiq_vip1.getNombreCompleto());
System.out.println("Dirección del cliente:");
System.out.println(tiq_vip1.getDireccionResidencia());
```

```
System.out.println("Estado de las pantallas del cliente:");
System.out.println(Arrays.toString(tiq_vip1.getPantallas()));
System.out.println("Credenciales del cliente:");
System.out.println(tiq_vip1.getCredenciales());
System.out.println("Credenciales del cliente para la previa:");
System.out.println(tiq_vip1.getCredencialesPrevia());
```

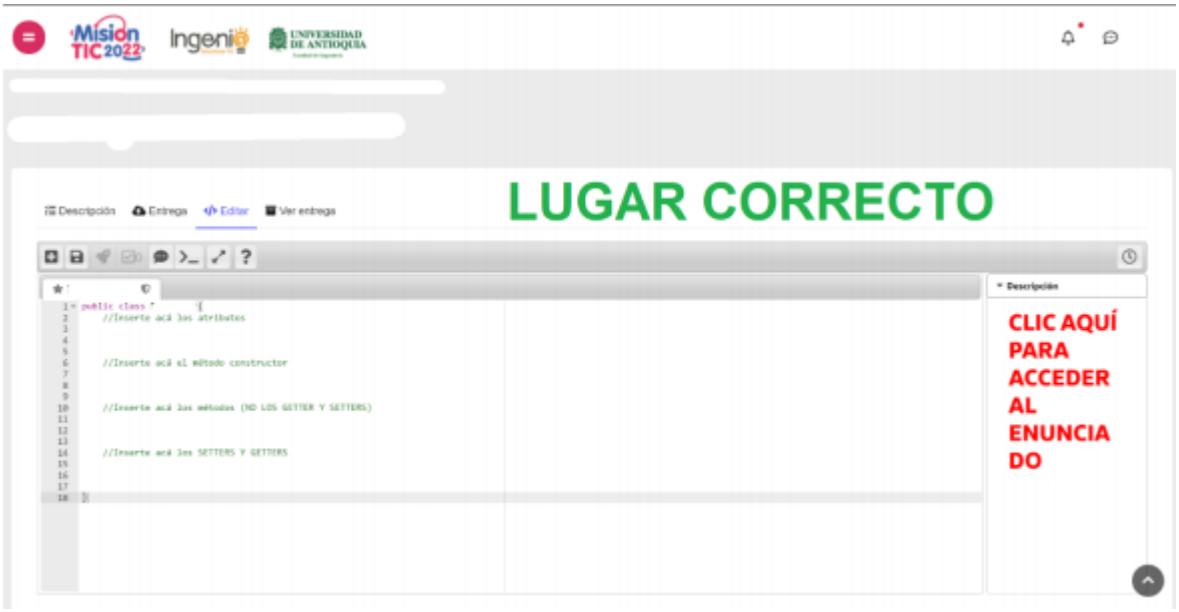
NOTA ACLARATORIA

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 3 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 versus cheating. 50th ACM Technical Symposium on Computer Science Education SIGCSE 2019*, Mineapolis, MN. DOI: 10.1145/3287324.3287443