Entornos de Desarrollo

PRUEBAS DE CAJA BLANCA

DEFINICIÓN

Se pretende verificar la estructura interna de cada componente de la aplicación.

NO se pretende asegurar que los resultados producidos sean correctos, sino que se trata de comprobar que se van a ejecutar todas las instrucciones del programa, es decir que no existe código no usado, que los caminos lógicos se van a recorrer, etc.

Habitualmente se denominan **pruebas de cobertura del código** y su cumplimiento determina la mayor o menor seguridad en la detección de errores.

CRITERIOS DE COBERTURA

- **Cobertura de sentencias**: generar casos de prueba suficientes para que cada instrucción se ejecute al menos una vez.
- **Cobertura de decisiones**: cada opción resultado de una comprobación lógica, se debe probar al menos una vez a cierto y otra a falso.
- **Cobertura de condiciones**: comprobar que cada elemento de una condición se evalúa al menos una vez a cierto y otra a falso.
- **Cobertura de camino de prueba**: cada bucle se debe ejecutar tres veces: la primera sin entrar en su interior, la segunda ejecutando sólo una iteración y la tercera ejecutando al menos dos iteraciones.

TÉCNICA DE DISEÑO

Para el diseño de los **casos de prueba** que garanticen la cobertura del desarrollo, se deberían realizar los siguientes pasos:

- Crear un **grafo** que represente el código del programa.
- Calcular la **complejidad ciclomática** del grafo.
- Determinar tantos **caminos** o recorridos del grafo como la complejidad ciclomática.
- Generar un **caso de prueba** por cada camino, determinando los datos de entrada y resultados esperados.
- Lanzar una **ejecución del programa** por cada caso de uso y **comparar los resultados obtenidos con los esperados** para comprobar el código.

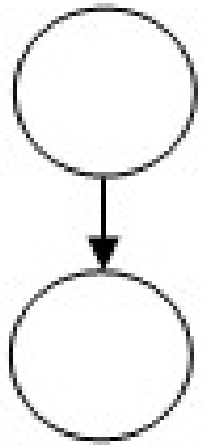
PASO 1: crear el grafo

Se basa en el tipo de instrucciones que tengamos implementadas en nuestro código.

Los tipos de instrucciones o **estructuras básicas** son: secuencia, condición. selección múltiple, iteración, iteración Do.

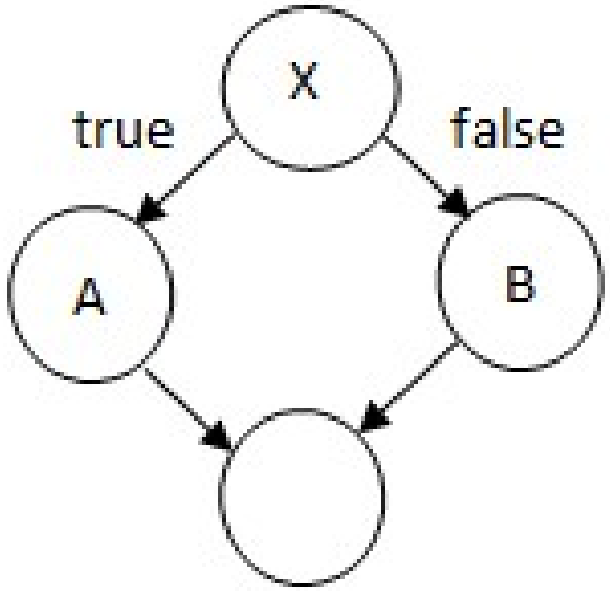
Par el caso de las **estructuras de decisión** tendremos: and y or.

