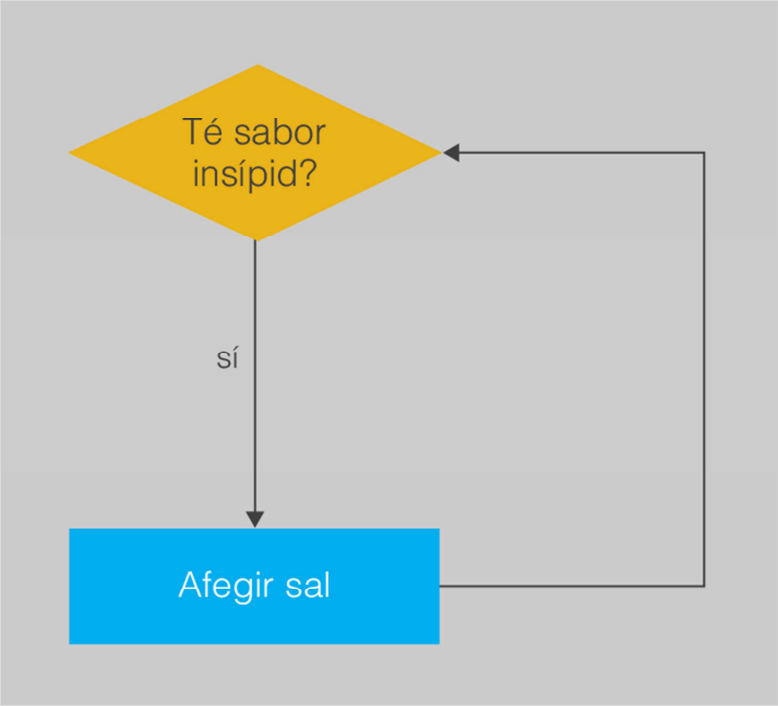
NF2 ESTRUTURES DE CONTROL

Estructures repetitives

Fins al moment, tots els programes que heu estudiat, ja fossin més complexos o més senzills, tenien una característica en comú. El seu flux de control avançava inexorablement cap a la instrucció final del mètode principal sense possibilitat de recular. Sigui quin sigui el camí alternatiu per on s'hi ha arribat, un cop una instrucció ha estat executada, ja no es tornarà a executar mai més. Aquest fet contrasta amb el comportament que podeu observar en moltes activitats que fem diàriament, com per exemple, quan posem sal al menjar que cuinem. Podem repetir diverses vegades el procés de posar sal i tastar-ho fins que el gust sigui el que volem (i llavors ja deixem de posar sal i de tastar). Aquest procés es representa com un diagrama de flux de control a la imatge següent:



Les estructures de repetició o iteratives permeten repetir una mateixa seqüència d'instruccions diverses vegades, mentre es compleixi una certa condició.

En el seu aspecte general, tenen molt de semblant a una estructura de selecció. Hi ha una sentència especial que cal escriure al codi font, unida a una condició lògica i un bloc de codi (en aquest cas, sempre en serà només un). Però en aquest cas, mentre la condició lògica sigui certa, tota la seqüència d'instruccions es va executant repetidament. En el moment en què es deixa de complir la condició, es deixa d'executar el bloc de codi i ja se segueix amb la instrucció que hi ha després de la sentència de l'estructura de repetició.

Un **bucle infinit** és una seqüència d'instruccions dins d'un programa que itera indefinidament, normalment perquè s'espera que s'assoleixi una condició que mai no s'arriba a produir.

Un bucle infinit és un error semàntic de programació. Si n'hi ha, el programa compilarà perfectament de tota manera.

Quan això succeeix, el programa no es pot aturar de cap altra manera que no sigui tancant-lo directament des del sistema operatiu (per exemple, tancant la finestra associada o amb alguna seqüència especial d'escapament del teclat) o usant algun mecanisme de control de l'execució del vostre programa que ofereixi l'IDE usat.

Tenint en compte el perill que un programa s'acabi executant indefinidament, forçosament dins de tot bucle hi ha d'haver instruccions que manipulin variables el valor de les quals permeti controlar la repetició o el final del bucle. Aquestes variables s'anomenen **variables de control**.

Garantir l'assignació correcta de valors de les variables de control d'una estructura repetitiva és extremadament important. Quan genereu un programa, és imprescindible que el codi permeti que, en algun moment, la variable canviï de valor, de manera que la condició lògica es deixi de complir. Si això no és així, tindreu un bucle infinit.

Normalment, les variables de control dins d'un bucle es poden englobar dins d'algun d'aquests tipus de comportament:

- **Comptador:** una variable de tipus enter que va augmentant o disminuint, indicant de manera clara el nombre d'iteracions que caldrà fer.
- **Acumulador:** una variable en què es van acumulant directament els càlculs que es volen fer, de manera que en assolir cert valor es considera que ja no cal fer més iteracions. Si bé s'assemblen als comptadors, no són ben bé el mateix.
- **Semàfor:** una variable que serveix com a interruptor explícit de si cal seguir fent iteracions. Quan ja no en volem fer més, el codi simplement s'encarrega d'assignar-li el valor específic que servirà perquè la condició avaluï false.

Ara veurem la sintaxi de les diferents estructures en Java

While

El nom de la sentència while bàsicament significa: "Mentre això es compleixi, fes això altre…".

Com podeu veure, el seu format és molt semblant a la sentència if, simplement canviant la paraula clau per while. Com ja passava amb les diferents sentències dins les estructures de selecció, si entre els parèntesis es posa una expressió que no avalua un resultat de tipus booleà, hi haurà un error de compilació.

```
while (expressió booleana) {
      //Instruccions per executar dins del bucle
}
```

La estructura do while

La sentència **do/while** permet repetir l'execució del bucle mentre es verifiqui la condició lògica. A diferència de la sentència while, la condició es verifica al final de cada iteració. Per tant, independentment de com avaluï la condició, com a mínim sempre es durà a terme la primera iteració.

```
do {
```

```
            //Instruccions per executar dins del bucle
      } while (expressió booleana);
```

Com podeu veure, és molt semblant a la sentència while, però invertint el lloc on apareix la condició lògica. Per poder distingir clarament on comença el bucle s'usa la paraula clau do. Les instruccions del bucle continuen encerclades entre claus, {…}.

El nom de la sentència do/while vol dir: "Fes això, i continua fent-ho mentre es compleixi aquesta condició".

La sentencia for

La sentència for permet repetir un nombre determinat de vegades un conjunt d'instruccions.

En algunes situacions especials ja es coneix, a priori, la quantitat exacta de vegades que caldrà repetir el bucle. En tal cas és útil disposar d'un mecanisme que representi de manera més clara la declaració d'una variable de control de tipus comptador, l'especificació de fins on s'ha de comptar, i que al final de cada iteració incrementi o disminueixi el seu valor de manera automàtica, en lloc d'haver de fer-ho nosaltres. Automatitzar aquest darrer punt és molt important, ja que evita que per un oblit no es faci i s'acabi generant un bucle infinit.

```
for (inicialització comptador ; expressió booleana ; increment comptador) {
            //Instruccions per executar dins del bucle


}
```