

INGENIERÍA DEL SOFTWARE

TRABAJO PRÁCTICO N° 3

ESTRATEGIAS PARA CASOS DE PRUEBA, AUTOMATIZACIÓN DE PRUEBAS DE
ACEPTACIÓN, PRUEBAS UNITARIAS Y PRUEBAS DE SISTEMA

2023

4k2

Grupo 10

Negro Luqui Franco

Herrera Macarena Del Valle

1. Pruebas de particiones

- a) Determinar las particiones de equivalencia para un programa, cuya especificación establece, que acepta de 4 a 8 entradas que son 5 dígitos enteros mayores que 10000.

Invalido	Valido	Invalido
Numero de entradas < 4	Numero de entradas entre [4-8]	Numero de entradas > 8
Valores < 10000	Valores entre [10000 - 99999]	Valores > 99999

b)

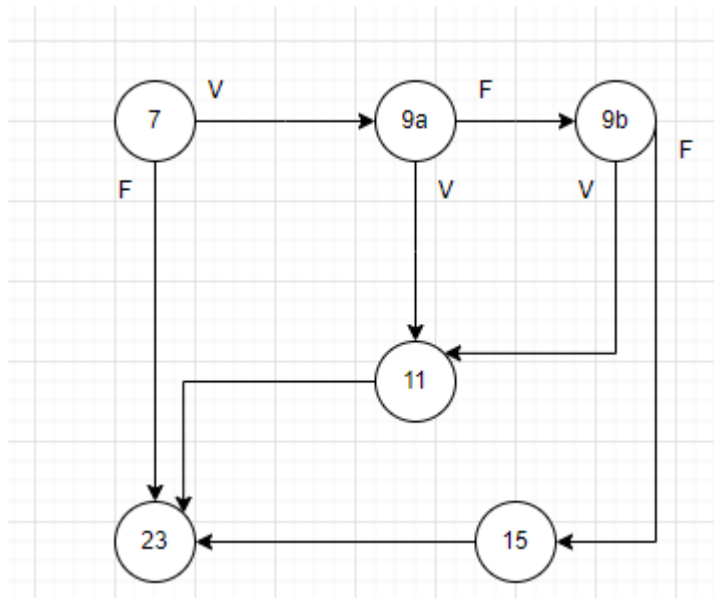
```
1 private static int fibonacci(int n)
2 {
3     int actual = 0;
4     int ant1, ant2;
5     ant1 = 1;
6     ant2 = 0;
7     if (n >= 0)
8     {
9         if ((n == 0) || (n == 1))
10        {
11            actual = n;
12        }
13        else
14        {
15            for (int i = 2; i <= n; i++)
16            {
17                actual = ant1 + ant2;
18                ant2 = ant1;
19                ant1 = actual;
20            }
21        }
22    }
23    return actual;
24 }
```

Numero de particion	Particion
1	$n < 0$
2	$n > 1$
3	$n = [0-1]$
4	$n \neq \text{int}$

2. Pruebas de caminos

Realizar el grafo de flujo para el código del apartado b) del punto 1 y para los siguientes métodos. Calcular la complejidad ciclomática asociada por los tres métodos.

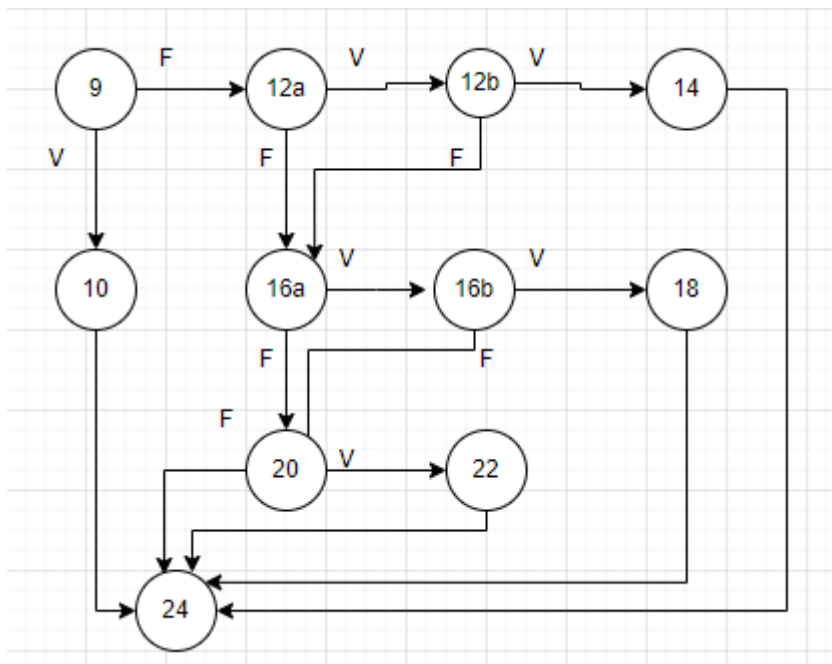
Apartado b punto 1:



