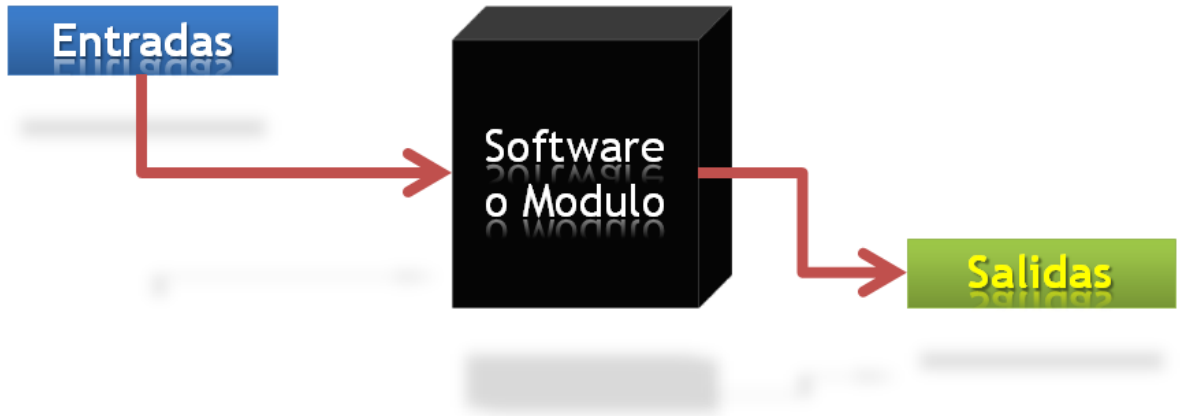
ETS-UT3-4. Pruebas de caja Negra. Clases de equivalencia.

Las pruebas de caja negra se llevan a cabo sobre la interfaz del software.

No disponemos del código fuente del programa que queremos probar.

Para llevarlas a cabo no es necesario conocer la estructura interna del programa ni su funcionamiento. El objetivo es obtenere casos de prueba que demuestren que las salidas que devuelve la aplicación son las esperadas en función de las entradas que se le proporcione



Partición o clases de equivalencia

La partición equivalente es un método de prueba de caja negra que divide los valores de los campos de entrada de un programa en clases de equivalencia.

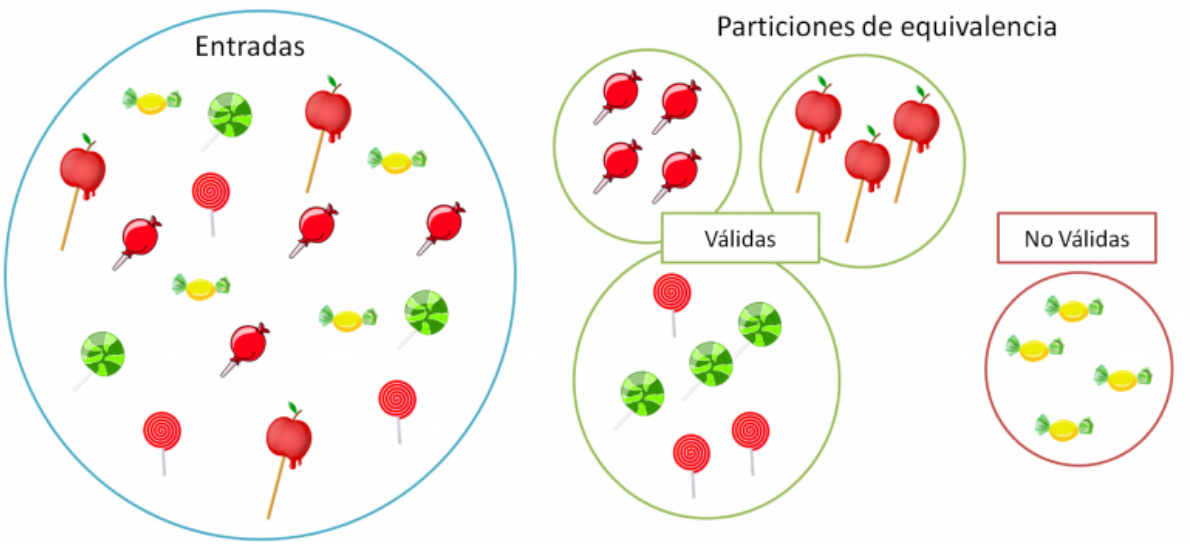
Todos los miembros de cada **clase de equivalencia** comparten ciertas características en común que no son compartidas con miembros de otras clases.

