Objective_JS

Objective-JS es un lenguaje de programación orientado a objetos que provee muchas funcionalidades con una manera muy fácil de hacerlo. Si estás familiarizado con Java o con el concepto de programación orientado a objetos, entonces vas a disfrutar mucho este lenguaje.

Tipos de datos

Los tipos de datos que soporta Objective_JS son los siguientes

1. int
2. float
3. char
4. string
5. bool
6. Null

Operadores

Las operaciones que se pueden realizar en Objective_JS son los siguientes:

Operador	**Tipo**
/	Aritmético
*	Aritmético
%	Aritmético
+	Aritmético
-	Aritmétco
==	Lógico
>	Lógico
>=	Lógico
<	Lógico

<=	Lógico
!=	Lógico
and	Lógico
or	Lógico
^	Aritmético

Los operadores están ordenados por prioridad

Caracteres especiales y palabras reservadas

Estos son los caracteres especiales y palabras reservadas que maneja Objective_JS

Reservados	**Uso**
=	Asignación
.	Llamada a funciones
;	Fin de linea
>>	Lectura
read	Llamada a lectura
print	Escritura a consola
function	Palabra para definir una función
returns	Indica que una función tiene un valor de retorno
return	Return va seguido de una expresión que es el valor de retorno de una función
Class	Indicador de que una clase inicia
main	Punto de entrada de la maquina virtual

Declaración de variables

Para declarar una variable en Objective_JS se puede hacer de la siguiente manera

```
var i : int;
var j : float;
var k, l : char;
```

También es posible declarar listas o matrices

```
var foo : list [10] int;
var bar : list [10][15] float;
```

Asignación de variables

Para asignar valores a una variable en Objective_JS se hace de la siguiente manera

```
var i = 5;
foo [2] = 23;
bar [8][13] = 3.1416
```

Estatutos condicionados

Para realizar estatus condicionales se puede hacer de la siguiente manera:

```
If (i > 2) {
    ...
} elsif (i < 5) {
    ...
} elsif (i < 9) {
    ...
}
```

Estatutos ciclados

Para realizar ciclos en Objective_JS proveemos el uso del tradicional while.

```
while (i < 5) {
    …
    I++
}
```

No obstante, introducimos un nuevo estatuto ciclado. Este estatuto está diseñado para repetir estatos de una manera muy específica pero sencilla

```
do n times {
  ...
  }
```

Donde N es un número entero positivo.

Lectura

Para realizar la lectura lo hacemos de la siguiente manera

```
read >> i;
```

De igual manera es posible leer de muchas variables a la vez

```
read >> i >> j;
```

Escritura

También es posible realizar escritura hacia la termina

```
print(“Hola”);
```

De igual manera es posible imprimir muchos estatutos al mismo tiempo

```
print(“Hola”, “Mundo”, i);
```

Funciones

Objective_JS permite implementar funciones dentro de un archivo que contenga en un main o dentro de una clase. Ambos siguen la misma estructura

```
function name(string a, int b, float c, …) returns int {
        ….
        return a;
}
```

La palabra reservada returns es utilizada para indicar que la función regresa algo. Si una función no regresa podemos omitir el uso de returns. Solamente se permite el uso de un return al final de la función.

Es importante notar que en este momento solamente es posible declarar argumentos con los tipos primitivos y listas (Int, Float, Char, String y Boolean) al igual que el valor de retorno de una función solamente pueden ser los primitivos sin listas.

Para declarar una función en una clase es necesario declarar su propósito y seguir exactamente el prototipo a la hora de implementarla

```
Class A {
        A();
        A(string b);
        Attributes {
                public :
                        string c;
                private :
                        String d;
        }
        Methods {
                public :
                        function getC() returns string;
        }
]
A() {
  this.c  = “C”;
}
…
function getC() returns string {
        return this.c;
}
```

Las funciones en clases tienen visibilidad. Nota importante!! Actualmente no es posible llamar funciones de clases dentro de clases.

Constructores

Como se vio en el ejemplo anterior los constructores al igual que las funciones tiene que ser implementados de igual manera a como fueron declarados.

Llamada a funciones

Para llamar a funciones que no pertenecen a una clase basta se puede hacer de la siguiente manera:

```
function sum(int a, int b) returns int {
  return a + b;
]
var c: int;
c = sum(2, 3):
```

Para mandar mandar a llamar a funciones de Objetos se hace de la misma manera, con la única diferencia que antes del nombre de la función este va precedido por el nombre del objeto y un punto.

```
var foo : A;
print(foo.getC());
```

Los constructores en Objective_JS son tratados como función así que para mandar a llamar un lo hacemos de igual manera a una función

```
foo.A(“bar”);
```

Objetos

Los objetos se declaran en un archivo diferente al main. Para utilizar un archivo dentro de una clase main basta simplemente con importarlos.
En Objective_JS estos tienen una estructura muy similar a la de Java. Primero se declaran y luego se implementan.

```
Class A {
        A();
        A(string b);
        Attributes {
                public :
                        string c;
                private :
                        String d;
        }
        Methods {
                public :
                        function getC() returns string;
        }
]
A() {
  this.c  = “C”;
}
…
function getC() returns string {
        return this.c;
}
```

Si no se desea declarar una variables o funciones de tipo público o privado es posible omitir esta sección.
A la hora de implementar los constructores y métodos es importante seguir el orden en el que se declararon junto con el mismo nombre de parámetros.

Herencia

En Objective_JS es posible tener herencia de una sola clase

```
Class B {
        ...
}
Class A Inherits B {
        ...
}
```

Ahora la clase A podrá acceder a todos los métodos y atributos públicos de la clase B.

Es importante recalcar que dentro de tu archivo main tienes que importar primero la clase B y luego la clase A. Si B heredara de C primero importariamos C, luego B y luego A.

Imports

En Objective_JS posible declarar nuestras clases en archivos diferentes y solamente importarlos en el archivo main cuando lo necesitamos.
Esto se hace con la palabra import seguido del nombre de la clase (sin extensión), también es posible hacer varios imports separados por un salto de línea

Estructura de un archivo main

Al igual que Java un archivo que va a ser ejecutado tiene que contar con un main. Este tiene la siguiente estructura

```
import B
import A
Class Foo {
        // Aqui se declaran variables globales
        // También métodos globales
        main() {
                ...
        }
}
```

Extension

Todos los archivo tienen que tener la terminación .Objective_JS

Instalacion

Para instalar el proyecto es necesario clonar el repositorio de Github
$ git clone https://github.com/jvazquez96/Objective-JS.git
Ejecutas el script install.sh que instala todo lo necesario para correr el proyecto.
Una vez instalado nos vamos a la carpeta src ahi se encuentra nuestro compilador y máquina virtual.
Los archivos a ejecutarse tienen que colocarse en la carpeta Tests.
Para ejecutar el programa hay que ejecutar el comando
$ python3 Objective_JS.py *NombreDeTuArchivo.Objective_JS*