Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ciencias y Sistemas
Lenguajes Formales y de Programación A
Ing. Luis Espino
Aux. Aylin Aroche
Vacaciones junio 2019

PROYECTO 2

Objetivos:
Este proyecto tiene como objetivo que el estudiante practique el análisis léxico y sintáctico de un lenguaje formal y que amplié sus conocimientos con el lenguaje de programación Java, así como aplicar los conocimientos adquiridos en la clase y en el laboratorio.

Descripción:
La empresa GSS (Galaxy Games Solutions) solicita sus servicios para crear un videojuego destinado al aprendizaje y el pensamiento estratégico. Este juego consiste en combatir entre los diferentes personajes que serán proporcionados por un archivo de entrada con comandos de un lenguaje específico, dichos comandos proporcionarán toda la información necesaria para cada personaje. Cada personaje tendrá valor de ataque, vida, imagen, entre otras características necesarias para poder ser un juego intuitivo. Además, cada usuario podrá escoger 3 personajes al inicio del juego para combatir con los personajes que se encuentran definidos en el archivo.

Los archivos de texto plano que almacenan el código fuente que recibe la aplicación están escritos en un lenguaje formal que será llamado GSS, el nombre hace referencia al nombre de la empresa. La aplicación debe proveer la opción de cargar archivos de texto con la extensión gss a un editor, y también debe existir la opción de analizar el contenido del editor para cargar el juego y los personajes.

Definición del lenguaje:
El lenguaje tiene varios bloques que se definen a continuación:

Bloque principal:
Este bloque contendrá toda la información principal del juego; personajes y rivales.

```
[Principal]:{
    //Bloque personaje
    //Bloque rival
}
```

A continuación, se detallan los elementos del bloque principal:

- ***Bloque Personaje***: El bloque personaje tendrá toda la información de personaje, se describirá posteriormente.

- ***Bloque Rival***: El bloque rival tendrá los nombres de los rivales a combatir, se describirá posteriormente.

Bloque Personaje:
Este bloque contendrá toda la información del personaje, pueden venir varios bloques personaje dentro del bloque principal.

```
[Personaje]:{
        [Nombre]:Jugador1;
        [Tipo]:Master;
        [Vida]:50;
        [Imagen]:"C:/Jugador1.jpg";
        [Sonido]:"C:/Sonido1.mp3";
        [Ataque]:{
            (Principiante,20),
            (Intermedio,15),
            (Master,10)
        }
    }
```

A continuación, se detallan los elementos del bloque personaje:

- ***Nombre:*** Este atributo definirá el nombre que represente al jugador. Debe empezar con una letra y puede contener más letras, números o guion bajo.
- ***Tipo:*** Este atributo representa al tipo del jugador, según el tipo los jugadores tendrán diferentes ataques hacia sus enemigos. Debe empezar con una letra y puede contener más letras, números o guion bajo.
- ***Vida***: Define la cantidad de vida que tendrá el personaje, debe ser un número entero mayor a 0.
- ***Imagen***: Define una ruta donde se encontrará una imagen que representará al personaje, debe de estar encerrado dentro de comillas dobles.
- ***Sonido:*** Define una ruta donde se encontrará un sonido que representará el ataque del personaje, debe de estar encerrado dentro de comillas dobles. Este elemento puede o no venir en el bloque personaje.
- ***Ataque:*** Este bloque define los diferentes ataques que tiene el jugador según el tipo del rival. Por ejemplo, si el rival es de tipo principiante, se le restará 20 puntos a su vida. Cada ataque estará encerrado entre paréntesis y separados por coma.

