Universidad Autónoma de Ciudad Juárez

División Multidisciplinaria Ciudad Universitaria

Cuadro comparativo de lenguajes de Programación Orientada a Objetos

Programación II

Docente: Alan Ponce

Alumno Jose Alejandro Roque Leyva

Licenciatura en Ingeniería de Software

08 de Marzo de 2019

Introducción

El objetivo de este trabajo es presentar su trabajo por lo que se debe incluir:

1.-Definición de lenguajes procedural
Es un paradigma de programación, derivado de la programación estructurada, basada en el concepto de la llamada procedimiento. los procedimientos, también conocidos como rutinas, subrutinas o funciones, simplemente contienen una serie de pasos computacionales a realizar. cualquier procedimiento dado puede llamarse en cualquier momento durante la ejecución de un programa, incluso por otros procedimientos o por si mismo.
Los primeros lenguajes de principales de programación de procedimientos aparecieron por primera vez en 1960. Estos fueron, Fortran, ALGOL, COBOL y BASIC. estos fueron publicados cerca de los años 70.

Uno de los más importantes son:
Pascal:
Este lenguaje de programación fue creado por el profesor suizo niklaus wirth a mediados de los años 1968 y 1969, este fue publicado en 1970. el punto principal de Pascal es crea un lenguaje que facilitara el aprendizaje de programación a sus alumno, que este estaría utilizando la programación estructurada y estructuración de datos.

```
type
tDimension = 1..100;
eMatriz(f,c: tDimension) = array [1..f,1..c] of real;
tRango = record
f,c: tDimension value 1;
end;
tpMatriz = ^eMatriz;
procedure EscribirMatriz(var m: tpMatriz);
var filas,col : integer;
begin
for filas := 1 to m^.f do begin
for col := 1 to m^.c do
write(m^[filas,col]:7:2);
writeln(resultado);
writeln(resultado)
end;
end;
```

Algol:
Algol fue desarrollado a finales de los años 1950 por un comité internacional para crear un lenguaje de programación internacional e independiente de la máquina, aunque no tuvo mucho éxito comercial es muy importante en la historia de la informática, ya que tuvo una

gran influencia en la mayoría de los lenguajes de programación posteriores, por ejemplo, fue el primer lenguaje que introdujo el concepto de variable locales a un bloque de código.

```
BEGIN
 FILE F (KIND=REMOTE);
 EBCDIC ARRAY E [0:12];
 REPLACE E BY "HOLA MUNDO!";
 WHILE TRUE DO
   BEGIN
   WRITE (F, *, E);
   END;
 END.
```

ADA:

ada es un lenguaje de programación orientado a objetos y fuertemente tipado de forma estática que fue diseñado por jean ichbiah de Cii Honeywell bull por encargo del departamento de defensa de los estados unidos.

```
with Ada.Text_IO;

  procedure Hola_Mundo is
  begin
     Ada.Text_IO.Put("¡Hola, Mundo");
  end Hola_Mundo;
```

2.- Definición de lenguaje orientada a objetos

Los lenguajes de programación orientados a objetos tratan a los programas como conjuntos de objetos que se ayudan entre ellos para realizar acciones,. Entendiendo como objeto las entidades que contienen datos. permitiendo que los programas sean más fáciles de escribir, mantener y reutilizar.

Los objetos tiene toda la información (atributos) que lo diferencia de otros pertenecientes a otra clase. por medio de unos métodos se comunican los objetos de una misma o diferente clase produciendo el cambio de estado de los objetos. esto hace que a los objetos se les trate como unidades indivisibles en las que no se separan la información ni los métodos usados en su tratamiento.

Los lenguajes de programación orientados a objetos tienen su origen en un lenguaje que fue diseñado por los profesores ole-Johan Dahl y Kristen Nygaard en Noruega

C++:

El lenguaje de programación c++ fue creado en los años 80 por el profesor Bjarne Stroustrup que esta persona se basó en el lenguaje C. El C++ es un lenguaje orientado a objetos al que se le añadieron características y cualidades de las que carecía el lenguaje C De esta forma nació C++ y como estaba sucediendo con C, depende mucho del hardware, este tiene un gran potencial en la programación de bajo nivel, y también se le agregaron varias herramientas en donde ayuda y permite programar también de alto nivel. El C++ es

