Clean Code Jacob Davis

Bloque 1:

```python
#region NombresConSentido

#MAL: el nombre de la funcion no tiene sentido, y los nombres de las variables no indican lo que son
        def funcionNumeros():
            i = 2
            b = 3
            return i + b

#BIEN: El nombre de la funcion deja claro lo que hace, y los nombres de las variables tienen sentido.
        def sumaNumeros():
            numeroUno = 2
            numeroDos = 3
            return numeroUno + numeroDos

#endregion NombresConSentido
```

```python
#region NombresPronunciables

#MAL: Los nombres de las cosas no se pueden pronunciar bien
        def nmbrsPrncbls():
            cdna = 'Hola que tal'
            return cdna

#BIEN: Los nombres son claros y se pueden pronunciar
        def nombresPronunciables():
            cadena = 'Hola que tal'
            return cadena

#endregion NombresPronunciables
```

```python
#region NombresBuscables

#MAL: El maximo esta puesto como 10 en vez de ser una constante
        def nombresNoBuscables():
            numeroElegido = input('Elige un numero')
            if (numeroElegido < 10):
                print('Tu numero es menor a 10')
            else:
                print('Tu numero es mayor que 10')


#BIEN: He extraido NUM_MAX a una constante en vez de poner 10 directamente
        NUM_MAX = 10
        def nombresBuscables():
            numeroElegido = input('Elige un numero')
            if (numeroElegido < NUM_MAX):
                print('Tu numero es menor a 10')
            else:
                print('Tu numero es mayor que 10')

#endregion NombresBuscables
```

```
#region Nombres de Clases

#MAL: Este ejemplo esta fatal. El nombre de la funcion no tiene verbo ni nada, y los nombres de las variables son horribles con un verbo.
        def numeroMasNumero():
            introduceNumero1 = input('Introduce el primer Numero')
            introduceNumero2 = input('Introduce el segundo Numero')

            return introduceNumero1 + introduceNumero2

#BIEN: En este ejemplo el nombre de la funcion es una accion, y las variables tienen nombres normales y claras
        def sumarNumeros():
            numero1 = input('Introduce el primer Numero')
            numero2 = input('Introduce el segundo Numero')

            return numero1 + numero2


#endregion Nombres de Clases
```

```
#MAL: Estas dos funciones hacen algo muy similar pero uno esta usando de nombre la palabra calculadora y la otra operador
        def calculadoraSuma():
            numero1 = input('Introduce el primer Numero')
            numero2 = input('Introduce el segundo Numero')

            return numero1 + numero2

        def operadorResta():
            numero1 = input('Introduce el primer Numero')
            numero2 = input('Introduce el segundo Numero')

            return numero1 - numero2

#BIEN: Como las dos funciones hacen casi lo mismo lo unico que cambia es que uno suma y el otro resta, usamos la palabra calculadora para los dos

        def calculadoraSuma():
            numero1 = input('Introduce el primer Numero')
            numero2 = input('Introduce el segundo Numero')

            return numero1 + numero2

        def calculadoraResta():
            numero1 = input('Introduce el primer Numero')
            numero2 = input('Introduce el segundo Numero')

            return numero1 - numero2
```

