Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería en Ciencias y Sistemas
Organización de Lenguajes y Compiladores 2
Primer semestre 2019

Catedráticos: Ing. Bayron López, Ing. Erick Navarro y Ing. Edgar Saban
Tutores académicos: Mike Gutiérrez, Javier Navarro, Julio Arango

Creator XML

Primer proyecto de laboratorio

Contenido

1 Aclaraciones Generales

- TODAS las palabras reservadas van sin tildes tanto como para gxml y FuncionScript
- TODAS las rutas son relativas únicamente, son relativas a la carpeta del archivo que se está analizando
- Las funciones **FILTRAR, BUSCAR, REDUCE, TODOS, ALGUNO**, tienen el mismo comportamiento que la funcion map que se encuentra en el ejemplo de la sección 2.5.5.1 específicamente en el paso 4.
- La función crear botón también recibe los parámetros opcionales **alto y ancho.**

1.1 Composición archivo gdato

Sintaxis

- Cada principal es un guardo de la ventana
- Una ventana tendrá un gdato

```
<lista>
   <principal>
      <nombreControl>”nombre intento 1”</ nombreControl >
      < nombreControl >”apellido intento 1”</ nombreControl >
      < nombreControl >”teléfono intento 1”</ nombreControl >
      < nombreControl >”Guatemala”</ nombreControl >
   </principal>
   <principal>
      < nombreControl >”nombre intento 2”</ nombreControl >
      < nombreControl >”apellido intento 2”</ nombreControl >
      < nombreControl >”teléfono intento 2”</ nombreControl >
      < nombreControl >”Guatemala”</ nombreControl >
   </principal>
   <principal>
      …
   </principal>
</lista>
```

- Gdato tendrá 2 tipos de datos, los cuales coinciden con los de funcionscript
    - Cadena (encerrado entre comillas)
    - Numérico (unicamente el numero)

Tabla Control con el tipo de dato que guardará

Tipo Control	Tipo Dato a guardar
Texto	Cadena
Numérico	Numérico
Desplegable	Cadena
TextoArea	Cadena

1.1.1.1Definición del ejemplo

Los Textos van dentro de contenedores y no de controles

```
<ventana id="principal" tipo="principal">
  <contenedor id="Uno">
    <Texto>
      Nombre
    </Texto>
    <control tipo="texto" alto="25" ancho="40" nombre="primer nombre">
                                                    Definición de nombre
    </Control>
   <Texto>
      Apellido
    </Texto>
    <Control type="texto" alto="25" ancho="40" nombre="Segundo nombre">
                                                    Definición de apellido
    </Control>
    <Texto>
      Telefono
    </Texto>
    <Control type="texto" alto="25" ancho="40" nombre="teléfono">
                                                    Definición de teléfono
      <Defecto>
        4545-4040          Teléfono por defecto
      </Defecto>
    </Control>
    <Texto>
      Ciudad
    </Texto>
    <Control tipo="desplegable" alto="25" ancho="40" nombre="país">
      <ListaDatos>
        <Dato>Guatemala</ Dato >
        < Dato >El Salvador</ Dato >
        < Dato >Honduras</ Dato >
        < Dato >Costa Rica</ Dato >
      </ListaDatos>
    </Control>
  </Contenedor>
```

Opciones de País

Definición de País

```
    <Contenedor id="dos">
        <Enviar alto="25" ancho="40">
            <texto>
                Enviar
            </texto>
        </Enviar>
    </contenedor>
</ventana>
```

Definición del botón Enviar

2 GenericXML

2.1 Lenguaje

- Dentro de las etiquetas que pueden contener texto o información entre etiquetas <Etiqueta> **TEXTO** </Etiqueta>, para llevar un buen manejo en la gramática, en ese texto no vendrá el carácter '<'
- Todos los valores de las etiquetas tendrán de sintaxis el tipo de valor que le corresponde, esto quiere decir que si es ancho y alto es numérico, Alto = 12 Ancho = 12 y como id es cadena, id = "identificador", a continuación, les presento la tabla con todos los elementos y su tipo.
- Los elementos no soportaran operaciones, únicamente los parámetros de las llamadas a los métodos soportaran operaciones aritméticas, relacionales y lógicas

Elemento	Tipo	Valor por defecto
Id	Cadena	Invalido
Tipo	Cadena	Invalido
Color	Cadena	Descrito en cada etiqueta
AccionInicial	{ Metodo(Parametros) }	Invalido
AccionFinal	{ Metodo(Parametros) }	Invalido
X	Numerico	Invalido
Y	Numerico	Invalido
Alto	Numerico	Descrito en cada etiqueta
Ancho	Numerico	Descrito en cada etiqueta
Borde	Booleano	Falso
Nombre	Cadena	Invalido
Fuente	Cadena	Arial
Tam	Numerico	14
Negrita	Booleano	Falso
Cursiva	Booleano	Falso
Maximo	Numerico	nulo
Minimo	Numerico	nulo
Accion	{ Metodo(Parametros) }	Invalido
Referencia	Cadena	Invalido