Bloque Rivales:
Este bloque tendrá una lista de los nombres de los personajes que serán los rivales, deben estar encerrados entre paréntesis y separados por coma. El último parámetro del rival es un número entero, y se refiere a la frecuencia en segundos en que el enemigo realizará un ataque.

```
[Rivales]:(Jugador5,Jugador3,Jugador4,3);
```

Bloque variables:
Las variables pueden utilizarse únicamente dentro de los bloques personaje para definir números enteros.

```
[Variables]:{
        //Declaración de variables
        //Asignación de variables
}
```

Declaración de variables:
Todas las variables declaradas son de tipo entero y en una sentencia de declaración se puede declarar más de una variable. Pueden ser utilizadas en cualquier bloque personaje y no se puede repetir el nombre de las variables.

```
[Variable]:a;
[Variable]:b,c,d;
```

Asignación de variables:
A todas las variables se les puede asignar un valor entero, el valor de otra variable o una expresión aritmética entre dos valores; que pueden ser números o variables. Las operaciones aritméticas permitidas son suma, resta, multiplicación y división. No se aceptan números negativos. El signo de asignación es “->”.

```
a->0;
b->2*5;
c->b+10;
d->c-5;
```

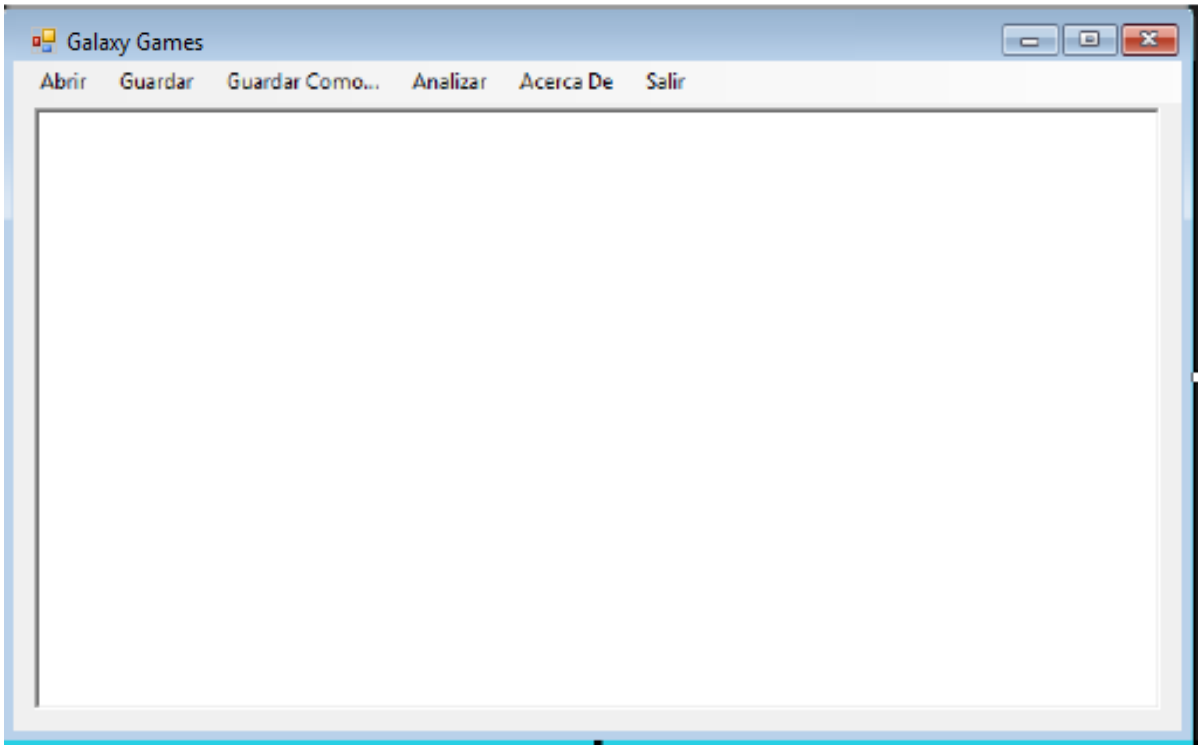
Archivo de entrada ejemplo:

```
[Principal]:{

        [Variables]:{
                [Variable]:a,b,c,d;
                a->0;
                b->2*5;
                c->b+b;
                d->c-5;
        }

        [Personaje]:{
                [Nombre]:Jugador1;
                [Tipo]:Master;
                [Vida]:50;
                [Imagen]:"C:/ Jugador1.jpg";
                [Sonido]:"C:/Sonido1.mp3";
                [Ataque]:{
```

