Basic Object Oriented Programming (BOOP)

Gramática del Lenguaje

Ever Mitta Flores

Índice

Gramática del Lenguaje BOOP

A continuación, se presenta algunos ejemplos de la gramática manejada por el lenguaje implementado.

1. Clase @Principal

La clase principal es la clase inicial, es decir es la clase que se ejecutará e interpretará primero. Dentro de ésta clase se podrán definir atributos y métodos; sin embargo, cabe resaltar que su método de inicio es el método "#ejecutar". La Ilustración 4.1 muestra la estructura básica de la clase principal.

```
@principal {
        /** Sección Atributos **/

        /** Sección Métodos **/

        #ejecutar{
                /** Líneas de Código **/
        }
}
```

Ilustración 1.1. Definición de Métodos

2. Comentarios

Un comentario es un conjunto de caracteres delimitados por la nomenclatura barra-asterisco " /* " y " */ " , todo el código introducido dentro de estos delimitadores no serán considerados al momento de compilación y ejecución. La Ilustración 4.2 muestra un ejemplo del uso de comentarios.

```
/*Declaración de Variables*/
$entero edad; /* edad en años de la persona */
$entero dni;/* dni de la persona */
```

Ilustración 2.1. Uso de Comentarios en BOOP

3. Definición de Atributos

Para definir un atributo dentro de una clase, se debe colocar el identificador #atributo seguido del símbolo : , seguido de las características del atributo. Las características del atributo son (identificador de acceso, tipo de dato y nombre).

La Ilustración 4.3 muestra como se debe realizar la definición de atributos.

```
$clase @Persona {
        #atributo:
        $publico $entero edad;

        #atributo:
        $privado $entero dni;
}
```

Ilustración 3.1. Definición de Atributos

4. Definición de Métodos

Para definir un método dentro de una clase, se debe colocar el identificador "**#metodo**" seguido del signo dos puntos "**:**" , seguido de las características del método. Las características del método son (identificador de acceso, tipo de dato a devolver, nombre del método, lista de parámetros limitados por paréntesis y su bloque de código correspondiente).

La Ilustración 4.4 muestra como se debe realizar la definición de métodos.

```
$clase @Persona {
        #metodo:
        $publico $nada imprimeEdad(){
                /*Lineas de Código*/
        }

        #metodo:
        $publico $entero duplicaEdad($entero edad){
                /*Lineas de Código*/
        }
}
```

Ilustración 4.1. Definición de Métodos

5. Declaración de Variables

Para la definición de variables se procede con colocar el tipo de dato seguido de la variable a declarar y finalizar la sentencia con un punto y coma.

Los tipos de datos con los que se cuentan $entero que hace referencia a números enteros, $decimal que hace referencia a números reales. Asimismo, una variable es un conjunto de letras alfanuméricas, teniendo como restricción que la letra inicial no puede ser un número.

La Ilustración 4.5 muestra como se debe realizar la declaración de variables.

```
$entero edad;
$entero dni;
$decimal estatura;
$decimal peso;
```

Ilustración 5.1. Declaración de Variables en BOOP

6. Asignación de Valores

Para la asignación de valores, se hará uso del signo igual "**=**". La Ilustración 4.6 muestra como se debe realizar la asignación de valores a las variables definidas.

```
edad = 22;
dni = 46410647;
estatura = 1.69;
```

Ilustración 6.1. Asignación de Valores en BOOP

7. Definición de Clases

Para definir una clase, se debe colocar el identificador $clase seguido del nombre de la clase (los nombres de las clases deben empezar con el símbolo arroba "@", seguido del su bloque de código correspondiente (bloque delimitado por dos llaves "**{**" y "**}**").

La Ilustración 4.7 muestra un ejemplo de la definición de clases.

```
$clase @Persona{
        /** SECCIÓN ATRIBUTOS **/
        #atributo:
        $publico $entero x=5;

        /** SECCIÓN MÉTODOS**/
        #metodo:
        $publico $noRetorno miPrimerMetodo(){
                /***CÓDIGO**/
                /***CÓDIGO**/
                /***CÓDIGO**/
        }
}
```

Ilustración 7.1. Definición de Clase Simple en BOOP

Asimismo si una clase quiere hacer uso de la herencia, se debe colocar la sección de herencia después del nombre de la clase; esta sección se denota colocando el identificador "**$hereda**" seguido de las clases a heredar. La Ilustración 4.8 muestra dos ejemplos de la herencia de clases.

```
$clase @Perro $hereda @Animal{
        /** CÓDIGO **/
        /** CÓDIGO **/
}

$clase @Instructor $hereda @Alumno , @Profesor{
        /** CÓDIGO **/
        /** CÓDIGO **/
}
```

Ilustración 7.2. Definición de Clases con Herencia

En caso de querer hacer uso de la implementación de interfaces, se deberá colocar después del nombre de la clase, el identificador "**$implementa**" seguido de la interfaz a implementar.