Path	Cadena	Invalido
Auto-Reproduccion	Booleano	Falso

2.2 Estructura del lenguaje

Etiqueta Importar

Para ventanas con el mismo id se toma de prioridad las del archivo que está importando, en dado caso el conflicto sea entre 2 importados la prioridad será según el orden de las importaciones (mayor prioridad a la primera importación)

Etiqueta Ventana

- Ventana contiene 3 opcionales, no solo 1 y pueden venir de 0 a 3 opcionales en la misma etiqueta.

Controladores

- La etiqueta control de tipo texto ya no tendrá el elemento acción

Etiqueta dato

- La etiqueta dato ya no tendrá referencia

Etiqueta Lista Datos

- Esta etiqueta ya no tendrá el elemento acción

Etiqueta Defecto

Multimedia

- Extensiones permitidas
  - Imagen: BMP, JPG, TPG, PNG
  - Audio: WAV, MP3
  - Video: AVI, MP4, MEPG, RM, FLV

Etiqueta Botón

En la etiqueta Botón, si viene una etiqueta texto el **único** elemento obligatorio del texto es **nombre**

Etiqueta Enviar

La etiqueta Enviar primero recolecta la información, segundo puede ejecutar una función si es que la trae (acción) y tercero puede hacer referencia a otra ventana, todo esto se ejecutará en este mismo orden

3 FuncionScript

- Se **elimina** el tipo de dato carácter
- El dato **undefined** es para las declaraciones realizadas sin valor
- Los arreglos **no** pueden contener objetos, todo lo obtenido por LeerGxml y CrearArrayDesdeArchivo son **estructuras de solo lectura**, inmutables, estos si pueden venir dentro de arreglos, **no** se hará arreglo[].item
- **No** hay objetos dentro de objetos
- En los objetos pueden venir arreglos solo de dato primitivo
- **Todas** las referencias pueden ser llamadas a métodos
- Se quitan **Todo** tipo de funciones lambda
- Los atributos de ventana, contenedor, boton, y etiquetas y todo lo que pueda devolver un LeerGxml, **NO** son palabras reservadas, se manejan como si fueran atributos de objetos comunes.
- Entre variables y funciones pueden tener el mismo id
- Todas las funciones de arreglos que retornan un valor, puede tener otra función encadenada

```
Var array = [18,20,1,243,31];

Var array1 = array.map(id).filtrar(id2)
```

- Los parámetros pueden ser todo tipo de variable, objeto, arreglos, ventanas, contenedores, etc
- Si pueden existir objetos dentro de funciones y solo existirán en ese entorno, funcionan igual que toda variable
- Al cargar una ventana, la ejecución no se detiene, se sigue ejecutando en top down

3.1 Notación dentro del enunciado

3.1.1 Variables

- Las declaraciones de variables pueden venir fuera de una función, es decir se podrá declarar una función en cualquier parte de un archivo FuncionScript.
- Cuando vienen múltiples variables en una misma declaración y una asignación al final, solo la última variable toma el valor.

3.1.2 Tipos de datos

- Se tomará en cuenta que verdadero > falso

3.2 Sintaxis

3.2.1 Bloque de Sentencias

- Todas las sentencias de control pueden ir fuera de una función, es decir que podrán escribir un “Si” fuera del cuerpo de una función.

3.2.2 Operaciones aritméticas

Potencia

- No se pueden operar valores booleanos

Aumento

- Solo va afectar variables

```
Id++;
A = id++:
Id[#]++:
Id.id++:
```

- Si estos operadores son usados para expresiones, primero devuelve el valor del identificador y luego incrementa el identificador

```
Var a = 1;
Var b = a++:  // A queda con valor de 2 y b con valor de 1
Var a = 5 + b++; // b queda con valor de 2 y a con valor de 6
```

Decremento

- Solo va afectar variables

```
Id--;
A = id--:
Id[#]--:
Id.id--:
```

- Si estos operadores son usados para expresiones, primero devuelve el valor del identificador y luego decrementa el identificador

```
Var a = 1;
Var b = a--:  // A queda con valor de 0 y b con valor de 1
Var a = 5 + b--; // b queda con valor de 1 y a con valor de 5
```

Asignación y operación

3.2.3 Operadores relacionales

- Se tomará en cuenta que verdadero > falso
- Para poder realizar operaciones con los signos relacionales (“==”, “<”, “>”, “<=”, “>=”, “!=”) ambos lados de la expresión deben de ser del mismo tipo.
- Con el operador igual que se puede comparar valores nulos

```
Variable == nulo //verdadero o falso
```

