Reparación de métodos

Gustavo Jose Hernandez Sotres

Inline

Codigo original:

```python
@property
def estado_de_animo(self):
    ''' Regresa el estado de ánimo de nuestro zombie basado en el tiempo
        que ha pasado sin comer '''
    if self.mas_de_10_segundos_sin_comer():
        # Aplicar inline variable
        estado = 'agresivo'
        return estado
    elif self.mas_de_20_segundos_sin_comer():
        return 'enfadado'
    elif self.mas_de_5_segundos_sin_comer():
        return 'normal'
    else:
        return 'contento'

# Aplicar inline method
def mas_de_10_segundos_sin_comer(self) -> bool:
    return datetime.now() - self.__ultimo_alimento > timedelta(seconds=10)

def mas_de_20_segundos_sin_comer(self) -> bool:
    return datetime.now() - self.__ultimo_alimento > timedelta(seconds=20)

def mas_de_5_segundos_sin_comer(self) -> bool:
    return datetime.now() - self.__ultimo_alimento > timedelta(seconds=5)
```

Codigo modificado:

```python
@property
def estado_de_animo(self):
    ''' Regresa el estado de ánimo de nuestro zombie basado en el tiempo
        que ha pasado sin comer '''
    if datetime.now() - self.__ultimo_alimento > timedelta(seconds=self.AGRESIVO):
        estado = 'agresivo'
        return estado
    elif datetime.now() - self.__ultimo_alimento > timedelta(seconds=self.ENFADADO):
        return 'enfadado'
    elif datetime.now() - self.__ultimo_alimento > timedelta(seconds=self.NORMAL):
        return 'normal'
    else:
        return 'contento'
```

En este caso movimos la comparación dentro del mismo if se entiende un poco mejor que hace la función y es mas claro que que valor es para Agresivo, enfadado y normal.

Extract variable

Original:

```
# Aplicar extract variable
@property
def peligrosidad(self):
    ''' Regresa el nivel de peligrosidad del zombie en tres niveles
        alta, media o baja '''
    if not (self.__tipo == TipoZombie.ZOMBIEINSTEIN or self.__tipo == TipoZombie.ZOMBIEPULTA) and (self.estado_de_animo == 'contento' or self.e
        return 'baja'
    elif not (self.__tipo == TipoZombie.ZOMBIEINSTEIN or self.__tipo == TipoZombie.ZOMBIEPULTA) and not (self.estado_de_animo == 'contento' or
        return 'media'
    else:
        return 'alta'
```

Nuevo:

```
@property
def peligrosidad(self):
    ''' Regresa el nivel de peligrosidad del zombie en tres niveles
        alta, media o baja '''

    zombie_feliz = (self.__tipo == TipoZombie.ZOMBIEINSTEIN or self.__tipo == TipoZombie.ZOMBIEPULTA) and (self
    zombie_buena_onda = not (self.__tipo == TipoZombie.ZOMBIEINSTEIN or self.__tipo == TipoZombie.ZOMBIEPULTA)

    if not zombie_feliz:
        return 'baja'
    elif not zombie_buena_onda:
        return 'media'
    else:
        return 'alta'
```

En este caso solo extrajimos la condición y le pusimos un nombre mas adecuado así ayuda a leer todo mejor

Replace Temp with Query

Original:

```python
# Aplicar replace temp with query
def ir(self, lugar, tiempo_del_recorrido: timedelta):
    ''' Regresa un mensaje con los detalles del recorrido, además de un valor
        boolean que indica si el zombie debería de ir a tal lugar basado en
        el tiempo de vida que le queda '''
    if self.tiempo_de_vida_total - (datetime.now() - self.__timestamp) >= tiempo_del_recorrido: # usar tiempo_de_vida_restante
        return f'Adelante, puede ir al {lugar}', True
    else:
        return f'No se recomienda ir al {lugar}. Desaparecerá antes', False
```

Nuevo:

```python
# Aplicar replace temp with query
def ir(self, lugar, tiempo_del_recorrido: timedelta):
    ''' Regresa un mensaje con los detalles del recorrido, además de un valor
        boolean que indica si el zombie debería de ir a tal lugar basado en
        el tiempo de vida que le queda '''
    if self.tiempo_de_vida_restante >= tiempo_del_recorrido: # usar tiempo_de_vida_restante
        return f'Adelante, puede ir al {lugar}', True
    else:
        return f'No se recomienda ir al {lugar}. Desaparecerá antes', False
```

En este caso solo usamos la @property de tiempo_de_vida_restante que lo hace mas legible y nos permite reutilizarlo.

Split Temporary Variable

Original:

```
def buscar_cerebros(self, radio: int):
    ''' Regresa los detalles asociados a la búsqueda de cerebros. El
        radio está en kilómetros'''
    return f'''Detalles de la búsqueda
    perímetro: {2 * pi * radio} km
    area     : {pi * pow(radio, 2)} km
    '''
```

Nuevo:

```
def buscar_cerebros(self, radio: int):
    ''' Regresa los detalles asociados a la búsqueda de cerebros. El
        radio está en kilómetros'''
    perimetro = 2 * pi * radio
    area = pi * pow(radio, 2)
    return f'''Detalles de la búsqueda
    perímetro: {perimetro} km
    area     : {area} km
    '''
```