uno de los lenguajes de programación más potentes porque como ya dicho, nos permite programar de tanto de alto nivel como de bajo nivel pero a la vez es algo difícil aprenderlo porque es necesario hacerlo casi todo manualmente.
El nombre fue propuesto por Rick mascitti al utilizarse C++ fuera de los laboratorios donde se creó. con el nombre de C++ que quiso dar a entender que es la continuidad de la extensión de el lenguaje de programación C.
ya como se ha mencionado acerca del lenguaje de programación C++, es un lenguaje de programación orientado a objetos, pero no es un lenguaje orientado a objetos puro. El C++ nació como una evolución de C++, y desde que salió de su creación, fue un lenguaje de programación hecho por programadores con un diseño muy práctico al que le fueron añadiendo todos los elementos que se comprobaron eran necesarios sin tener en cuenta aspectos como su imagen , diseño etc.
Todo esto ha estado ocasionando que sus detractores lo usen como argumento de crítica sobre el C++, pero por otra parte precisamente esto es lo que le da mayor valor, al ser un lenguaje de programación más pragmático y sencillo que su antecesor el lenguaje C.

JAVA:
El lenguaje de programación java es un lenguaje de programación orientada a objetos creado por el profesor james gosling en el año 1990. su código era muy similar al del lenguaje C++ y C con un modelo de objetos mucho más sencillo. lo que tenía de diferente entre java y los lenguajes de programación C y C++ es que java es un lenguaje de programación plenamente orientado a objetos.
Java es muy fácil de aprender, prácticamente en java es relativamente sencillo programar desde principio. todos los programadores que ya hayan programado anteriormente con el C o el C++, les costará mucho menos su aprendizaje por la gran similitud entre ellos.
Java supuso un gran avance en los lenguajes de programación, tiene un enorme potencial para el diseño de orientado a objetos con un código, sencillo en un entorno muy estable y agradable. El java nos permite realizar aplicaciones que podemos incluir directamente en páginas web.
Estas aplicaciones se conocen con el nombre de applets. Estos son unos programas que se transfieren dinámicamente a través de internet, los applets tienen un comportamiento inteligente, pueden reaccionar cuando un visitante entra en una página web, y cambian de forma, Todo esto se ha posibilitado que el java sea un lenguaje interactivo entre el usuario y la aplicación.
La mayoría de los lenguajes de programación están compilados en código fuente, mientras que el java es compilado en un bytecode

Python:
El lenguaje de programación Python es un lenguaje de programación moderno, está orientado a objetos, es muy sencillo de usar a la vez potente y de código abierto.
Toda la información está relativamente para el lenguaje(es libre), hay un sitio web en donde se puede encontrar lo relativo con este lenguaje.
El python es un lenguaje de programación que se suele comparar con otros lenguajes como TCL, Perl. Scheme, java, o Ruby. Este lenguaje fue creado por guido van rossum basándose en otro lenguaje de programación, El abc. El nombre proviene de los humoristas británicos Monty python que tanto le gustaban a guido van rossum.

Este mismo es un lenguaje de programación de scripting. Los lenguajes scripting son aquellos lenguajes que usan un intérprete en vez de ser compilados. es opuesto al perl, lenguaje con el que se rivaliza amistosamente. la mayoria de usuarios del python lo consideran como un lenguaje más limpio y elegante a la hora de programar.
Python nos permite separar el programa en módulos, este lenguaje tiene una gran variedad de módulos estándar que se pueden utilizar para programar o incluso como una base para aprender a programar en python
El lenguaje está interpretando el ahorro muchísimo tiempo en la creación de programas puesto que no es preciso compilar código. El intérprete que usa el python de modo interactivo lo que nos permite experimentar con este lenguaje mientras se está programando.
Este lenguaje es un lenguaje de programación que permite que podamos programar en varios estilos: programación orientada a objetos, programación estructurada, programación funcional, y programación orientada a aspectos. A esto se le conoce como lenguaje de programación multiparadigma

C#:
El lenguaje de programación fue creado por el danés anders hejlsberg que diseño también los lenguajes turbo pascal y delphi. el C# (con pronunciación de c sostenido) es un lenguaje de programacin orientada a objetos. con este nuevo lenguaje se quiso mejorar con respecto de los 2 lenguajes anteriores de los que deriva el c y el c++.
Con el c# se pretende que incorporarse las ventajas o mejoras que tiene el lenguaje java, así se consiguió que tuviese las ventajas del c, del c++ pero además la productividad que posee el lenguaje java y se le denomino c#.
algunas de las características del lenguaje de programación c# son: Su código se puede tratar íntegramente como un objeto. sus sintaxis es muy similar ala del java. es un lenguaje orientado a objetos y a componentes. armoniza la productividad visual basic con el poder y flexibilidad del c++, ahorramos tiempo en la programación ya que tiene una librería de clases muy completa y bien diseñada.
A pesar que el lenguaje C# forma parte de la plataforma .NET que es una interfaz de programación de aplicaciones. C# es un lenguaje independiente que originalmente se creó para producir programas sobre esta plataforma .NET.
El visual basic no tiene algunas de las características necesarias como la herencia, los métodos virtuales, la sobrecarga de operadores, etc. que han conseguido con el C# y la plataforma .NET.

