Sesión 2 : Caja blanca

Conjunto S: Especificaciones del cliente
Lo ideal es que P == S, pero esto nunca se cumple.
La intersección PS (Parte programada y que nos ha pedido el cliente)
El trabajo del teste seleccionar ese conjunto de objetivos que seleccionamos para saber si hemos hecho el trabajo bien.

El tester debe escoger un conjunto de caso de pruebas (S o P)
Una vez escogido generaré un subconjunto T1, T2… que rellenara el conjunto de pruebas programado (no se saldrá de este conjunto (en este caso P).

Este tipo de método(CAJA BLANCA)
- Punto de partida, el código.
- Seleccionar un subconjunto del código para obtener pruebas de ello.
  - Creo grafo (donde están todos todos los comportamientos de P).
  - Obtengo un subconjunto del grado.
  - Obtengo los casos de prueba de estos subconjuntos

Independiente del criterio que va a tener mi conjunto, el numero de filas de esa tabla debe ser el menor posible para sacar el mayor número de errores. (EFECTIVO Y EFICIENTE).

Este tipo de métodos se aplican en pruebas unitarias.

FLUJO DE CONTROL
En cada nodo se representan instrucciones
Cada camino del grafo va a representar un comportamiento, y a partir de ello, de forma dinámica yo puedo hacer mi trabajo.

FLUJO DE DATOS
En cada nodo se representan datos.
Se pueden representar de forma estática y dinámica.

Flujo de control

Dos grandes grupos(sentencias): asignaciones y condiciones.

Cada camino corresponde a un comportamiento.

La representación solo debe tener un punto de entrada y uno de salida.

UNIDAD de prueba = metodo Java.

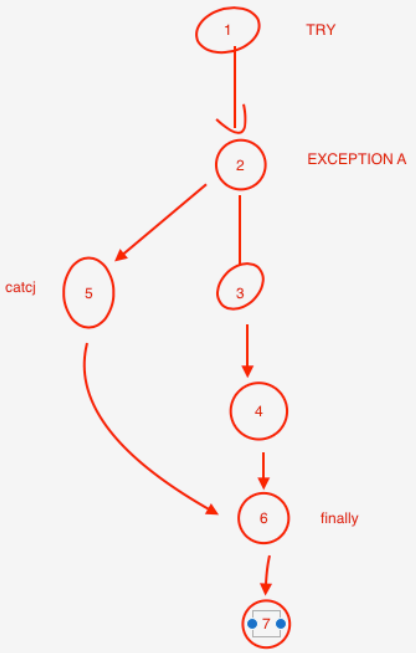
Grafo de flujo de control
CFG representación grafica de una UNIDAD de programa
Dentro de cada nodo, nunca puede haber mas de una condición.
Cada nodo representa un conjunto de sentencias que tengo en el código.
Hay que identificar los nodos, da igual como.
Si el nodo representa una condición, ponerla al lado. Etiquetar aristas cual es cierta y cual es falsa.

CFG (try/catch)
Cada sentencia que puede lanzar una excepción (puede que si y puede que no) bifurcación like a if.

Finally se ejecuta siempre.

```
1.  try {
2.    s1; //puede lanzar Exception1;
3.    s2; //no lanza ninguna excepción
4.    ...
5.  } catch (Exception1 e) {
6.    ...
7.  } finally {
8.        ...
9.  }
10. siguienteSentencia;
```

Inténtalo!



El grafo contiene mi conjunto P, cada uno de los caminos es un comportamiento que esta en el código implementado.

El siguiente paso es seleccionar un subconjunto de caminos. En el momento que selecciono me estoy dejando fuera cosas.

¿cuantos caminos elijo?

- Elegimos todos los caminos.
- Elegimos el conjunto minimo de caminos.
- Elegimos todas las condiciones para que pasemos al menos una vez.

METODO DEL CAMINO BASICO

El objetivo es garantizar con el mínimo numero de casos de prueba, que no nos dejamos ninguna por recorrer y pasamos tanto por si es cierto o falso.

Descripción

- Construimos el grafo de flujo
- Calcular Complejidad ciclomática (CC)
- Obtenemos caminos independientes
- Determinar datos de entrada que se ejecuten todos los caminos
- Finalmente determinar el resultado esperado. (NUNCA JAMAS RELLENAR EL ESPERADO CON EL CODIGO, EL RESULTADO ESPERADO SE OBTIENE SIEMPRE DE EL CONJUNTO S)

Complejidad ciclomatica

Num arcos - Nº nodos +2

Cota superior del numero de filas que va a tener mi tabla.

El numero mide lo complejo que es el código, cuanto mayor sea, mas complejo es.

CC numero de ciclos que hay en un gráfico.

Tambien se puede calcular:

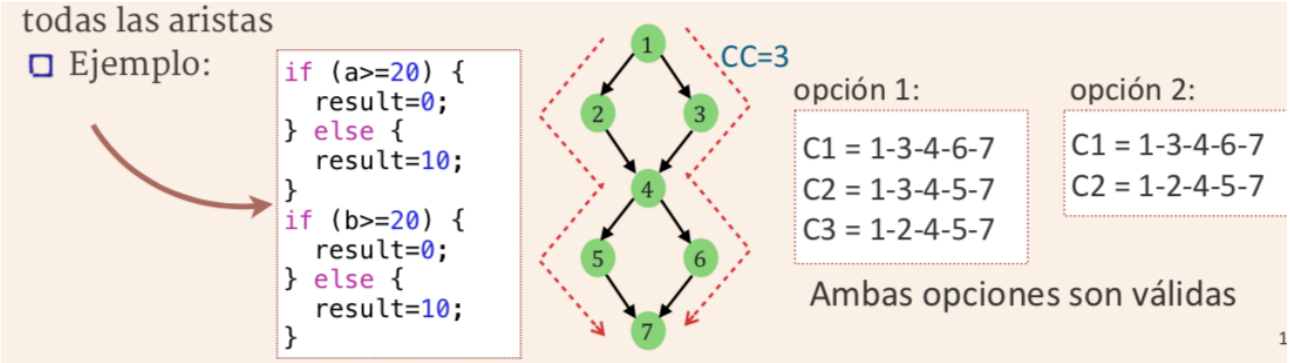
Numero de decisiones +1

Numero de regiones

(SIN SALTOS INCONDICIONALES)

Camino independiente recorre al menos un nodo o una arista que no hemos recorrido antes.

El numero máximo debe ser CC, puede ser inferior.



Bondad de este metodo, localiza el error.

Los datos de entrada del resultado y salida deben ser concreto ya que los casos deberán repetirse.

Hay que saber entradas y salidas no siempre son solo las que están en los parámetros y en el return.