USO DEL PARADIGMA DE ORIENTACIÓN A ASPECTOS EN EL JUEGO BRICK BREAK

ALUMNOS:

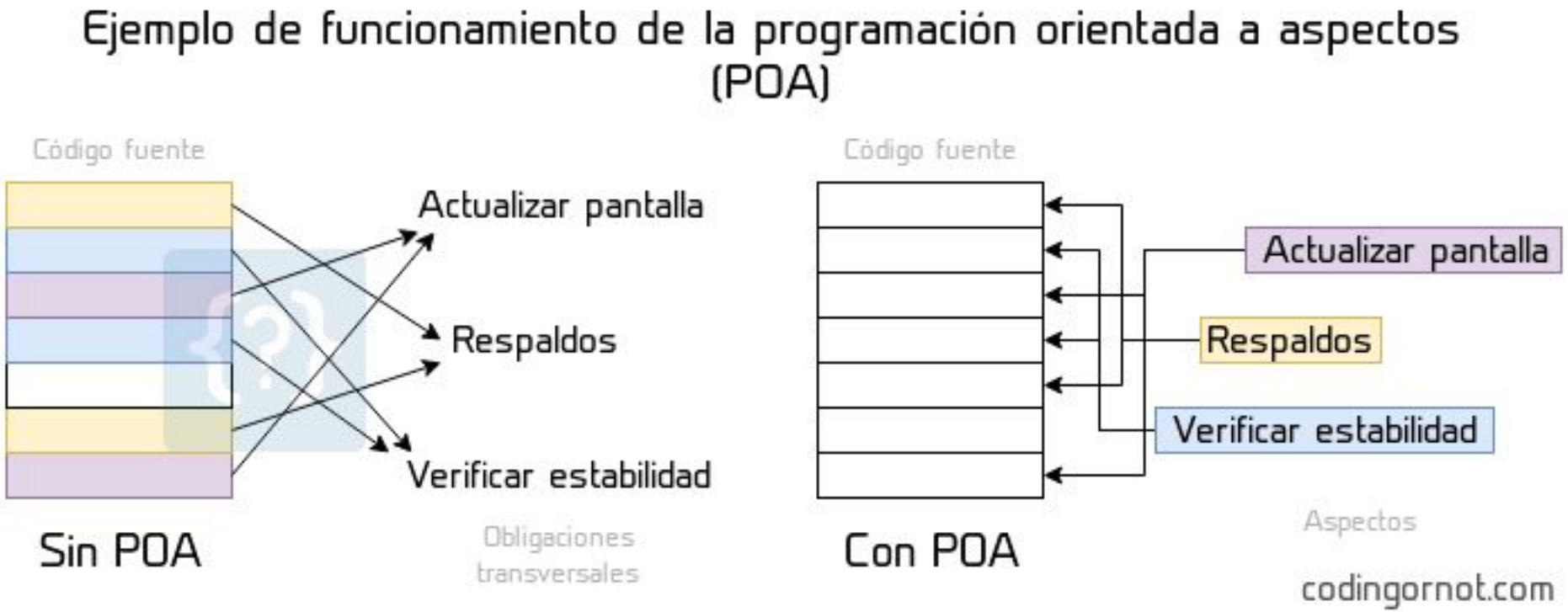
EDSON LÁZARO
CARLOS ESPINOZA
VÍCTOR PONCE
JOSIAS RUEGG

PROGRAMACIÓN ORIENTADA A ASPECTOS

La programación orientada a aspectos (POA) es un paradigma de programación que basa su filosofía en tratar las obligaciones transversales de nuestros programas como módulos separados (aspectos) para lograr una correcta separación de responsabilidades.

Una obligación transversal es aquella que se repite en varias partes de un programa independientemente de si las secciones en las que aparece tienen relación directa.

Ejemplo:


Ejemplo de funcionamiento de la programación orientada a aspectos (POA)
Código fuente
Actualizar pantalla
Respaldos
Verificar estabilidad
Sin POA
Obligaciones transversales
Código fuente
Actualizar pantalla
Respaldos
Verificar estabilidad
Con POA
Aspectos
codingornot.com

Conceptos de POA

Los siguientes 3 conceptos son los más importantes de la programación orientada a aspectos general:

Aspecto (aspect): funcionalidad transversal (se repetirá a lo largo del sistema) que será implementada de forma separada. Es el concepto principal de este paradigma puesto que representa la sección de código que se separó del resto del programa.