3.2.4 Operaciones lógicas

Precedencia de operadores

Cualquier duda tomar en base Java

Nivel	Operador	Asociatividad
1	? (ternario) :	Right to left
2	|| (Or)	Left to Right
3	&& (And)	Left to Right
4	! (not)	No associative
5	== (igual que) != (diferente que)	Left to Right
6	<, >, <=, >=	Left to Right
7	+, -	Left to Right
8	* , /	Left to Right
9	^	No associative
10	++,--, -(unario), +(unario)	Right to Left

3.2.5 Declaración de variables

Cuando se declara una lista de variables y se da un valor de inicialización únicamente la última variable se le asignara ese valor.

3.2.6 Arreglos

Asignación en una posición del arreglo

1. La siguiente sintaxis no es válida **“MetodoNombre()[pos]”**
2. La siguiente sintaxis no es válida **“MetodoNombre().atr[pos]”**
3. La siguiente sintaxis si es válida **“MetodoNombre().atr”**

```
Variable[posicion] = Expresion:
```

Obtención del valor en una posición del arreglo

```
variable[posicion]

Ejemplos:
Variable = variable[posicion]
Metodo(variable[posicion])
```

3.2.7 Importar

Para funciones con el mismo nombre se toma de prioridad las del archivo que está importando, en dado caso el conflicto sea entre 2 importados la prioridad será según el orden de las importaciones (mayor prioridad a la primera importación)

3.2.8 Sentencias de selección

Selecciona

- El flujo de la sentencia Selecciona es que evalúa todos los casos, si entra a uno y este tiene detener se sale, si entra y no tiene detener sigue evaluando los demás casos, **unicamente** entra al defecto si no entro a ningún caso
- El entorno es por caso.

3.2.9 Funciones y procedimientos

Cuando se define una función o procedimiento, no es necesario definir los tipos de los parámetros, este es un error de redacción.

Los parámetros dentro de una función están definidos únicamente por el nombre del parámetro, es decir que no se antepone la palabra var, ver ejemplo. Ejemplo

```
funcion suma(var operador1, var operador2) {
    retornar operador1 + operador2;
}
```

3.2.10 Llamada a procedimientos y funciones

No serán permitidos las variables contenedoras de funciones (lambda)

```
funcion suma(var operador1, var operador2) {
   retornar operador1 + operador2;
}

Var suma = (x,y)=>x+y;
```

3.3 Funciones con arreglos propias de FuncionScript

Se puede asignar y acceder a los valores de una posición de arreglo con la siguiente sintaxis

```
Arreglo[#] = valor;  // asigna valor en arreglo en posición #
Var a = Arreglo[#]; // asigna a A el valor de Arreglo[#]
```

- Si una función map, reduce, buscar, filtrar, todos y alguno reciben una función con más o menos parámetros de los definidos por su función nativa, se deberá marcar error.
- Si la función a la cual llamaron no tiene retorno, estos devolverán un arreglo vacío.

3.3.1 Crear array desde archivo

Se le agrega una funcionalidad a la función, donde si la función viene sin parámetros esta guardara la información de los controles de esa ventana

SINTAXIS

```
Ventana.crearArrayDesdeArchivo();
```

3.3.2 ObtenerPorNombre

Obtener por Nombre tambien tendrá de parámetro el id de la ventana del cual se quiere recuperar el nombre, este id vendrá como cadena.

```
Var nodo = archivo.ObtenerPorNombre("nombre", id);
```

Ejemplo

```
Var nodo = archivo.ObtenerPorNombre("boton1", "ventanaInicio"); // retorna el nodo del objeto gxml que tenga de nombre boton1  de la con el id ventanaInicio
```

3.3.3 Crear Ventana

- **Alto**: Parámetro de tipo entero que definirá el alto de la ventana.
- **Ancho**: Parámetro de tipo entero que definirá el ancho de la ventana.
- Se le agrega el parámetro id a la ventana, esto para poder guardar el gdato, la dirección de gdato por ventana es id + “.gdato”

Sintaxis

```
Var ventana1 = CrearVentana(“Color hexadecimal”, Alto, ancho, id)
```

```
Var ventana1 = CrearVentana(“#000000”, 500,500,”VentanaUno”);
Ventana1.CrearArrayDesdeArchivo();// dirección de Gdato donde se va a guardar: VentanaUno.gdato
```

3.3.4 CrearContenedor

3.3.5 Crear Texto

Se le agrega el parámetro “valor” a la función, este parámetro es de tipo cadena y recibe el texto que se mostrará

Se quita el parámetro referencia debido a que en la etiqueta no viene este elemento