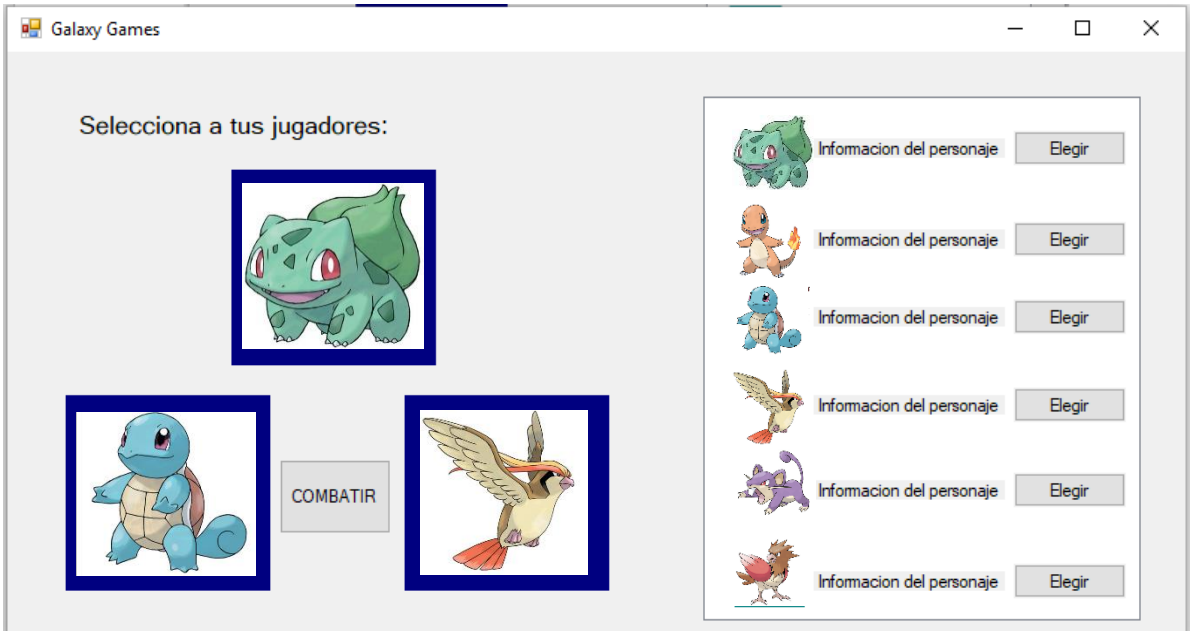
```
                (Principiante,c),
                (Intermedio,d),
                (Master,b)
        }
}

[Personaje]:{
        [Nombre]:Jugador2;
        [Tipo]:Intermedio;
        [Vida]:30;
        [Imagen]:"C:/ Jugador2.jpg";
        [Sonido]:"C:/Sonido1.mp3";
        [Ataque]:{
                (Principiante,b),
                (Intermedio,8),
                (Master,5)
        }
}

[Personaje]:{
        [Nombre]:Jugador3;
        [Tipo]:Principiante;
        [Vida]:c;
        [Imagen]:"C:/Jugador3.jpg";
        [Ataque]:{
                (Principiante,5),
                (Intermedio,7),
                (Master,b)
        }
}

[Personaje]:{
        [Nombre]:Jugador4;
        [Tipo]:Intermedio;
        [Vida]:30;
        [Imagen]:"C:/Jugador4.jpg";
        [Sonido]:"C:/Sonido1.mp3";
        [Ataque]:{
                (Principiante,b),
                (Intermedio,8),
                (Master,5)
        }
}
```

```
        [Personaje]:{
                [Nombre]:Jugador5;
                [Tipo]:Principiante;
                [Vida]:c;
                [Imagen]:"C:/Jugador5.jpg";
                [Sonido]:"C:/Sonido1.mp3";
                [Ataque]:{
                        (Principiante,5),
                        (Intermedio,7),
                        (Master,b)
                }
        }

        [Rivales]:(Jugador5,Jugador3,Jugador4,3);
}
```