PASO 1: crear el grafo
Secuencia



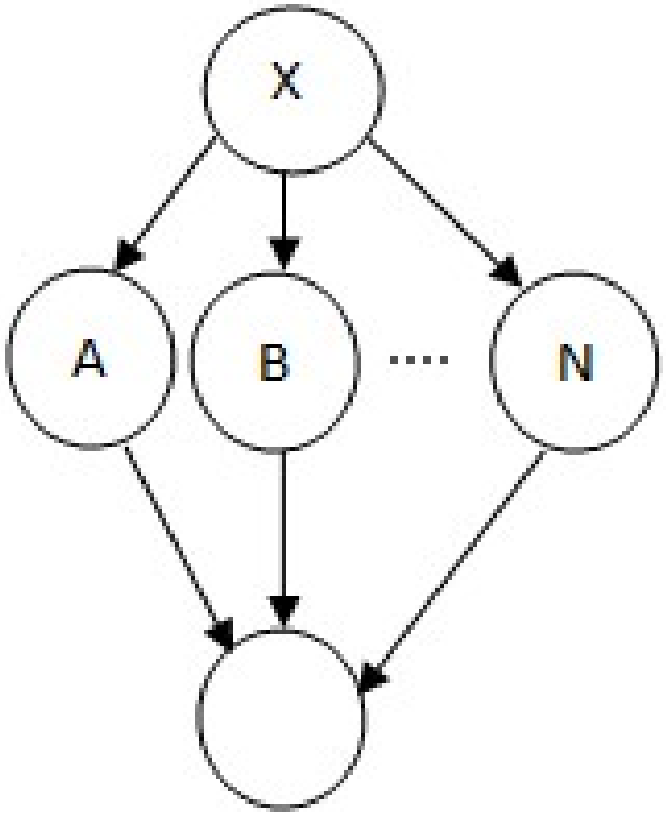
```
String sNota = "10";
System.out.println("Tu nota es: " + sNota);
```

PASO 1: crear el grafo
Condición



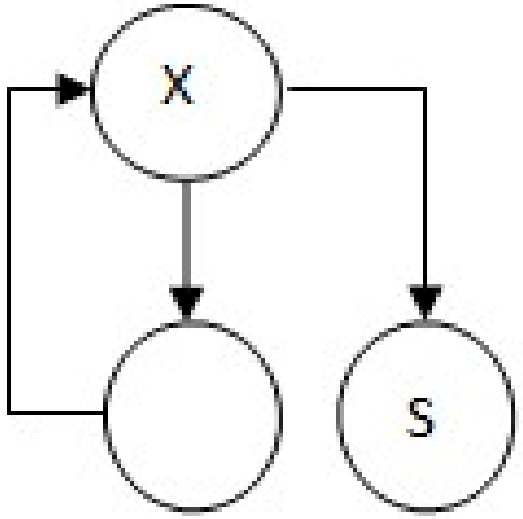

```
int iNota = 3;

if( iNota>= 5)
{ System.out.println("Enhorabuena. Superado."); }
else if ( iNota < 5)
{ System.out.println("La proxima vez sera"); }
```

PASO 1: crear el grafo
Selección múltiple



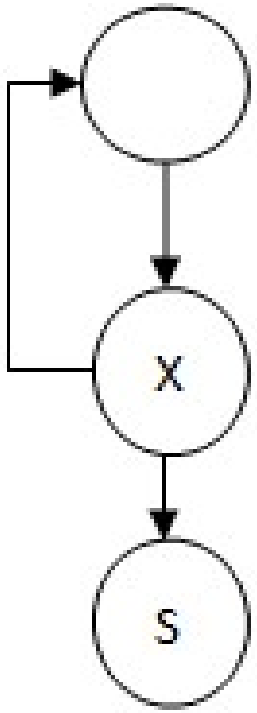

```
switch (iNota) {
case 1: case 2: case 3: case 4:
     { System.out.println("La proxima vez sera");
       break;}
     default:
          { System.out.println("Enhorabuena. Superado."); }
}
```

PASO 1: crear el grafo
Iteración



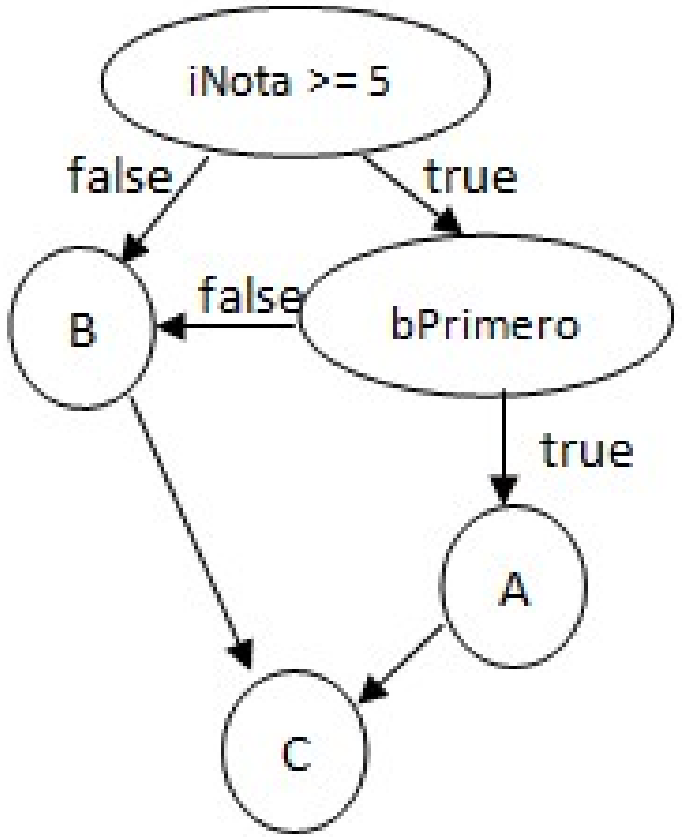

```
int iNumSal = 2;

while(iNumSal > 0)
{
      System.out.println("Hola !!!!!");
      iNumSal--;
}
```

