## Universidad Autónoma de Ciudad Juárez

## División Multidisciplinaria Ciudad Universitaria

## 

## Cuadro comparativo de lenguajes de Programación Orientada a Objetos

## Programación II

## Docente: Alan Ponce

## 

## 

## George Iván Rodríguez Gómes 16000

## Licenciatura en Ingeniería de Software

## 08 de Marzo de 2019

# Introduccion

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

1.-Definicion de lenguajes procedural

Programación imperativa o procedural, tiene como característica principal en sus programas poseer secuencias de instrucciones que deben llevarse a cabo como una receta o guión para resolver un problema determinado(Galindo y Pastó, 2015). Es importante aclarar que estos pueden trabajar con procedimientos y funciones, pero no debe confundirse con el paradigma funcional, en el cual todo se describe con una función(Juganaru Mathieu, 2014).

Indicar lenguajes de programación y ejemplos de código

Algunos ejemplos de lenguajes de programación procedurales o imperativos son: lenguaje máquina, lenguaje ensamblador, C, Fortran, Cobol, Pascal, Ada, C++, C#, Java. Fortran, Cobol, Pascal, Ada, C++, C#, Java. A excepción del lenguaje máquina y el lenguaje ensamblador, los demás son lenguajes estructurados. Un lenguaje de programación puede verificar uno o más paradigmas. Por ejemplo, el lenguaje Java comprueba el paradigma orientado a objetos y el código que compone la parte de métodos de los objetos verifica el paradigma estructurado(Juganaru Mathieu, 2014).

Ejemplo de un paradigma funcional y orientado a objetos en java(Hoyos y Puertas, 2017):

El método mostrado en el código a) calcula la solución para una solución en particular, así el método debe ser aplicado en forma recursiva a cada una de las entradas, mediante la llamada al método invoke que recibe como parámetro el objeto task, representativo de una tarea recursiva, que se traduce en un conjunto de tareas individuales generadas para cada uno de las entradas contenidas en el objeto juegos (arreglo de String), tal como se muestra en el código b) . En este caso, es el método compute de la clase Task, el encargado de crear las tareas individuales para cada una de las entradas, tal como se muestra en el código c).

a)

public Grid solve(String cadena){

Grid grid = parseGrid(cadena);

return search(grid);

}

b)

Task task = new Task(juegos);

pool.invoke(task);

c)

protected void compute() {

Grid all = (new Sudoku()).solve(juego);

if(enlistados<juegos.length){

Task task2 = new Task(juegos);

invokeAll(task2);

}

}

2.- Definición de lenguaje orientada a objetos

Existen tres principios fundamentales que gobiernan este tipo de programación:

Encapsulación: en este principio se encapsulan datos, estados, operaciones y, en ocasiones, también eventos, en objetos. El código sería ejecutado, entonces, según la ocurrencia de eventos o de creación/destrucción de instancia de objetos.

Prototipos, clases y herencias: El prototipo y la clase son las abstracciones del objeto; otros prototipos se definen de acuerdo con un prototipo existente.

Tipificación y polimorfismo: constituyen la comprobación del tipo con respecto a la jerarquía de las clases.

Indicar lenguajes de programacion y una resena de creacion y evolucion (C++, Java, Python, C#)

3.- Cuadro comparativo de estos dos paradigmas

|  |  |
| --- | --- |
| Paradigma Procedural | Paradigma Orientada a Objetos |
|  |  |

# 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

Herencia

Polifirmismo

Encapsulamiento

### Java

Incluir codigo de como implementar:

Una clase

Herencia

Polifirmismo

Encapsulamiento

### Python

Incluir codigo de como implementar:

Una clase

Herencia

Polifirmismo

Encapsulamiento

### C#

Incluir codigo de como implementar:

Una clase

Herencia

Polifirmismo

Encapsulamiento

# Conclusiones

En esta seccion debera plantear sus conclusiones de acuerdo a su analis comparitivo

# Bibliografía

Galindo, J. M., y Pastó, V. J. (2015). *Introducción a la programación*. Cataluña.

Hoyos, J. G., y Puertas, A. (2017). Desempeño de los paradigmas funcional y orientado a objetos en arquitecturas multicore: Caso de estudio haskell-Java. *Informacion Tecnologica*, *28*(5), 65-74. http://doi.org/10.4067/S0718-07642017000500009

Juganaru Mathieu, M. (2014). *Introducción a la programación*. *Universitat Jaume I*. Mexico, D.F.: Grupo Editorial Patria.