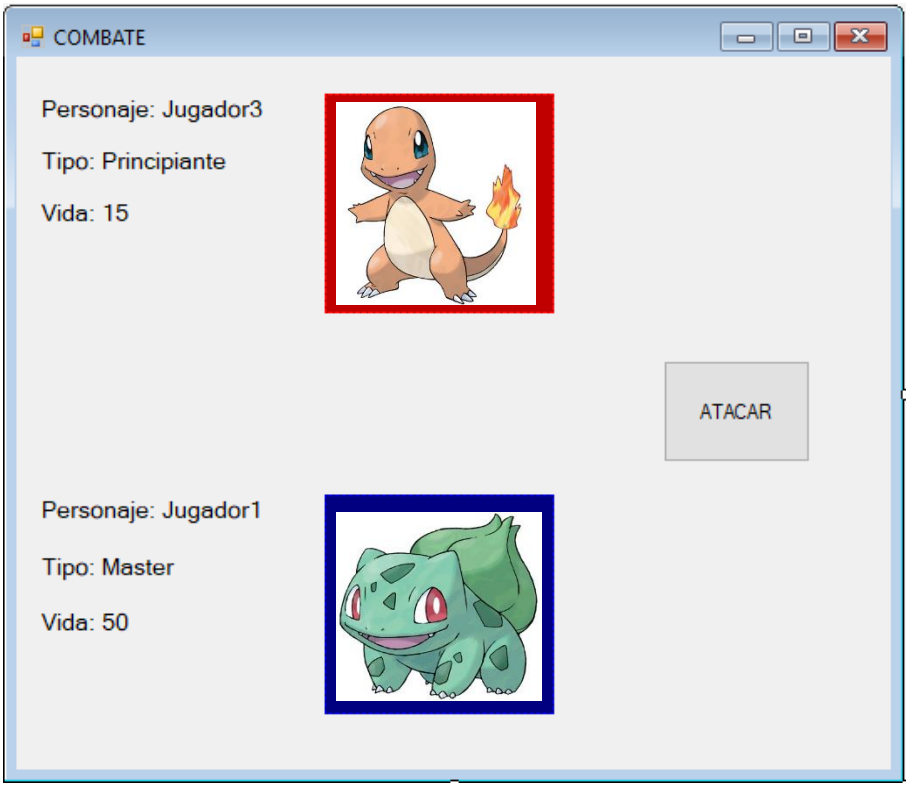
Interfaz gráfica:



Al ejecutar las instrucciones del editor se debe de mostrar la siguiente pantalla:



En está pantalla se mostrará 3 personajes al azar (random) para que el usuario pueda empezar la partida, también debe mostrar una lista con todos los personajes y sus características para que el usuario pueda escoger cualquier otro personaje. No puede escoger el mismo personaje dos veces.



Se debe mostrar en todo momento el nombre, tipo, vida y ataques del personaje.

Funcionalidad del juego:

- En el tablero debe existir un botón para atacar al rival, en donde al atacar se deberá restar los puntos correspondientes al rival. Al momento de atacar se debe de escuchar el sonido descrito en el bloque personaje.
- El rival tendrá un tiempo definido para atacar al personaje, cada cierto tiempo deberá atacar en donde restará los puntos al personaje y se debe de escuchar el sonido del rival.
- Si la vida del personaje llega a cero, este morirá y deberá de cambiar entre los personajes que se escogieron al principio del juego.
- Los rivales deben ir apareciendo en el orden que aparece el archivo de entrada, si muere alguno debe de continuar con el siguiente personaje que aparece en el bloque rivales.
- El juego se gana cuando el personaje vence a todos los rivales, debe mostrar un mensaje que se ha ganado el juego.
- El juego se pierde cuando el último personaje disponible se queda sin vida, debe mostrar un mensaje indicando que se ha perdido el juego y la puntuación total.
- Al momento de perder o ganar el juego, se debe de preguntar al usuario si desea iniciar otro combate, si selecciona “si” mostrará la pantalla para seleccionar nuevamente los personajes de la lista y selecciona “no” cerrará el juego.
- Se evaluará la creatividad del estudiante al momento de crear la interfaz.