Sintaxis

```
Contenedor.CrearTexto(Fuente, Tamaño, Color, X, Y, Negrilla, Cursiva, valor)
```

3.3.6 Crear Caja texto

Se le agrega el parámetro “defecto” a la función, este parámetro es de tipo cadena y recibe el texto que se mostrará si trae texto por defecto

Se le agrega el parámetro “nombre” a la función, este parámetro es de tipo cadena y es el nombre del control

Sintaxis

```
Contenedor.CrearCajaTexto(Alto, Ancho, Fuente, Tamaño, Color, X, Y, Negrilla, Cursiva, defecto, nombre)
```

3.3.7 Crear Área Texto

Se le agrega el parámetro “defecto” a la función, este parámetro es de tipo cadena y recibe el texto que se mostrará si trae texto por defecto

Se le agrega el parámetro “nombre” a la función, este parámetro es de tipo cadena y es el nombre del control

Sintaxis

```
Contenedor.CrearAreaTexto(Alto, ancho, Fuente, Tamaño, Color, X, Y, Negrilla, Cursiva, defecto, nombre)
```

3.3.8 Crear Control Numérico

Se le agrega el parámetro “defecto” a la función, este parámetro es de tipo entero y recibe el texto que se mostrará si trae texto por defecto

Se le agrega el parámetro “nombre” a la función, este parámetro es de tipo cadena y es el nombre del control

Sintaxis

```
Contenedor.CrearControlNumerico(Alto, Ancho, Maximo, Minimo, X, Y, defecto, nombre)
```

3.3.9 CrearDesplegable

- Sintaxis para acceder a la lista de datos:

```
Variable.lista // esta sera la lista de datos, lo devolverá en forma de arreglo
Variable.lista[#] // esta sera un acceso a la lista de datos
```

- No se puede acceder a los atributos de los datos debido a que no tiene elementos

Sintaxis

```
~~Desplegable.lista[0].atributo = ##;~~
```

- La palabra reservada para el acceso a la lista de un desplegable es lista, esta retornara un arreglo de cadenas con todos los datos de la lista

- Se le agrega el parámetro “nombre” a la función, este parámetro es de tipo cadena y es el nombre del control

- Este en lista recibirá un arreglo de cadenas

Sintaxis

```
Contenedor.CrearDesplegable(Alto, Ancho, lista, X, Y, Defecto, nombre)
```

3.3.10 Crear Botón

Se le agrega el parámetro “valor” a la función, este parámetro es de tipo cadena y recibe el texto que se mostrará

Se le agrega Alto y Ancho al botón

Sintaxis

```
Var bot = Contenedor.CrearBoton(Fuente, Tamaño, Color, X, Y,Referencia, valor, Alto,
        Ancho)
Bot.AlClic(Metodo(12));
```

3.3.11 Crear Imagen

- **Alto**: Parámetro de tipo entero que definirá el alto del multimedia.
- **Ancho**: Parámetro de tipo entero que definirá el ancho del multimedia.

Sintaxis

```
Contenedor.CrearImagen(Ruta, X, Y, Alto, Ancho)
```

3.3.12 Crear Reproductor

- **Auto-reproductor**: Parámetro de tipo booleano que representará si habrá auto reproducción.
- **Alto**: Parámetro de tipo entero que definirá el alto del multimedia.
- **Ancho**: Parámetro de tipo entero que definirá el ancho del multimedia.

Sintaxis

```
Contenedor. CrearReproductor (Ruta, X, Y, Auto-reproductor, Alto, Ancho)
```

3.3.13 Crear Video

- **Auto-reproductor**: Parámetro de tipo booleano que representará si habrá auto reproducción.
- **Alto**: Parámetro de tipo entero que definirá el alto del multimedia.
- **Ancho**: Parámetro de tipo entero que definirá el ancho del multimedia.

Sintaxis

```
Contenedor. CrearVideo (Ruta, X, Y, Auto-repr oductor, Alto, Ancho)
```

3.4 Eventos

- La opción de ACCION de las etiquetas, se define de la siguiente manera accion = {metodo()} donde solo puede venir la llamada a un método.
- Los eventos si son llamados más de una vez, el método al cual se les setea se sobre escribe

```
Ventana.AlCargar(Metodo1());
Ventana.AlCargar(Metodo2()); // el método que ejecutará al cargar es Metodo2 únicamente
```

3.4.1 Al Cargar

El evento si viene sin parámetros abrirá la ventana sobre la cual está siendo llamada y cerrará toda aquella que este abierta en ese momento

```
Ventana.AlCargar();   // Abrira la ventana
```

3.4.2 Al Cerrar

El evento si viene sin parámetros cerrará la ventana sobre la cual está siendo llamada

```
Ventana.AlCerrar();   // Cerrara la ventana
```