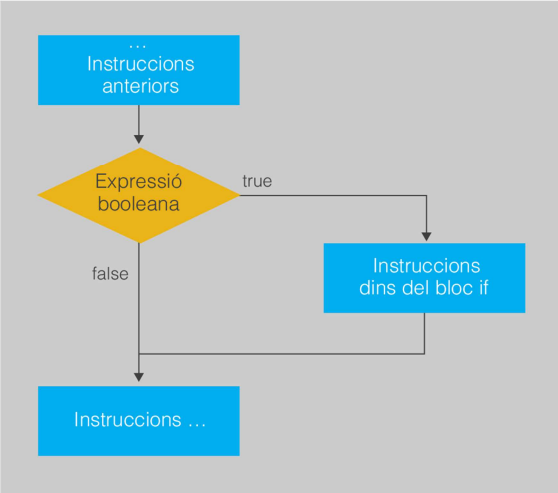
NF2 Estructures de control

Ara que ja sabem la sintaxi bàsica hem de conèixer com escriure estructures condicionals i iteratives, en general estructures de control de flux. Ja hem vist la seva representació en diagrames de flux i pseudocodi, anem a veure com es representen en Java.

Condicional simple

Recordem la seva representació diagramàtica:



La seva sintaxi en java es la següent:

```
//instruccions del programa
if (expressió booleana) { //inici del bloc d'accions

   //Instruccions per executar si l'expressió avalua a true

} // fi del bloc d'accions
//resta d'instruccions del programa
```

Aspectes importants:

- L'expressió booleana que denota la condició lògica pot ser tan complexa com es vulgui, però ha d'estar sempre entre parèntesis.
- Les instruccions que cal executar si la condició és certa estan englobades entre dues claus ({, }). Aquest conjunt es considera un bloc d'instruccions associat a la sentència if (bloc if).
- La línia on hi ha les claus o la condició no acaba mai en punt i coma (;), al contrari que altres instruccions.
- Tot i que no és imprescindible, és un bon costum que les instruccions del bloc estiguin sagnades.
- Si el bloc if només té una única instrucció, l'ús de les claus és opcional, de totes formes es recomanable fer-les servir sempre ja que es més clar la seva interpretació.

Exemple:

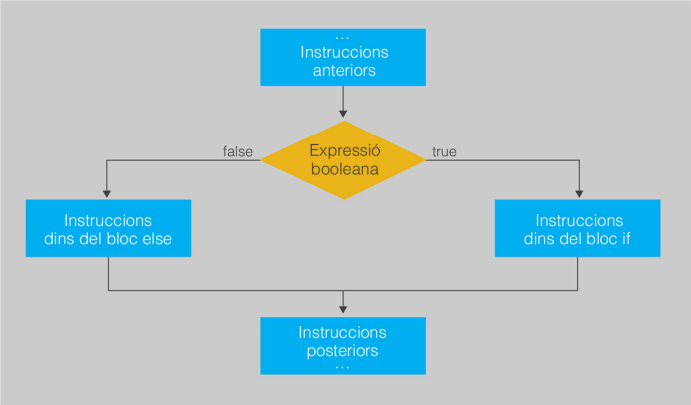
```
if (preu >= COMPRA_MIN)
        float descompteFet = preu * DESCOMPTE / 100;
preu = preu - descompteFet;

        //és equivalent a (fixeu-vos on estan ubicades les claus)

if (preu >= COMPRA_MIN) {
        float descompteFet = preu * DESCOMPTE / 100;
}
        preu = preu - descompteFet;
```

Condicional doble

Recordem la seva representació diagramàtica:



El bloc if té exactament les mateixes característiques que quan s'usa en una estructura de selecció simple. Aquestes es compleixen també per al cas del bloc else, amb la particularitat que no té cap expressió assignada. Simplement, quan la condició lògica de la sentència if no es compleix s'executen les instruccions del bloc else.

```
if( j==0)
{
        ///instruccions
}else{

        ///instruccions
}
```