Aplicación:

La aplicación debe analizar el texto del editor, de encontrar errores léxicos o sintácticos, la aplicación debe informar al usuario y mostrar un reporte de errores en formato HTML, en este caso no debe mostrar la ventana del juego. En la tabla del reporte de errores se deben mostrar los siguientes campos:

No.	Error	Tipo	Descripción	Fila	Columna
1	%	Léxico	Elemento léxico desconocido	3	4
2	(	Sintáctico	Se esperaba {	8	10

De no encontrar error alguno, se debe generar la ventana de salida correspondiente. La aplicación debe pintar dentro del editor cada uno de los lexemas, según su tipo, los colores se definen a continuación:

Tipo	Color
Palabras reservadas	Celeste
Números	Rojo
Cadenas	Verde
Identificadores	Anaranjado
Llaves, corchetes y paréntesis	Morado
Punto y coma, dos puntos y coma	Café
Signo de asignación (->)	Azul
Signos aritméticos	Amarillo
Otros	Negro

Además de colorear los elementos léxicos y generar la ventana del juego, se debe generar otra página HTML que contenga una Tabla de tokens con los siguientes campos:

No.	Token	Lexema	Tipo	Fila	Columna
1	5	{	Llave abierta	2	1
2	19	Variable	Palabra reservada	2	12
3	8	:	Dos puntos	2	11

Interfaz gráfica:

Componentes mínimos de la interfaz:

1. **Editor de texto:** Debe ser un área de texto en el cual se puedan escribir nuevas instrucciones de entrada o bien cargar texto desde archivos con extensión .gss
2. **Abrir:** La interfaz debe proveer la capacidad de abrir archivos con extensión .gss
3. **Guardar:** La aplicación debe proveer la capacidad de guardar el texto contenido en el editor. Si no ha sido guardada nunca, debe proveer una opción para ingresar el nombre del nuevo archivo y la ubicación en la que se desea guardar.
4. **Guardar como:** Está opción permite guardar el archivo de entrada con otro nombre, se debe preguntar el nombre del nuevo archivo.
5. **Analizar:** Debe realizar el análisis léxico y sintáctico del lenguaje que se encuentra actualmente en el editor y generar la salida correspondiente.
6. **Botones:** Deberá tener los botones necesarios para la ejecución del juego.
7. **Acerca de:** Debe desplegar una ventana con los datos del estudiante y del curso.
8. **Salir:** Debe terminar la ejecución de la aplicación.

Entregables:

- Manual de Usuario
- Manual Técnico:
    - Plataforma de ejecución
    - Diagrama o diccionario de clases
    - Herramientas utilizadas
    - El autómata finito determinista que se utilizó para la implementación del analizador léxico. Para la obtención del autómata se debe de utilizar expresiones regulares y el método del árbol, los cuales también deben de adjuntarse en este documento.
    - Se debe de adjuntar la gramática independiente del contexto utilizada para la implementación del analizador sintáctico.
- Código fuente.
- Ejecutable del proyecto.

Documentación a entregar de forma física el día de la calificación:

- Hoja de calificación (Original y una copia)

Notas importantes:

- El proyecto se debe desarrollar de forma individual.
- Este proyecto se deberá desarrollar utilizando Java con Netbeans.
- No se debe de utilizar ninguna herramienta al momento de realizar los analizadores.

- El proceso de obtener tokens, se debe hacer a través de la implementación del autómata finito determinista desarrollado por el propio estudiante a través del método del árbol y expresiones regulares.
- **El analizador sintáctico debe ser programado por el estudiante, y el código fuente debe coincidir con la gramática tipo 2 que presente en el manual técnico.**
- Sin analizador sintáctico NO se calificará.
- La calificación se realizará ÚNICAMENTE con los archivos que el auxiliar provea en el momento de la calificación, dichos archivos no se deben modificar.
- La calificación del proyecto será personal y durará como máximo 30 minutos, en un horario que posteriormente será establecido. Se debe tomar en cuenta que durante la calificación no podrán estar terceras personas alrededor, de lo contrario no se calificará el proyecto.
- No se dará prórroga para la entrega del proyecto.
- **Copia parcial o total del proyecto tendrá una nota de 0 puntos, y se notificará a la Escuela de Sistemas para que se apliquen las sanciones correspondientes.**
- **En el caso de no cumplir con alguna de las indicaciones antes mencionadas, NO se calificará el proyecto; por lo cual, se tendrá una nota de cero puntos.**

Fecha de entrega: 28 de junio del 2019