UT3 - Diseño y Realización de pruebas - Ejercicios

Ejercicio 1

Condición de entrada	Clases válidas	COD	Clases no válidas	COD
código	200 <= código <= 999	V1	numero < 200	NV1
			numero > 999	NV2
			No es un número	NV3
operación	4 <= longitud operación <= 24	V2	longitud operación < 4	NV4
			longitud operación > 24	NV5
			No es una cadena	NV6
orden	orden = “cheque”	V3	otro valor	NV7
	orden = “depósito”	V4		
	orden = “pago”	V5		
	orden = “retirada de fondos”	V6		

Figura 1. Clases de equivalencia

Resultados esperados:

- **S1**: datos correctos
- **ERR1**: código no es correcto
- **ERR2**: operación no es correcto
- **ERR3**: orden no es correcto

Caso de prueba	Clases de equivalencia	Condiciones de entrada			Resultado esperado
		numero	invertir	orden	
CP1	V1, V2, V3	205	“ingreso”	“cheque”	S1
CP2	NV1, V2, V3	10	“ingreso”	“cheque”	ERR1
CP3	NV2, V2, V3	1000	“ingreso”	“cheque”	ERR1
CP4	NV3, V2, V3	“ingreso”	“ingreso”	“cheque”	ERR1

CP5	V1, NV4, V3	205	“out”	“cheque”	ERR2
CP6	V1, NV5, V3	205	“ingreso en el banco con tarjeta”	“cheque”	ERR2
CP7	V1, NV6, V3	205	205	“cheque”	ERR2
CP8	V1, V2, V4	205	“ingreso”	“depósito”	S1
CP9	V1, V2, V5	205	“ingreso”	“pago”	S1
CP10	V1, V2, V6	205	“ingreso”	“retirada de fondos”	S1
CP11	V1, V2, NV7	205	“ingreso”	“fondos”	ERR3

Figura 2. Casos de prueba

Ejercicio 3

Condición de entrada	**Clases válidas**	**COD**	**Clases no válidas**	**COD**
códigoBanco	200 <= códigoBanco <= 999	V1	numero < 200	NV1
			numero > 999	NV2
	códigoBanco nulo	V2		
			no es un número	NV3
códigoSucursal	1000 <= códigoSucursal <= 9999	V3	códigoSucursal < 1000	NV4
			códigoSucursal > 9999	NV5
			No es un número	NV6
numCuenta	10000 <= numCuenta <= 99999	V4	numCuenta < 10000	NV7
			numCuenta > 99999	NV8
			No es un número	NV9
clave	clave = cadena longitud 5	V5	clave longitud < 5	NV10
			clave longitud > 5	NV11
			no es una cadena	NV12

orden	orden = “Talonario”	V6		
	orden = “Movimientos”	V7		
	orden es nulo	V8		

Figura 1. Clases de equivalencia

Resultados esperados:

- **S1**: el usuario recibe un talonario de cheques
- **S2**: el usuario recibe los movimientos del mes en curso
- **S3**: el usuario recibe los dos documentos
- **ERR1**: códigoBanco no es correcto
- **ERR2**: códigoSucursal no es correcto
- **ERR3**: numCuenta no es correcto
- **ERR4**: clave no es correcto

Caso de prueba	**Clases de equivalencia**	**Condiciones de entrada**					**Resultado esperado**
		código Banco	códigoSucursal	numCuenta	clave	orden	
CP1	V1,V3, V4, V5,V6	250	2500	11111	“D3F4G”	“Talonario”	S1
CP2	V1,V3, V4, V5,V7	250	2500	11111	“D3F4G”	“Movimientos”	S2
CP3	V1,V3, V4, V5,V8	250	2500	11111	“D3F4G”	null	S3
CP4	V2,V3, V4, V5,V6	null	2500	11111	“D3F4G”	“Talonario”	S1

Figura 2. Casos de prueba

Ejercicio 5

Condición de entrada	Clases válidas	COD	Clases no válidas	COD
códigoBanco	códigoBanco = 200	V1a	numero = 199	NV1
			numero = 1000	NV2
	códigoBanco = 999	V1b		
	códigoBanco nulo	V2		
			no es un número	NV3
códigoSucursal	códigoSucursal = 1000	V3a	códigoSucursal = 999	NV4
			códigoSucursal = 10000	NV5
			No es un número	NV6
	códigoSucursal = 9999	V3b		
numCuenta	numCuenta = 10000	V4a	numCuenta = 9999	NV7
			numCuenta = 100000	NV8
			No es un número	NV9
	numCuenta = 99999	V4b		
clave	clave = cadena longitud 5	V5	clave longitud = 4	NV10
			clave longitud = 6	NV11
			no es una cadena	NV12
orden	orden = “Talonario”	V6		
	orden = “Movimientos”	V7		
	orden es nulo	V8		