PASO 1: crear el grafo
Iteración Do




```
int iNumSal = 2;

do
{
    System.out.println("Hola !!!!!");
    iNumSal--;
}
while(iNumSal > 0);
```

PASO 1: crear el grafo
And

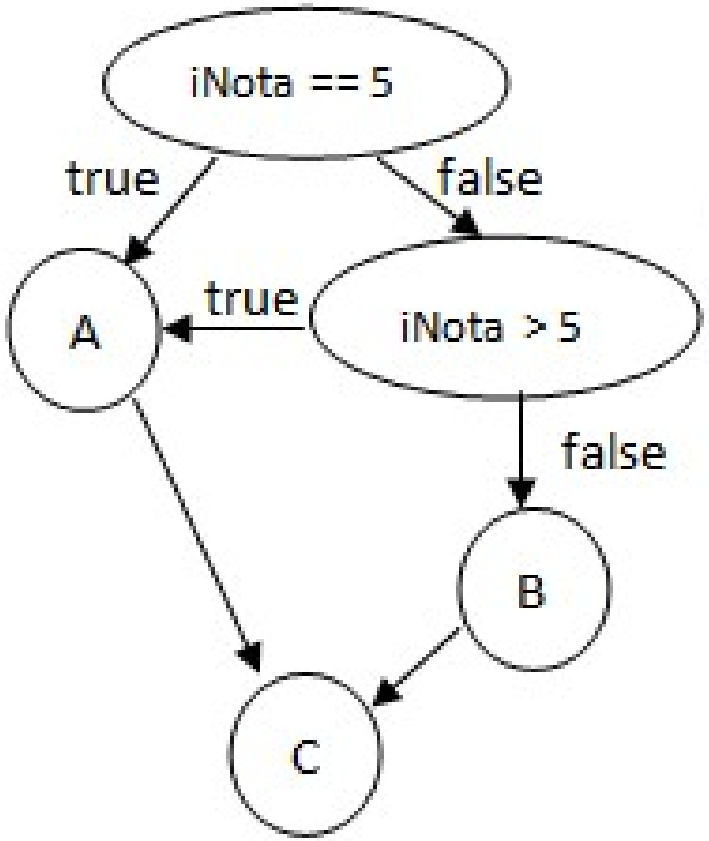



```
int iNota = 6;
boolean bPrimero = true;

if (iNota >= 5 && bPrimero) {
    System.out.println("Enhorabuena. Pasas a 2."); }

else    {
    System.out.println("La proxima vez sera"); }
```

A

B

PASO 1: crear el grafo
Or




```
int iNota = 6;

A   if (iNota == 5 || iNota > 5) {
        System.out.println("Enhorabuena. Superado."); }

B   else    {
        System.out.println("La proxima vez sera"); }
```

PASO 2: complejidad ciclomática

Existen tres métodos de cálculo:

- **Complejidad = a – n + 2**, donde a = número arcos y n = número nodos.
- **Complejidad = r**, donde r = número de regiones cerradas del grafo, incluida la externa.
- **Complejidad = c + 1**, donde c = número de nodos de condición.

PASO 3: caminos de prueba

El número de **caminos de prueba** será igual a la complejidad ciclomática calculada.

Consiste en documentar todos los caminos desde el inicio hasta el final del programa, documentando todos los nodos por los que va pasando la ejecución del camino.

Los caminos deberán ser distintos entre ellos, por tanto deberán diferir en al menos un arco o un nodo.

El orden de definición de caminos deberá ir de más sencillo a más complicado.

PASO 4: casos de prueba

Se definirán tantos **casos de prueba** como caminos de prueba hayamos encontrado.

Consiste en definir los datos de entrada que vamos a utilizar para probar cada camino y los resultados previstos, de modo que cuando ejecutemos el caso de prueba, podamos comparar el resultado esperado con el previsto para determinar la corrección del código.