Identificación de clases de equivalencia

Para identificar las clases de equivalencia se examinan las condiciones de entrada y se dividen en grupos en función de las salidas que producen.

Hay dos tipos de clases de equivalencia:

- **Clases válidas**: son las que incluyen valores de entrada válidos. El programa no debería fallar con ellos.
- **Clases no válidas**: son las que incluyen valores de entrada no válidos. Representan los valores de entrada que son erróneos y a los que el programa debería responder o bien rechazándolos o manejando una excepción.



Por ejemplo, supongamos que uno de los campos de entrada es el **numero de empleado** que tiene las siguientes condiciones:

- Debe ser numérico de tres dígitos
- El primer dígito no puede ser 0

Entonces se puede definir una clase de equivalencia**no válida**: **número de empleado < 100**; y otra válida: **número de empleado** entre 100 y 999. Por tanto, para este valor de entrada hemos definido dos clases de equivalencia (1 válida y otra no válida)

El proceso de definir las clases de equivalencia es subjetivo por lo que dos personas podrían obtener clases de equivalencia distintas. Se establecen una serie de directrices que nos guían en el proceso.

Directrices para identificar clases de equivalencia.

1. Rangos

Si una condición de entrada especifica un **rango** de valores, se define una clase de equivalencia válida para todos los valores pertenecientes al rango y dos no válidas, una para los valores menores al límite inferior del rango y otra para los mayores al límite superior.

Ejemplo: días de la semana en formato numérico; números del 1 al 7 ambos inclusive:

- Clase válida: 1 <= dia <= 7
- Clase no válida: dia < 1
- Clase no válida: dia > 7

2. Valor específico:

La condición de entrada requiere de un **valor específico**. De forma análoga a la anterior se crean 1 clase de equivalencia válida y dos no válidas.

Ejemplo: un descuento que se aplica sólo a personas que tienen 18 años:

- Clase válida: edad = 18
- Clase inválida: edad < 18
- Clase inválida: edad > 18

3. Perteneciente a un conjunto

Cuando un valor es válido si es **miembro de un conjunto** se define una clase de equivalencia válida para cada uno de los valores del conjunto y una no válida, para aquellos valores que no pertenecen a él.

Ejemplo: el valor de entrada sólo puede corresponder a uno de los colores RGB escrito en minúscula: «red», «green», «blue»:

- Clase válida: color = "red"
- Clase válida: color = "green"
- Clase válida: color = "blue"
- Clase no válida: color = cualquier otro valor

4. Valor lógico

El valor de entrada se corresponde a la evaluación de una condición. Se establece una clase de equivalencia válida (se cumple la condición) y otra no válida (no se cumple).

Ejemplo: La edad debe ser mayor o igual a 18

- Clase válida: edad >= 18
- Clase no válida: edad < 18

En resumen:

Condición de entrada	Nº clases de equivalencia válidas	Nº clases de equivalencia no válidas
1. Rango	1 válida (incluye valores del rango)	2 no válidas (1 por encima rango, 1 por debajo rango)
2. Valor específico	1 válida (incluye dicho valor)	2 no válidas (1 por encima valor, 1 por debajo valor)
3. Miembro conjunto	1 válida (incluye miembros del conjunto)	1 no válida (valores que no pertenecen al conjunto)
4. Lógica	1 válida (valores cumplen condición)	1 no válida (valores no cumplen condición)

Identificar los casos de prueba

Una vez identificadas las clases válidas y no válidas, para hacer los casos de prueba se siguen los siguientes pasos:

- Los casos de prueba válidos cubren tantas clases de equivalencia válidas como sea posible.
- Los casos de prueba no válidos cubre una sola clase de equivalencia no válida. (Una clase no válida puede enmascarar a otra al terminar la ejecución del caso de prueba antes de ejecutar las siguientes)
- Las clases de equivalencia se van añadiendo a la tabla hasta que todas las clases de equivalencia válidas y no válidas han sido cubiertas.

A cada clase (válida y no válida) se le asigna un identificador

Supuesto

Se va a realizar la entrada de datos de un empleado desde un formulario. Se definen 3 campos de entrada:

- **Empleado**: número de 3 dígitos que no empiece por 0. Campo de entrada numérico.
- **Departamento**: en blanco o número de 2 dígitos. Entrada se teclea manualmente.
- **Oficio**: Analista, Diseñador, Programador o Elige oficio. Entrada se elige de lista desplegable.