Figura 1. Clases de equivalencia

Resultados esperados:

- **S1**: el usuario recibe un talonario de cheques
- **S2**: el usuario recibe los movimientos del mes en curso
- **S3**: el usuario recibe los dos documentos
- **ERR1**: códigoBanco no es correcto
- **ERR2**: códigoSucursal no es correcto
- **ERR3**: numCuenta no es correcto
- **ERR4**: clave no es correcto

Caso de prueba	**Clases de equivalencia**	**Condiciones de entrada**					**Resultado esperado**
		código Banco	códigoSucursal	numCuenta	clave	orden	
CP1	V1a,V3,V4a, V5,V6	100	2500	10000	“D3F4G”	“Talonario”	S1
CP2	V1b,V3,V4b, V5,V6	999	2500	99999	“D3F4G”	“Movimientos”	S2
CP3	V1,V3a,V4a, V5,V7	250	1000	10000	“D3F4G”	null	S3
CP5	V2,V3b,V4b, V5,V6	null	9999	99999	“D3F4G”	“Talonario”	S1
CP5	NV1, V3a,V4a, V5,V7	199	1000	10000	“D3F4G”	null	ERR1
CP6	NV2, V3a,V4a, V5,V7	1000	1000	10000	“D3F4G”	null	ERR1
CP7	NV3,V3,V4a, V5,V6	“444”	2500	10000	“D3F4G”	“Movimientos”	ERR1
CP8	V1a,NV4,V4a, V5,V6	200	999	10000	“D3F4G”	“Movimientos”	ERR2
CP9	V1a,NV	200	10000	10000	“D3F4G”	“Movimient	ERR2

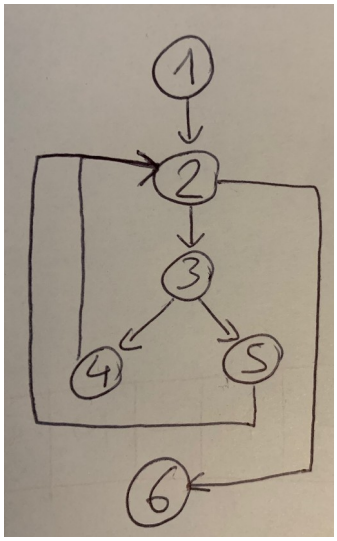
	5,V4a, V5,V6					os”	
CP10	V1a,NV 6,V4a, V5,V6	200	“juan”	10000	“D3F4G”	“Movimient os”	ERR2
CP11	V1a,V3 ,NV7, V5,V6	100	2500	9999	“D3F4G”	“Talonario”	ERR3
CP12	V1a,V3 ,NV8, V5,V6	100	2500	100000	“D3F4G”	“Talonario”	ERR3
CP13	V1a,V3 ,NV9, V5,V6	100	2500	“pedro”	“D3F4G”	“Talonario”	ERR3
CP14	V1a,V3 ,V4a, NV10,V 6	100	2500	10000	“D3F4”	“Talonario”	ERR4
CP15	V1a,V3 ,V4a, NV11,V 6	100	2500	10000	“D3F4G 2”	“Talonario”	ERR4
CP16	V1a,V3 ,NVV4a 9, NV12,V 6	100	2500	10000	45678	“Talonario”	ERR4

Figura 2. Casos de prueba

Ejercicio 6

1. Elaborar el grafo

```
public int mcd (int x, int y){
        int a,b;
1       a=x;
        b=y
    2   while (a!=b){
            3 if (a<b)
                    4 b=b-a;
              else
                    5 a=a-b;
        }
    6   return a;
}
```



2. Calcular la complejidad ciclomática

Número de nodos = 6 ; Número de aristas = 7

V(G) = 7 - 6 + 2 = 3

3. Seleccionar un conjunto de caminos básicos

Camino 1: 1 - 2 - 6
Camino 2: 1 - 2 - 3 - 4 - 2 - 6
Camino 3: 1 - 2 - 3 - 5 - 2 - 6

4. Generar casos de prueba

Camino	Caso de prueba	Resultado esperado
1	x = 5 , y = 5	Se devuelve 5
2	x = 1 , y = 2	Se devuelve 1
3	x = 2 , y = 1	Se devuelve 1

```
float obtener_media()
{
        int n, suma, conta, suma2, total_num, resul;

        scanf (“%d”,&n); /*Lee el dato introducido por pantalla y lo
almacena en la variable n.*/

        do {
                if ((n>=20) || (n<=50)){
                    suma=suma+n;
                        conta=conta+1;
                }
                else
                        Suma2=suma2+n;

                total_num=total_num+1;

                scanf(“%d”,&n);

        }while (n!=0);

        printf(“%d,%d”,total_num,suma2); /*printf en C equivale a
System.out.println en Java*/

        resul=suma/conta;

        return(resul);

}
```