Bloque2:

```
#region Funciones Cortas y Funciones que solo hacen una cosa

#MAL: Estamos metiendo las cuatro operaciones dentro de una funcion. Esto es malo porque hace que la funcion sea mas larga,
# y que la funcion siempre haga varias cosas

    def operacionesConDosNumeros(numero1, numero2):
        suma = numero1 + numero2
        resta = numero1 - numero2
        multiplicacion = numero1 * numero2
        division = numero1 / numero2

        print(f"La suma es: {suma}")
        print(f"La resta es: {resta}")
        print(f"La multiplicación es: {multiplicacion}")
        print(f"La división es: {division}")

#BIEN: Dividimos las operaciones en funciones separadas. Cada funcion es corta, y solo hace una cosa
    def suma(numero1, numero2):
        return numero1 + numero2

    def resta(numero1, numero2):
        return numero1 - numero2

    def multiplicacion(numero1, numero2):
        return numero1 * numero2

    def division(numero1, numero2):
        if numero2 != 0:
            return numero1 / numero2
        else:
            return None

    def mostrarResultado(operacion, resultado):
        print(f"El resultado de {operacion} es: {resultado}")

    def operaciones_con_dos_numeros(numero1, numero2):
        resultadoSuma = suma(numero1, numero2)
        resultadoResta = resta(numero1, numero2)
        resultadoMultiplicacion = multiplicacion(numero1, numero2)
        resultadoDivision = division(numero1, numero2)

        mostrarResultado("suma", resultadoSuma)
        mostrarResultado("resta", resultadoResta)
        mostrarResultado("multiplicación", resultadoMultiplicacion)
        mostrarResultado("división", resultadoDivision)
```

```
#region No abusar de Switch Case

#MAL: Estamos usando un match case (lo mismo que un switch). Se podria decir que no estamos abusando de ello,
# pero abajo hay un ejemlo de otra manera que se puede conseguir lo mismo, que cumple más con las normas
        def determinar_tipo_de_fruta(fruta):
            match fruta:
                case "manzana":
                    return "Fruta común"
                case "plátano":
                    return "Fruta alargada"
                case "naranja":
                    return "Cítrico"
                case "sandía":
                    return "Fruta grande"
                case "uva":
                    return "Fruta en racimo"
                case _:
                    return "Tipo de fruta desconocido"

#BIEN: Aqui en vez de usar un switch, estamos usando un diccionario para mappear la descripcion a cada fruta.
# De esta manera puedes buscar una fruta, y te devolvera su descripcion

        def determinarTipoDeFruta(fruta):
            tiposDeFruta = {
                "manzana": "Fruta común",
                "plátano": "Fruta alargada",
                "naranja": "Cítrico",
                "sandía": "Fruta grande",
                "uva": "Fruta en racimo"
            }

            return tiposDeFruta.get(fruta, "Tipo de fruta desconocido")
```

```
#region Pocos parametros


#MAL: Esta funcion toma muchos parametros y luego los muestra todos.

        def describirPersona(altura, edad, nombre, colorPelo, sexo, peso):
            print('La persona mide ' + altura + 'cm, tiene ' + edad + ' años, se llama ' + nombre + ' su pelo es de color ' + colorPelo +


#BIEN: Aqui creamos diccionarios de personas(Se podrian crear tambien clases), y la funcion solo pide una de las personas, y nos muestra s

        pablo = {
            "altura" = 180
            "edad" = 20
            "nombre" = pablo
            "colorPelo" = rubio
            "sexo" = Hombre
            "peso" = 70
        }

        maria = {
            "altura" = 165
            "edad" = 25
            "nombre" = maria
            "colorPelo" = morena
            "sexo" = Mujer
            "peso" = 60
        }

        def describirPersona(persona):
            print(persona)
```

```python
#region Flag Arguments

#MAL: Estamos creando una funcion que mira si la verdura esta cortada, cocinada y lavada,
# y si no lo esta los corta, y pone los booleans como True, para decir que ya lo hemos hecho todo.

    def procesarVerdura(verdura, cortado=False, cocinado=False, lavado=False):
        if not cortado:
            # Cortamos la verdura
            pass

        if not cocinado:
            # Cocinamos la verdura
            pass

        if not lavado:
            # Lavamos la verdura
            pass


    procesar_verdura("zanahoria", cortado=True, cocinado=False, lavado=True)

#BIEN: En este ejemplo estamos creando varias funciones para hacer cada cosa a las que podemos
# llamar cuando nosotros queramos (como se ve abajo) de esta manera evitamos tener que mirar booleans siempre

    def cortarVerdura(verdura):
        # Cortamos la verdura
        pass

    def cocinarVerdura(verdura):
        # Cocinamos la verdura
        pass

    def lavarVerdura(verdura):
        # Lavamos la verdura
        pass


    verdura = "zanahoria"
    cortar_verdura(verdura)
    lavar_verdura(verdura)
```

```python
#region Flag Arguments ⋯

#region Side Effects

#MAL: Esta funcion se llama mostrarNumero pero no solo esta haciendo eso sino que tambien
# esta cambiando su valor. Esto es un side effect ya que no es lo que nos indica el nombre de la funcion

    def mostrarNumero(num):
        num -= 1
        print(num)


#BIEN: Esta funcion indica claramente lo que hace y no hace nada a escondidas
    def restarYMostrarNumero(num):
        num -= 1
        print(num)
```

```python
#region Dont Repeat Yourself

numeros = [0, 2, 1, 5, 2]

#MAL: Estoy repitiendo un monton de veces la suma de cada numero del array.
# Tal que si yo quisiera cambiar a en vez de sumar, restar, tendria que cambiarlo en las 5 lineas

    numeros[0] += 1
    numeros[1] += 1
    numeros[2] += 1
    numeros[3] += 1
    numeros[4] += 1

#BIEN: Aquí hago un bucle para hacer lo mismo. Asi que para cambiar a una resta solo tendria que cambiar un simbolo

    for numero in numeros:
        numero += 1
```

Bloque 3

```
#region Los comentarios Mienten, Codigo autoexplicativo

#MAL: Al usar a y b como nombres de dos numeros, se ve la necesidad de poner mucho comentario para
# que se entienda. Eso es lo que queremos evitar.

        def suma(a, b):
            """
            Esta función recibe dos números y devuelve su suma.

            Args:
            a: El primer número.
            b: El segundo número.

            Returns:
            La suma de los dos números.
            """
            # Se suma a y b
            resultado = a + b

            # Se devuelve el resultado de la suma
            return resultado

#BIEN: Esta funcion se entiende perfectamente sin comentarios, por usar nombres un poco mas claros

        def suma(numero1, numero2)

            resultado = numero1 + numero2

            return resultado
```

```
#region Los comentarios dicen que hace el codigo no como

#MAL: El comentario encima de la funcion esta explicando exactamente como funciona la funcion

        #esta funcion consigue el resultado de una operacion cogiendo los dos numeros como parametros,
        # guardando el resultado en una variable, y devolviendo esa variable
        def suma(numero1, numero2)
            resultado = numero1 + numero2
            return resultado

#BIEN: El comentario de la funcion explica el objetivo de la funcion solamente

        #Esta funcion devuelve el resultado de una suma entre dos numeros
        def suma(numero1, numero2)
            resultado = numero1 + numero2
            return resultado
```