Si la entrada es correcta el programaa asigna un salario según estas normas:

- Oficio Analista -> **S1** = 2500
- Oficio Diseñador -> **S2** = 1500
- Oficio Programador -> **S3** = 2000

Si la entrada es incorrecta el programa muestra mensaje indicando la entrada incorrecta:

- **ER1** si el **Empleado** no es correcto.
- **ER2** si el **Departamento no es correcto.
- **ER3** si no se ha elegido **Oficio**

Tenemos la siguiente función de Python, de la que no disponemos del código, que recibe los parámetros de entrada y debe generar la salida correspondiente:

```
def salario_empleado(num_empleado, num_departamento, oficio):
    """
    Dada información de empleado devuelve su salario
    :param num_empleado: int
    :param num_departamento: int
    :param oficio: str
    :return int
    """
```

1. Empezamos creando tabla que represente las clases de equivalencia

Para representar las clases de equivalencia para cada condición de entrada se puede usar una tabla de la siguiente forma:

Condición de entrada	Clase de equivalencia	Clases Válidas	COD	Clases no válidas	COD
Empleado	Rango	100 <= empleado <= 999	V1	empleado <100	NV1
				empleado > 999	NV2
Departamento	Lógica	En blanco	V2	No es un número	NV3
Departamento	Valor	Valor de 2 dígitos	V3	Número más de 2 dígitos	NV4
				Número menos 2 dígitos	NV5
Oficio	Miembro conjunto	"Programador"	V4	Oficio = "Elige oficio"	NV6
		"Analista"	V5		
		"Diseñador"	V6		

2. A partir de esa tabla se generan los **casos de prueba**:

Caso de prueba	Clases de equivalencia	Empleado	Departamento	Oficio	Resultado esperado
CP1	V1, V3, V4	200	20	Programador	S3
CP2	V1, V3, V5	110	10	Analista	S1
CP3	V1, V2, V6	220		Diseñador	S2
CP4	NV1, V2, V4	90	35	Programador	ER1
CP5	NV2, V2, V5	1300	90	Analista	ER1
CP6	V1, NV3, V6	600	AB	Diseñador	ER2
CP7	V1, NV4, V4	600	110	Programador	ER2
CP8	V1, NV5, V5	500	7	Analista	ER2
CP9	V1, V3, NV6	600	95	Elige oficio	ER3

Para hacer la tabla de casos de prueba se siguieron los pasos descritos anteriormente:

- Los casos de prueba válidos (CP1 a CP6) cubren tantas clases de equivalencia válidas como sea posible.
- Los casos de prueba no válidos cubre una sola clase de equivalencia no válida. (Una clase no válida puede enmascarar a otra al terminar la ejecución del caso de prueba antes de ejecutar las siguientes)

- Las clases de equivalencia se van añadiendo a la tabla hasta que todas las clases de equivalencia válidas y no válidas han sido cubiertas. Por eso en el caso anterior no valtan combinaciones posibles en las conbinaciones válidas (V1, V2, V5), (V1, V2, V6).

3. Una vez que tenemos los casos de prueba. En caso de usar **Doctest** para los casos de prueba estos podrían ser:

```
def salario_empleado(num_empleado, num_departamento, oficio):
    """
    Dada información de empleado devuelve su salario
    :param num_empleado: int
    :param num_departamento: int
    :param oficio: str
    :return int

    CP1
    >>> salario_empleado(200, 20, "Programador")
    2000

    CP2
    >>> salario_empleado(110, 10, "Analista")
    2500

    CP3
    >>> salario_empleado(220, , "Diseñador")
    1500


    CP4
    >>> salario_empleado(90, 35, "Programador")
    'Número de empleado incorrecto'

    CP5
    >>> salario_empleado(1300, 90, "Analista")
    'Número de empleado incorrecto'

    CP6
    >>> salario_empleado(600, 'AB', "Diseñador")
    'Número de departamento incorrecto'

    CP7
    >>> salario_empleado(600, 110, "Programador")
    'Número de departamento incorrecto'

    CP8
    >>> salario_empleado(500, 7, "Analista")
    'Número de departamento incorrecto'

    CP9
    >>> salario_empleado(600, 95, "Elige oficio")
    'Debes elegir un oficio'
    """
```

Recursos

- Particiones de equivalencia: Pruebas de caja negra - Educando con TIC

- Pruebas de caja negra. Técnica de partición equivalente - PV

tags: `ets` `ut3` `partición` `clases de equivalencia`