Condicional múltiple

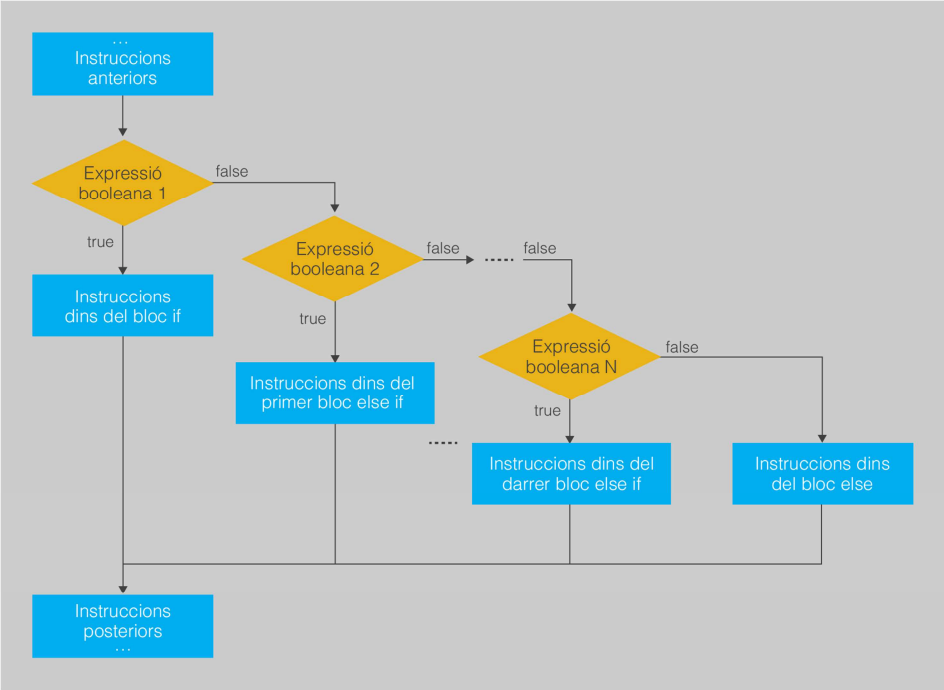
Aquest tipus de control sobre les instruccions del programa té associada la sentència if/else if/else (si és el cas, en aquest altre cas, si no és cap cas). Bàsicament, es tracta de la mateixa estructura que la sentència if/else, però amb un nombre arbitrari de blocs if (després del primer bloc, anomenats else if). En el cas del Java, la sintaxi és la següent:

```
//instruccions del programa
if (expressió booleana 1) {
//Instruccions per executar si l'expressió 1 l'avalua a true
(cert) - Bloc if
```

```
} else if (expressió booleana 2) {
//Instruccions per executar si l'expressió 2 l'avalua a true
(cert) - Bloc else if
} else if (expressió booleana 3) {
//Instruccions per executar si l'expressió 3 l'avalua a true
(cert) - Bloc else if

//...es repeteix tant cops com calgui...

} else if (expressió booleana N) {
//Instruccions per executar si l'expressió N avalua a true (cert)
- Bloc else if
} else {
//Instruccions alternatives si totes les expressions 1...N
avaluen a false (fals) - Bloc else
}
//resta d'instruccions del programa
```



Tot i que no és imprescindible, és un bon costum que les instruccions del bloc estiguin sagnades.

El punt important d'aquesta sentència és que només s'executarà un únic bloc de tots els possibles. Fins i tot en el cas que més d'una de les expressions booleanes pugui avaluar a cert, només s'executarà el bloc associat a la primera d'aquestes expressions dins de l'ordre establert en la sentència.

També és destacable el fet que el bloc else és opcional. Si no volem, no cal posar-lo. En aquest cas, si no es compleix cap de les condicions, no s'executa cap instrucció entre les incloses dins de la sentència.

La sentència switch

Hi ha una estructura de selecció una mica especial, motiu pel qual s'ha deixat per al final. El que la fa especial és que no es basa a avaluar una condició lògica composta per una expressió

booleana, sinó que estableix el flux de control a partir de l'avaluació d'una expressió de tipus enter o caràcter (però mai real).

La sentència switch enumera, un per un, un conjunt de valors discrets que es volen tractar, i assigna les instruccions que cal executar si l'expressió avalua en cada valor diferent. Finalment, especifica què cal fer si l'expressió no ha avaluat en cap dels valors enumerats. És com un commutador o una palanca de canvis, en què s'assigna un codi per a cada valor possible que cal tractar. Aquest comportament seria equivalent a una estructura de selecció múltiple en què, implícitament, totes les condicions són comparar si una expressió és igual a cert valor. Cada branca controlaria un valor diferent. El cas final és equivalent a l'else.