La Ilustración 4.9 muestra dos ejemplos de la implementación de clases:

```
$clase @BoletaDePago $implementa @Imprimible {
        /*Lineas de Código*/
}

$clase @Animal $implementa @Movimiento , @Sonido {
        /*Lineas de Código*/
}
```

Ilustración 7.3. Definición de Clases con Interfaces

8. Llamar a Atributos

Para utilizar un atributo definido dentro de la clase en la que se está trabajando, se tiene que hacer uso del identificador "**$entorno**", el cual se acompaña del signo punto "**.**" y es seguido por el nombre del atributo.

Asimismo, en el caso de querer llamar a un atributo perteneciente a un objeto, se nombrará al objeto seguido del signo punto y se colocará el nombre del atributo

La Ilustración 4.10 muestra un ejemplo de la llamada a atributos.

```
@principal {
        #atributo:
        $publico $entero miNumero;

        #ejecutar{
                $entorno.miNumero = 20;
        }
}
```

Ilustración 8.1. Llamar a Atributo de la Clase

La Ilustración 4.11 muestra un ejemplo de la llamada a atributos pertenecientes a un objeto.

```
$clase @Persona {
        #atributo:
        $publico $entero edad;
}

@principal {
        #atributo:
                $publico @Persona persona1;
        #ejecutar{
                $entorno.persona1.edad = 50;
        }
}
```

Ilustración 8.2. Llamar a Atributo de un Objeto

9. Llamar a Métodos

Para realizar una llamada a un método se tiene que hacer uso del identificador "**$llamar**", seguido de dos corchetes "**[**" y "**]**" que contengan el nombre de la función a llamar.

Para colocar el nombre de la función, se debe utilizar la misma estructura utilizada para el llamado de atributos (uso de $entorno o llamada desde objeto).

La Ilustración 4.12 muestra un ejemplo de llamada a métodos propios de una clase. Mientras que la Ilustración 4.13 muestra un ejemplo de llamada a métodos pertenecientes a un objeto.

```
@principal {
        #metodo:
        $publico $entero imprimeCabezera(){
                /** CÓDIGO **/
        }

        #ejecutar{
                $llamar[$entorno.imprimeCabezera()];
        }
}
```

Ilustración 9.1. Llamar a Método de la Clase

```
$clase @Persona{
        /** CÓDIGO **/
        #metodo:
        $publico $entero imprimeNombre(){
                /** CÓDIGO **/
        }
        /** CÓDIGO **/
}

@principal {
        #ejecutar{
                @Persona miPersona;
                /** CÓDIGO **/
                $llamar[miPersona.imprimeNombre()];
        }
}
```

Ilustración 9.2. Llamar a Método de un Objeto

En caso de que el método a llamar retorne un valor, se hará uso del signo igual "**=**" antes de realizar la llamada al método. La Ilustración 4.14 muestra un ejemplo de llamada a un método con retorno.

```
@principal {
        #metodo:
        $publico $entero calculaSuma($entero a, $entero b){
                $retorna (a+b);
        }

        #ejecutar{
                $entero sumaAB;
                sumaAB = $llamar[$entorno.calculaSuma(10,15)];
        }
}
```

Ilustración 9.3. Llamar a Método con Retorno

10. Operaciones Matemáticas

El lenguaje también cuenta con funciones matemáticas propias. Entre ellas están las funciones básicas, tales como suma, resta, multiplicación, división, división entera, residuo, potencia.

