BLOC 3

Índex:

1. Blocs d'un programa informàtic

Un programa informàtic es pot dividir en dos apartats, declaracions i instruccions, els quals, es podran dividir en diversos sub-apartats:

> **Declaracions:** Són tots els objectes/materials que necessita el programa. Un exemple d'aquests són les constants, les variables o els arxius.
>
> **Instruccions:** Són totes les accions o operacions que s'utilitzen per a aconseguir els resultats volguts/necessaris.

Dins d'aquestes instruccions, podem veure 3 subgrups:

> **Entrada:** Són les instruccions provinents de dispositius externs les quals s'emmagatzemen a la memòria interna del dispositiu.
>
> **Procés:** Instruccions que modifiquen els objectes d'entrada i que poden crear altres objectes.
>
> **Sortida:** Conjunt d'instruccions que recullen totes les dades i les passen de la memoria interna a dispositius externs.

2. Tipus de dades compostes

Els tipus de dades compostes són aquelles que estan formades per un conjunt de tipus de dades primitives.

Els tipus de dades composte sprincipals són els següents:

Producte Cartesià: Aquest tipus ens permet agrupar un nombre finit de tipus de dades primitives com a una entitat individual.

Utilitzar aquest tipus significa que:

- Sempre és declara les dimensions i quins són els seus identificadors abans de declarar una instància per a l'unitat en concret.

- Es tracta d'una estructura estàtica, donat que, no es sol augmentar o reduir dimensions després de la definició inicial d'aquestes.

Mapejant: Aquest tipus mapeja valors d'un tipus cap a altres utilitzant relacions entre aquestes.

Això implica que:

- La relació pot ser part de la definició d'altra relació.

- Una relació pot ser part de la definició d'altra relació més gran.

Conjunts disjunts: Amb aquest tipus es fan separacions d'un conjunt. Per a utilitzar una partició s'utilitza un identificador de cadascuna d'aquestes particions.

- L'ús d'aquest identificador ajuda a saber quin tipus de dada és el que s'està utilitzant per a una interpretació i a més a més per a garantir l'ús correcte de cada tipus de dada.

3. Estructures de selecció, repetició i salt

- **Estructures de selecció:**
  Aquest tipus d'estructura es basa en les comandes <if else>, <then> i <end if> amb la qual es declara que si hi ha una condició, la qual podem nomenar x, podrà actuar d'una forma diferent a la inicial.

- **Estructures de repetició:**
  Aquest tipus d'estructura es basa en les comandes <for>, <while> i <do...while> amb la qual es declara que es repetirà el bloc de codi tantes vegades com es dicti o fins que hi hagi una variable que pugui trencar aquesta repetició.

- **Estructures de salt:**
  Aquest tipus d'estructura es basa en les comandes <break> i <continue>, amb les quals pots trencar repeticions o tornar-les a fer utilitzar.

4. Tractament de cadenes

Les cadenes són seqüències de caràcters. D'aquesta informació, podem treure 2 tipus:

String:

String s'utilitza quan es treballa en cadenes que no es poden cambiar

StringBuffer:

StringBuffer s'utilitza quan la cadena que tenim la volem manipular posteriorment.

Com podem crear una cadena?

Molts Strings es poden crear mitjançant caràcters que el compilador (en aquest cas parlem de Java) pugui reconèixer, en aquest cas ficarem aquests caràcters entre parèntesis i cometes <(“...”)>.

Ús de cadenes:

- **Concatenació de cadenes:**
  En el cas de Java, se’ns permet concatenar cadenes mitjançant l’ús de l’operador <+>.
  Exemple: “L’alumne te” + contador + “anys.”

- **Longitud de cadenes:**
  Un mètode que es pot utilitzar a un String és *length,* amb el qual obtindrem el nombre de caràcters que tenim a la cadena.

- **Extracció de caràcters:**
  Aquest és un mètode que s’utilitza un caràcter específic dins d’una cadena. Es pot fer servir mitjançant la comanda <charAt>.

- **Comparació de cadenes:**
  Aquest és un mètode per a comparar 2 cadenes entre si. Per a fer-li és l’utilitzarem mitjançant la comanda <equals>. En cas de que siguin iguals ens donarà el paràmetre <true>. Un altre mètode per comparar és mitjançant la comanda <equalsIgnoreCase> amb la qual es compararà entre 2 cadenes ignorant si hi ha majúscules o minúscules.

  Igualtat:
  S’utilitza la comanda <equals> o amb l’operador <==>. Amb la comanda <equals> compara els caràcters que componen la cadena, mentre que, amb l’operador <==> compara dos referències d’objectes per comprovar si pertanyen a la mateixa instància.

  CompareTo:
  Amb la comanda <CompareTo> podem comparar cadenes per a ordenar-les.

Amb aquest mètode se'ns *retornarà 0* si 2 cadenes tenen el mateix contingut, *negatiu* si la cadena dóna un valor menor al paràmetre que lo donem i *positiu* si aquest valor és major al paràmetre donat.

- **Conversió d'objectes a cadenes:**

  toString() i reverseIt():
  De vegades necessitarem convertir objectes a cadenes quan el mètode que volem utilitzar només accepta cadenes. Per a aquests casos, utilitzarem la comanda <reverseIt()>. Per a fer un pas cap a enrere podem utilitzar la comanda <return dest.toString();>

  valueOf:
  Amb la comanda <valueOf()> podem convertir variables de diferents tipus a cadenes.