Nodos Predicados + 1 = 4

Aristas - Nodos + 2 = 8 - 6 + 2 = 4

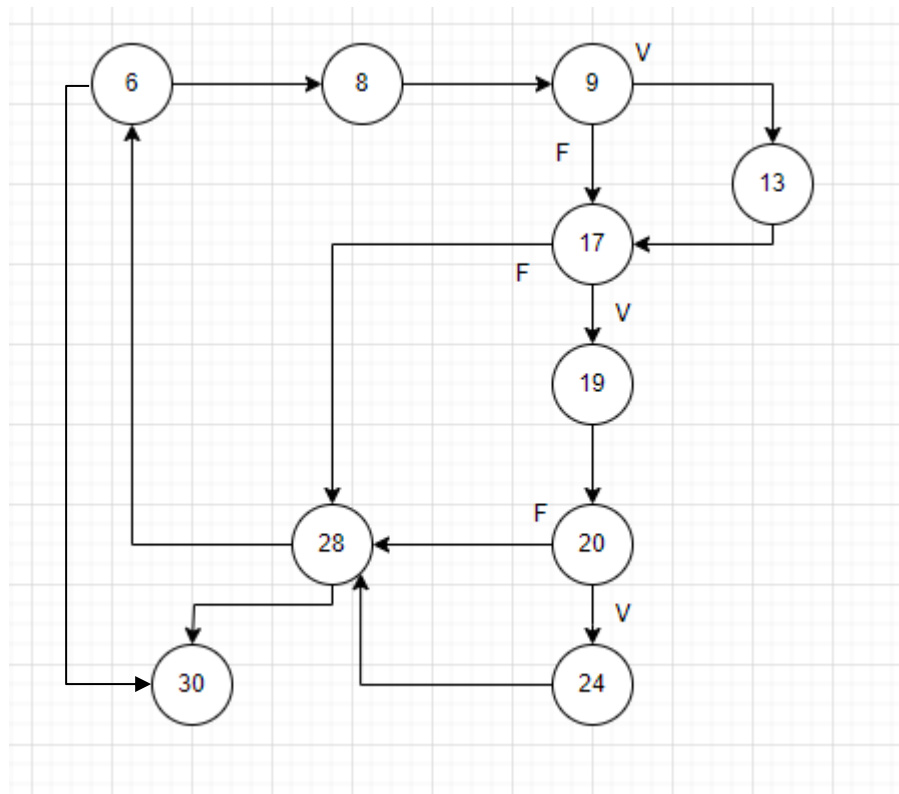
Regiones Cerradas + 1 = 4



Nodos Predicados + 1 = 7

Aristas - Nodos + 2 = 16 - 11 + 2 = 7

Regiones Cerradas + 1 = 7



Nodos Predicados + 1 = 5

Aristas - Nodos + 2 = 12 - 10 + 2 = 5

Regiones Cerradas + 1 = 5

```

1 public class ReglaDeDescuento
2 {
3     private static final double porcentajeMenor = 0.03d;
4     private static final double porcentajeIntermedio = 0.05d;
5     private static final double porcentajeMayor = 0.10d;
6
7     public double Calcular(double total)
8     {
9         if(total <= 0)
10             throw new IllegalArgumentException("El total debe ser mayor a 0");
11
12         if (total > 5000 && total <= 10000)
13         {
14             return total * porcentajeMenor;
15         }
16         if (total > 10000 && total <= 25000)
17         {
18             return total * porcentajeIntermedio;
19         }
20         else if (total > 25000)
21         {
22             return total * porcentajeMayor;
23         }
24         return 0;
25     }
26 }