3.- Cuadro comparativo de estos dos paradigmas

Paradigma Procedural	Paradigma Orientada a Objetos
● Diseño de arriba hacia abajo ● Reutilizacion de codigo limitada ● Código complejo ● Utiliza diferentes métodos en todo el código que la que la programación	● Diseño enfocado a objetos ● Reutilizacion de codigo eficiente ● Diseño complejo ● utiliza objetos ● utiliza clases en las que la

orientada a objetos ● Utiliza registros	programación de procedimientos utiliza llamadas a procedimientos.

Lenguajes de Programación Orientada a Objetos

En esta seccion se debera focalizar en las diferencias entre los diversos lenguajes de programacion que soportan el paradigma de la programacion orientada a objetos.

C++

Incluir codigo de como implementar:

Una clase:

```
class Libro {
  public: Libro();
  void establecerTitulo(char*);
  void establecerPaginas(int);
  void establecerCodigo(int);
  void imprime();
  private:
  char Titulo[40];
  int Cantpaginas;
  int Codigo;
}; //Importante notar el ; despues de cerrar la llave
```

Herencia:

```
#include <iostream>

using namespace std;

class Animal {
  private:
     int age;
  public:
     Animal(int a = 1){
        age = a;
     };
     int getAge() const {return age;}
     void setAge(int a = 9){
        age = a;
     }
};

class Dog:public Animal { };
```

```
class Cat:public Animal {
  private:
     float weight;
  public:
     Cat(int a = 2, float w = 3.2):Animal(a){
        weight = w;
     }
     float getWeight() const {return weight;}
};

int main(){
    Dog mastin;
    cout << "____DOG____" << endl;
    cout << "Before: " << mastin.getAge() << endl;
    mastin.setAge(2);
    cout << "After: " << mastin.getAge() << endl;
    Cat miau(3, 4.1);
    cout << "____CAT____" << endl;
    cout << miau.getAge() << " " << miau.getWeight() << endl;
}
```

Poliformismo:

```
#include <iostream.h>
#include <math.h>
#include <conio.h>

// clase Shape

class Shape {
      public:
          virtual double area() const { return 0.0; }
          virtual double volume() const { return 0.0; }

          // funcion virtual pura sobrepuesta en las clases derivadas
          virtual void print() const=0;
};
```

Encapsulamiento:

```
class encap {

public:
   int  A;        // Propiedades
   int  B;
   TColor     Color;
   TPaintBox *PaintBox;

   void Mostrar (void); // Métodos
```

};

Java

Incluir codigo de como implementar:

Una clase:

```
/* Un programa que usa la clase Vehiculo
 El archivo se llama DemoVehiculo.java
 */

class Vehiculo {
    int pasajeros;  //números de pasajeros
    int capacidad;  //capacidad del combustible en galones
    int mpg;        //combustible consumido en millas por galon
}

//Esta clase declara un objeto de tipo Vehiculo

class DemoVehiculo {

    public static void main(String[] args) {
        Vehiculo minivan = new Vehiculo();
        int rango;

        //asignando valores a los campos de minivan
        minivan.pasajeros = 9;
        minivan.capacidad = 15;
        minivan.mpg = 20;

        //Calcular el rango asumiendo un tanque lleno
        rango = minivan.capacidad * minivan.mpg;

        System.out.println("La Minivan puede llevar " +
minivan.pasajeros + " pasajeros con un rango de " + rango + " millas");
    }
}
```

Herencia:

```
//Clase para objetos de dos dimensiones
class DosDimensiones{
    double base;
    double altura;
    void mostrarDimension(){
        System.out.println("La base y altura es: "+base+" y"+altura);
    }
}

//Una subclase de DosDimensiones para Triangulo
```

```
class Triangulo extends DosDimensiones{
    String estilo;
    double area(){
        return base*altura/2;
    }
    void mostrarEstilo(){
        System.out.println("Triangulo es: "+estilo);
    }
}
class Lados3{
    public static void main(String[] args) {
        Triangulo t1=new Triangulo();
        Triangulo t2=new Triangulo();
        t1.base=4.0;
        t1.altura=4.0;
        t1.estilo="Estilo 1";
        t2.base=8.0;
        t2.altura=12.0;
        t2.estilo="Estilo 2";
        System.out.println("Información para T1: ");
        t1.mostrarEstilo();
        t1.mostrarDimension();
        System.out.println("Su área es: "+t1.area());
        System.out.println();
        System.out.println("Información para T2: ");
        t2.mostrarEstilo();
        t2.mostrarDimension();
        System.out.println("Su área es: "+t2.area());
    }
}
```