La Ilustración 4.15 muestra un ejemplo del uso de todas las operaciones matemáticas posibles.

```
@principal {
   #ejecutar{
     $entero a;                          $entero b;
     $entero sumaAB;           $entero restaAB;
     $entero multiplicacionAB; $decimal divisionAB;
     $entero divisionEnteraAB; $entero residuoAB;
     $entero potenciaAB;

     a=12;b=5;

     sumaAB = a+b;                       /** resultado=17 **/
     restaAB = a-b;                      /** resultado=12 **/

     multiplicacionAB = a*b;             /** resultado=60 **/
     divisionAB= a/b;                    /** resultado=2.4 **/

     divisionEnteraAB = a $divisionEntera b;  /** resultado=2 **/
     residuoAB = a $residuo b;           /** resultado=2 **/

     potenciaAB = $potenciar(a,b);       /** resultado=248832 **/
   }
}
```

Ilustración 10.1. Llamar a Método con Retorno

11. Otras Funcionalidades

Al igual que otros lenguajes de programación, el lenguaje BOOP también cuenta con funcionalidades de apoyo al usuario. A continuación se dará un ejemplo de las principales funciones que posee este lenguaje:

11.1 Incrementar y Decrementar

Se cuenta con las funciones "**$inc**" y "**$dec**" cuyas finalidades son aumentar en 1 y disminuir en 1 el valor de una variable dada como parámetro de entrada.

La Ilustración 4.16 muestra un ejemplo del uso de estas funciones.

```
@principal {
          #ejecutar{
                    $entero a;
                    a=5;
                    $inc(a); /** a = a + 1 , a = 6 **/
                    $dec(a); /** a = a - 1 , a = 5 **/
          }
}
```

Ilustración 11.1. Incrementar y Decrementar

11.2 Métodos de Impresión

Se cuenta con dos funciones, la función "**$impr**" que permite imprimir un texto en consola y la función "**$imprime**" que permite imprimir un texto en consola y realizar inmediatamente un salto de línea.

La Ilustración 4.17 muestra un ejemplo del uso de las funciones de impresión.

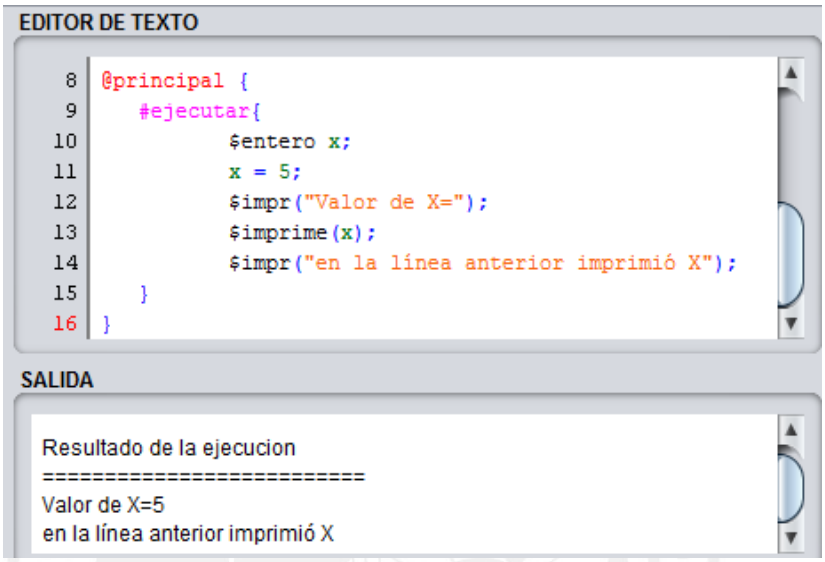

Ilustración 11.2. Métodos de Impresión

11.3 Condicionales

Por medio de los identificadores "**$si**" y "**$sino**" permite condicionar acciones acorde al cumplimiento o no cumplimiento de una sentencia.

La ilustración 4.18 muestra un ejemplo del manejo de condicionales.

```
@principal {
   #ejecutar{
        $entero edad;
        edad = 15;

        $si(edad>=18){
             $imprime("MAYOR DE EDAD");
        }$sino{
             $imprime("MENOR DE EDAD");
        }
    }
}
```

Ilustración 11.3. Condicional SI-SINO

11.4 Iterativas

Con la función "**$mientras**" podremos generar bucles asociados a una condición cualquiera, mientras que la función "**$para**" es mejor aprovechada al realizar bucles asociados al ascenso o descenso de un número.

La Ilustración 4.19 muestra un ejemplo del manejo de la función $mientras.

```
@principal {
    #ejecutar{
        $entero puntaje;
        puntaje = 12;
        $mientras(puntaje<=18){
            $imprime("puntaje bajo");
            $inc(puntaje);
        }
    }
}
```

Ilustración 11.4. Iterativa Mientras

La Ilustración 4.20 muestra un ejemplo del manejo de la función $para.

```
@principal {
    #ejecutar{
        $entero pasos;
        $para(pasos=1; pasos<100;$inc(pasos)){
            /** CÓDIGO **/
        }
    }
}
```

Ilustración 11.5. Iterativa Para