Punto de corte (pointcut): es el que se encarga de especificar mediante expresiones regulares (regex) en qué parte del programa se debe de insertar un aspecto.

Consejo (advice): es el código que ejecutará el aspecto (cuerpo del algoritmo).

Herramientas para la implementación de POA:

AspectC++: para C++

AspectJ: extensión de Eclipse para Java

Aspect: módulo para Perl

Spring: para Java

Aspectlib: para Python

AspectGo: framework en desarrollo para Golang

Otra manera de implementarlo usando python...

DECORADORES:

Los decoradores en Python son funciones que toman como argumento a otra función con el objetivo de agregarle funcionalidades extra a la misma.

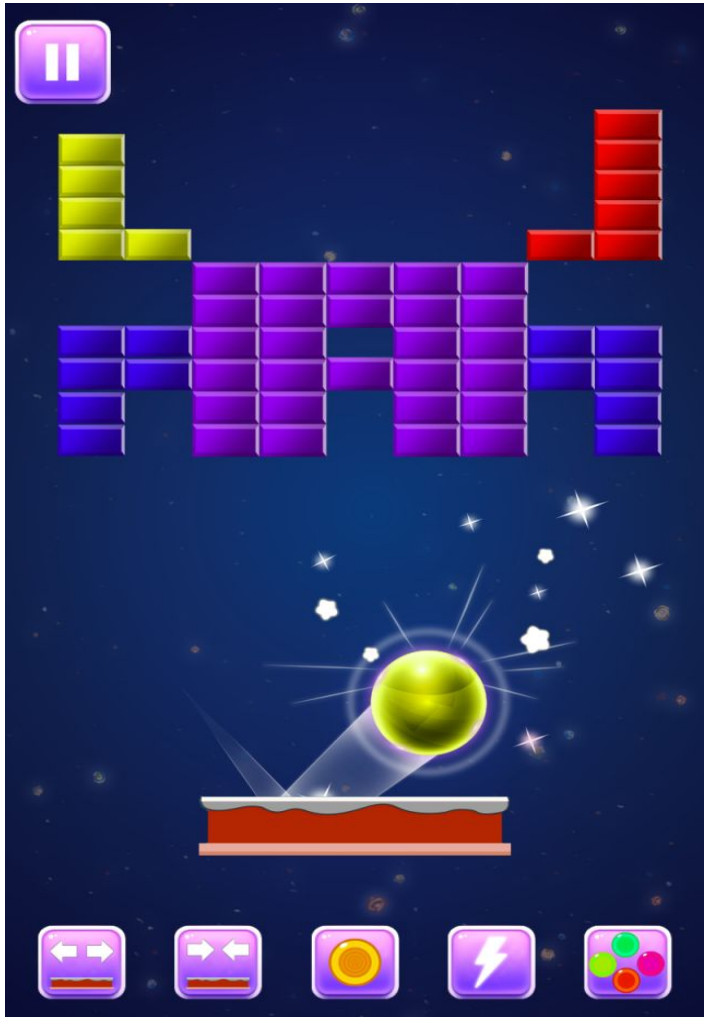
Ejemplo de decorador:

```python
def decorator_function(original_function):
    def wrapper_function(*args,**kwargs):
        print(f"Another function was here: {original_function.__name__}")
        return original_function(*args,**kwargs)

    return wrapper_function

@decorator_function
def display():
    print('display function ran')

@decorator_function
def display_info(name,age):
    print(f'display_info ran with arguments {name} and {age}')


#decorated_display = decorator_function(display)
#decorated_display()

display()
display_info('Nombre',55)
```

Game Brick Break

Objetivo:

Eliminar todos los bloques sin dejar caer la bolita.



Aplicación de AOP:

En el juego se tienen diferentes materiales de bloques, y diferentes efectos al momento de eliminar estos tales como el sonido y explosión.

AOP se aplica a estos efectos.

Paradigma POO

Objetos Utilizados:

- Pelota
- Paleta
- Ladrillo

Funciones generales:

- Crear set de ladrillos
- Crear matriz de posiciones
- Colisión set de ladrillos