Modelado y programación 2020-1
Proyecto

Fecha de entrega: 29 de noviembre de 2019

El proyecto se realizará de manera **individual**. Deberán escoger entre una de las siguientes propuestas.

Proyecto A

Crear un sistema para administración de rutas de autobuses y venta de boletos, donde habrá dos perfiles:

- **Administrador:** Puede dar de alta nuevas terminales (serán las paradas en las rutas) y crear rutas que pasen por distintas paradas. En cada ruta debe establecer cuál es la cantidad máxima de pasajeros que cabe en el autobús. Puede eliminar rutas si es que no hay boletos vendidos para dicha ruta. Podrá eliminar terminales si ninguna ruta pasa por dicha parada.
- **Vendedor:** Para hacer una venta deberá ingresar la cantidad de pasajeros y el nombre o identificador de dos paradas para obtener todas las rutas que pasen por ellas. Debe haber un filtro para descartar las rutas que no cuentan con la suficiente cantidad de asientos. Luego de escoger una ruta se finaliza la venta generando un identificador único para cada boleto. Para cancelar un boleto debe ingresar su identificador.

A tomar en cuenta:
El cálculo del precio de los boletos es libre, pueden establecer una tarifa única, una tarifa que varíe según el número de paradas, una tarifa que varíe según la ruta, etc.

Sea R una ruta con las paradas $A - B - C - D$ con capacidad máxima de 10 pasajeros. Si tiene vendidos 8 boletos con origen B y destino C no quiere decir que en toda la ruta hay 8 pasajeros, solo es en ese tramo, por lo que **sí** debe ser capaz de realizar una venta de 5 boletos con origen A y destino B, o 5 boletos con origen C y destino D. Lo que no debería ser posible es realizar una venta de 5 boletos con origen A y destino D porque en el tramo $B - C$ no hay suficiente cantidad de asientos.

Por último, las rutas solo van **en un solo sentido**, por lo que si se quiere hacer una venta con origen D y destino A, entonces R no debería listarse.

Proyecto B

Crear un juego conversacional de búsqueda de tesoro. El juego en ningún momento debe preguntarle al usuario sobre las acciones a realizar, es el usuario el que dé las instrucciones al jugador **explícitamente**.

- **Sobre el juego:** El jugador estará ubicado en una región de $n \times m$ casillas con el objetivo de encontrar el tesoro para ganar. Cada casilla puede estar vacía o contener algún objeto:
  - **Trampa:** Hiere al jugador.
  - **Poción:** Hiere o cura al jugador. El jugador no sabe qué efecto tendrá hasta que la beba.
  - **Obstáculo:** El jugador no puede pasar por esa casilla.
  - **Tesoro:** Si el jugador lo encuentra termina el juego.

  El juego no debe mostrarle en ningún momento al usuario el mapa del juego. No puede haber dos objetos en una misma casilla.

- **Desarrollo del juego:** Según lo que escriba el usuario en la terminal, es lo que hará el jugador. Ejemplo:

```
"El juego ha comenzado, buena suerte."
>> Avanzar 3 casillas a la derecha
>> Ir 2 espacios al norte.
"Caíste en una trampa, tu vida se reduce a 2".
>> Sube 2 espacios.
"Encontraste una poción"
>> Beber poción
"Te envenenaste, tu vida se reduce a 1."
>> Hacer 5 lagartijas
"No reconozco esa orden, pero suerte haciendo eso."
```

A tomar en cuenta:
Es elección libre cuánta vida tiene el jugador, así cómo qué tanto aumenta o disminuye según los objetos con lo que se encuentre. Si la vida del jugador llega a cero, el juego termina.

Pueden agregar más tipos de objetos al juego, pero solo se calificarán los que fueron solicitados.

Deberán incluir una documentación con el vocabulario aceptado por su programa, así como algunos ejemplos. El programa debe ser capaz de reconocer **al menos 20 palabras**. Además deben incluir un mapa de su región.

Restricciones del proyecto

1. Cualquier proyecto elegido deberá implementar al menos dos patrones de diseño vistos en clase.
2. Se deberá hacer uso de Bases de datos en el proyecto seleccionado.
3. Una vez seleccionado el proyecto y los patrones de diseño , no se podrán cambiar.

Lineamientos de entrega

1. Crear un método main que despliegue un menú que brinde al usuario la opción de interactuar con las funcionalidades del proyecto. El programa debe ser **robusto**, esto es, que sea a prueba de errores.

2. Incluir un archivo Readme.txt con una descripción del funcionamiento del programa.

3. Subir a la plataforma de Google Classroom el enlace a su repositorio de github.
   En caso de haber trabajado sin github podrán subir a la plataforma una carpeta comprimida con la solución de los ejercicios. El nombre de la carpeta comprimida debe ser el apellido paterno del alumno.

4. Deberán entregar en su plan de trabajo: el proyecto a realizar, los patrones de diseños seleccionados, el motivo por el cual escogieron dichos patrones de diseño y cómo son empleados en el proyecto.