Poliformismo:

```
public class Polimorfismo {

    public static void main(String[] args) {

        // Creamos una variable del tipo MiClaseA, que sera un array de 3 elementos

        MiClaseA [] misClases=new MiClaseA[3];
```

```
                misClases[0]=new MiClaseA("Esther");
                // Asignamos a la variable misClases que son del tipo MiClaseA un objeto del
                // tipo MiClaseB, ya que hereda de MiClaseA
                misClases[1]=new MiClaseB("Juan", "Azul");
                misClases[2]=new MiClaseA("Rosa");
                for(MiClaseA e:misClases) {
                        // ejecutara la función info() de la clase que haya sido instanciada.
                        // Esto se llama polimorfismo
                        System.out.println(e.info());
                }
        }
}
class MiClaseA {
        private String name;
        // constructor
        public MiClaseA(String name) {
                this.name=name;
        }
        public String info() {
                // Devolvemos el nombre
                return this.name;
        }
}
class MiClaseB extends MiClaseA {
        private String color;
        // constructor
        public MiClaseB(String name, String color) {
                // ejecutamos el constructor de la superclase (MiClaseA) enviandole el nombre
                super(name);
                // guardamos el nombre del color en la variable color de MiClaseB
                this.color=color;
        }
        public String info() {
                // devolvemos el contenido de MiClaseA.info() mas la variable color
                // de la clase MiClaseB
                return super.info() + " (" + this.color + ")";
        }
}
```

Encapsulamiento:

```
public class people{
  prívate int altura;   //Atributos
   //Métodos
        public int getAltura(){
        return this.Altura;
        }

       public void setAltura(int unaAltura){
        this.altura = unaAltura;
          }
        }
```

Python

Incluir codigo de como implementar:

Una clase:

```
def prueba_ambitos():
    def hacer_local():
        algo = "algo local"
    def hacer_nonlocal():
        nonlocal algo
        algo = "algo no local"
    def hacer_global():
        global algo
        algo = "algo global"
    algo = "algo de prueba"
    hacer_local()
    print("Luego de la asignación local:", algo)
    hacer_nonlocal()
    print("Luego de la asignación no local:", algo)
    hacer_global()
    print("Luego de la asignación global:", algo)

prueba_ambitos()
print("In global scope:", algo)
```

Herencia:

```
#!/usr/bin/python
# Nombre de Fichero : polimorfismo.py
class Gato:
   "CIsase Gato"
   def __init__(self):
      self.pos = 0
   def trepar(self):
      pass
   def caminar(self):
```

```
        self.pos = self.pos + 10

    def maullar(self):
        print "Miu Miu"

class Perro:
    "Clase Perro"
    def __init__(self):
        self.pos = 0
    def caminar(self):
        self.pos = self.pos +4
    def ladrar(self):
        print "Gua Gua"

def pasearMascota(mascota):
    for i in range(5):
        mascota.caminar()
    print "La mascota quedo en " + str(mascota.pos)

perro=Perro()
gato=Gato()
pasearMascota(perro)
pasearMascota(gato)
```

Poliformismo:

```
#!/usr/bin/python
# Nombre de Fichero : polimorfismo.py
class Gato:
    "Clsase Gato"
    def __init__(self):
        self.pos = 0
    def trepar(self):
        pass
    def caminar(self):
        self.pos = self.pos + 10

    def maullar(self):
        print "Miu Miu"

class Perro:
    "Clase Perro"
    def __init__(self):
        self.pos = 0
    def caminar(self):
        self.pos = self.pos +4
    def ladrar(self):
        print "Gua Gua"

def pasearMascota(mascota):
    for i in range(5):
        mascota.caminar()
    print "La mascota quedo en " + str(mascota.pos)
```

```
perro=Perro()
gato=Gato()
pasearMascota(perro)
pasearMascota(gato)
```