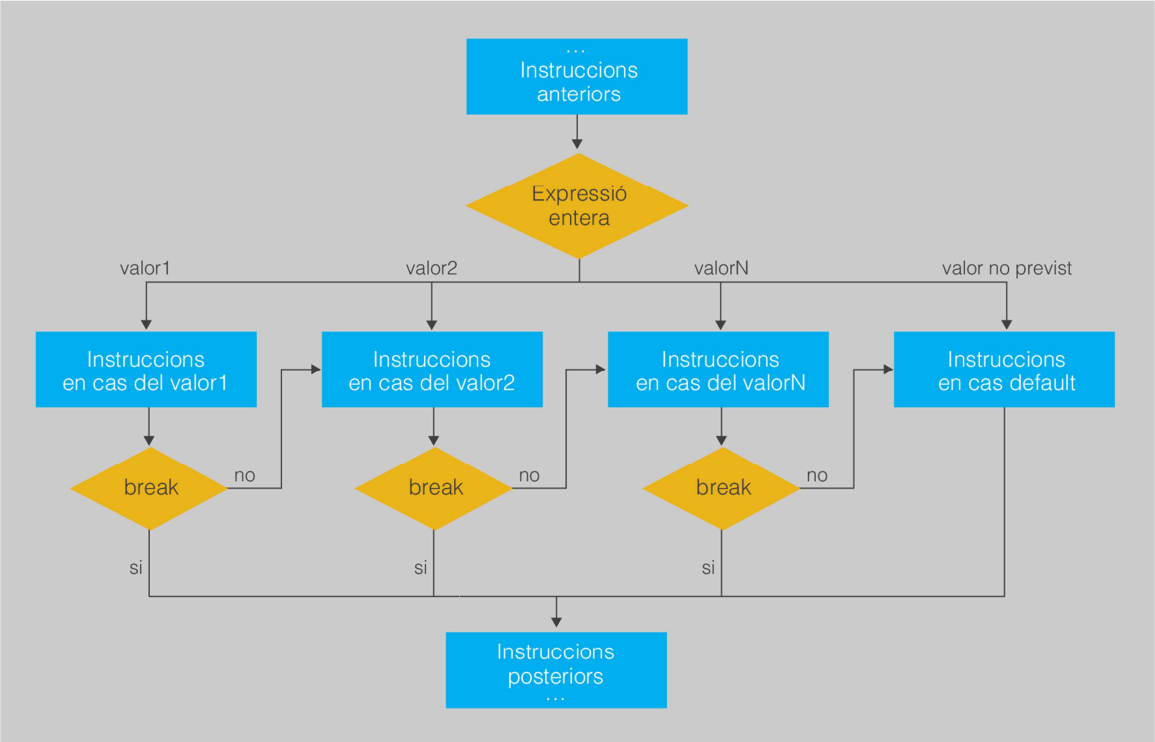
De fet, en la majoria de casos, aquesta sentència no aporta res des del punt de vista del flux de control que no es pugui fer amb una selecció múltiple. Però és molt útil amb vista a millorar la llegibilitat del codi o facilitar la generació del codi del programa. Ara bé, sí que hi ha un petit detall en què aquesta sentència aporta alguna cosa que la resta d'estructures de selecció no poden fer directament: executar de manera consecutiva més d'un bloc de codi relatiu a diferents condicions.

```
switch(expressió de tipus enter) {
              case valor1:
                //instruccions si l'expressió avalua a valor1
                //(opcionalment) break;
              case valor2:
                //instruccions si l'expressió avalua a valor2
                //(opcionalment) break;
              ...
              case valorN:
              //  instruccions si l'expressió avalua a valorN
              //  (opcionalment) break;
              default:
              //  instruccions si l'expressió avalua a algun valor que no és
valor1...valorN
            }
```

break

La paraula clau break és com dir "trenca, surt fora d'aquesta sentència switch".

Ara bé, si algun dels apartats no acaba en break, en executar la darrera instrucció, en lloc de saltar la resta d'apartats, el que es fa és seguir executant les instruccions de l'apartat que ve immediatament després. Així anirà fent, apartat per apartat, fins trobar-ne algun que acabi en break.



Exemple

```
     int month = 8;
String monthString;
switch (month) {
    case 1:  monthString = "January";
             break;
    case 2:  monthString = "February";
             break;
    case 3:  monthString = "March";
             break;
    case 4:  monthString = "April";
             break;
    case 5:  monthString = "May";
             break;
    case 6:  monthString = "June";
             break;
    case 7:  monthString = "July";
             break;
    case 8:  monthString = "August";
                break;
    case 9:  monthString = "September";
             break;
    case 10: monthString = "October";
             break;
    case 11: monthString = "November";
             break;
    case 12: monthString = "December";
             break;
    default: monthString = "Invalid month";
             break;
}
System.out.println(monthString);
```