```

```
1 public static int[] cocktailSort(int[] numbers)
2 {
3     boolean swapped = true;
4     int i = 0;
5     int j = numbers.length - 1;
6     while(i < j && swapped){
7         swapped = false;
8         for(int k = i; k < j; k++){
9             if(numbers[k] > numbers[k + 1]){
10                 int temp = numbers[k];
11                 numbers[k] = numbers[k + 1];
12                 numbers[k + 1] = temp;
13                 swapped = true;
14             }
15         }
16         j--;
17         if(swapped){
18             swapped = false;
19             for(int k = j; k > i; k--){
20                 if(numbers[k] < numbers[k - 1]){
21                     int temp = numbers[k];
22                     numbers[k] = numbers[k - 1];
23                     numbers[k - 1] = temp;
24                     swapped = true;
25                 }
26             }
27         }
28         i++;
29     }
30     return numbers;
31 }
```

3. Pruebas de Unidad (unitarias)

Plantear las pruebas unitarias para la clase **ReglaDeDescuento**.

```
@Test
public void calcularDescuentoConTotalMenora0yTiraExcepcion() {
    var desc = new ReglaDeDescuento()
    try {
        desc.Calcular(-5);
        fail();
    } catch (Exception error){
        assertEquals("El total debe ser mayor a 0", error.getMessage());
    }
}
```

```
@Test
public void calcularDescuentoConTotalIgualA0yTiraExcepcion() {
    var desc = new ReglaDeDescuento()
    try {
        desc.Calcular(0);
        fail();
    } catch (Exception error){
        assertEquals("El total no puede ser igual a 0", error.getMessage());
    }
}
```

```
@Test
public void entre5000y10000elPorcentajeDebeSerMenor(){
    var descMenor = new ReglaDeDescuento();
    valor = descMenor.Calcular(7000);
    assertEquals(valor, 210);
}
```

```

@Test
public void entre10000y25000elPorcentajeDebeSerIntermedio(){
    var descIntermedio = new ReglaDeDescuento();
    valor = descIntermedio.Calcular(15000);
    assertEquals(valor, 750);
}

@Test
public void totalMayorA25000ElPorcentajeDebeSerMayor(){
    var descMayor = new ReglaDeDescuento();
    valor = descMayor.Calcular(50000);
    assertEquals(valor, 5000);
}

```

4. Automatización de Pruebas de Aceptación y Pruebas Unitarias

- Automatizar, por lo menos, 3 (tres) escenarios en Gherkin realizados para el TP N° 2.
- Durante el proceso de automatización deberán realizarse, por lo menos, 3 (tres) pruebas unitarias.

5. Pruebas de Versión (sistema)

Para el caso de uso Realizar Venta diseñar 2 (dos) casos de prueba. Los casos se deben preparar en la plantilla que se adjunta.

Plantilla para caso de prueba

Caso de Prueba	
ID:	Nombre: Eliminar Articulo de la venta
Descripción: Eliminar Articulo de la venta	
Prioridad: Media/Alta	CU / HU: Realizar Venta
Módulo / Funcionalidad:	
Diseñado por:	Fecha:
Ejecutado por:	Fecha:
Precondiciones: Venta en proceso Uno o mas artículos seleccionados	

Paso	Acción	Resultado Esperado	Pasó / Falló	Comentarios
1	Se elimina el artículo de la tabla de artículos seleccionados	Se elimina el artículo de la venta		
2		Se actualiza el Sub Total de la venta		
3				
4				

ID: Identificador | CU: Caso de Uso | HU: Historia de Usuario

Ejemplo

Caso de Prueba	
ID: 10	Nombre: Crear nueva OP
Descripción: Crear una nueva OP	
Prioridad: Alta	CU / HU: Crear OP
Módulo / Funcionalidad: Administración de OP	
Diseñado por:	Fecha:
Ejecutado por:	Fecha:

Precondiciones:

Usuario autenticado y con permisos para crear OP
 Modelos activos cargados
 Colores activos cargados
 Línea activa seleccionada

Paso	Acción	Resultado Esperado	Pasó / Falló	Comentarios
1	Ingresar número de OP			
2	Seleccionar el modelo	Visualizar el objetivo relacionado al modelo		
3	Seleccionar el color			
4	Confirmar	Mensaje de creación exitosa. Registro almacenado correctamente en la base de datos.		

ID: Identificador | CU: Caso de Uso | HU: Historia de Usuario

Nota: el trabajo será entregado a través de un repositorio Git a definir.