Encapsulamiento:

```
#!/usr/bin/env python
# Nombre de Fichero : encapsulacion.py

class Persona(object) :
   "Calse Persona"
          def __init__(self, pNombre,pEdad,pSueldo) :
          self.setNombre(pNombre)
          self.setEdad(pEdad)
       self.__setSueldo(pSueldo);

   def setEdad(self, pEdad) :
      self.__edad = pEdad

   def getEdad(self) :
      return self.__edad

          def setNombre(self, pNombre) :
          self.__nombre = pNombre

          def getNombre(self) :
       return self.__nombre

          def __setSueldo(self,pSueldo):
          self.__sueldo = pSueldo
          def getSueldo(self):
       return self.__sueldo

          nombre = property(getNombre, setNombre)
          edad = property(getEdad, setEdad)

class Gerente(Persona) :
          def __init__(self, pNombre,pEdad):
          Persona.__init__(self, pNombre,pEdad,5000)

class Secretaria(Persona) :
          def __init__(self, pNombre,pEdad) :
          Persona.__init__(self, pNombre,pEdad,500)

if __name__ == '__main__':
```

```
g = Gerente("Mariano", 56)
s  = Secretaria("Rocio", 33)

print "El Gerente es", g.nombre , " gana ", g.getSueldo()
print "La Secretaria es ", s.getNombre(), " gana ",s.getSueldo
```

C#

Incluir codigo de como implementar:

Una clase:

```
//Declaring an object of type MyClass.
MyClass mc = new MyClass();

//Declaring another object of the same type, assigning it the value of the first object.
MyClass mc2 = mc;
```

Herencia:

```
using System;

public class A
{
  private int value = 10;

  public class B : A
  {
        public int GetValue()
        {
        return this.value;
        }
  }
}

public class C : A
{
//      public int GetValue()
//      {
//      return this.value;
//      }
}

public class Example
{
        public static void Main(string[] args)
        {
```

```
        var b = new A.B();
        Console.WriteLine(b.GetValue());
        }
}
```

Poliformismo:

```
Persona[] personas = new Persona[2];
personas[0] = new Alumno("Jaime Correa");
personas[1] = new Empleado("Juan eduardo");
for (int i = 0; i < personas.Length; i++)
{
    Console.WriteLine(personas[i].Saludar());
}
```

Encapsulamiento:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace FrancysSytem_Encapsulamiento
{
    class Program
    {
        class Salon
        {
            private string nombre = "Omar";

            public string Nombre
            {
                get { return nombre; }
                set { nombre = value; }
            }
            public string nombres = "Carlos";


        }
        class MainClass
        {
            public static void Main()
            {
                Salon nuevo_salon = new Salon();
                Console.WriteLine("Encapsulación  en c#");
```

```
                    Console.WriteLine("Nombre uno : {0}",
nuevo_salon.nombres); ///Variable Publica
                    Console.WriteLine("Nombre dos : {0}",
nuevo_salon.Nombre); ///Encapsulación
                    Console.WriteLine("\n\nfrancysystem.blogspot.com");
                    Console.ReadLine();

                }
            }
        }
    }
```

Conclusiones

Lo que da a entender la programación orientada a objetos es que es un paradigma que utiliza objetos como elementos fundamentales en la construcción de una solución, todo esto inició a principios de los años 70 y todas las propiedades y métodos comunes a los objetos se encapsulan o se agrupan en clases.
y de lo que es la programación procedural es que el procedural describe paso a paso un conjunto de instrucciones que deben ejecutarse para variar el estado del programa y hallar la solución, es decir un algoritmo en el que se describen los pasos necesarios para solucionar el problema

Referencias.

- Revistainformatica. (2005). [programacion orientada a objetos]. Recuperado 8 marzo, 2019, de http://www.larevistainformatica.com/lenguajes-programacion-orientada-objetos.htm
- Revistainformatica. (2005b). [lenguaje de programacion c++]. Recuperado 8 marzo, 2019, de http://www.larevistainformatica.com/C++.htm
- La revista informatica. (2006). Lenguaje de programación java. Recuperado 8 marzo, 2019, de http://www.larevistainformatica.com/Java.htm
- La revista informatica. (2006b). Lenguaje de programación Python. Recuperado 8 marzo, 2019, de www.larevistainformatica.com/Python.htm
- La revista informatica. (2006c). Lenguaje de programación C#. Recuperado 8 marzo, 2019, de www.larevistainformatica.com/Python.htm
- Wikipedia. (2019, 16 enero). procedural_programming. Recuperado 8 marzo, 2019, de en.wikipedia.org/wiki/Procedural_programming
- Wikipedia. (2019b, 16 enero). Anexo:Ejemplos de implementación del «Hola mundo». Recuperado 8 marzo, 2019, de https://es.wikipedia.org/wiki/Anexo:Ejemplos_de_implementaci%C3%B3n_del_%C2%ABHola_mundo%C2%BB#En_ALGOL