BLOC B

Andreu Cutts Panisello
1r Asix/Dam

ÍNDEX

1. Variables

-Què és una variable?

Una variable és com se li'n diu a una unitat per a guardar dades en un programa, el valor d'aquestes variables es guarda sobre un espai en la memòria RAM el qual es reserva per a guardar aquestes dades. Les variables s'utilitzen a la majoria de llenguatges de programació on a un nom se li asocia un contingut.

-Tipus de variables:

Hi han tres tipus de variables a Java:

- Variables locals.
- Variables d'instància.
- Variables estàtiques.

Variable Local

Una variable definida dintre d'un bloc se li'n diu variable local.

Aquestes variables es creen quan el bloc ingressat o mètode es crida i destrueix després de sortir del bloc o quan la crida torna del mètode.
L'abast de aquestes variables només existeix dins del bloc en el que es declara la variable, és a dir, podem accedir a aquestes variables només dins d'aquest bloc.

```
public class StudentDetails
{
    public void StudentAge()
    {   //variable local age
        int age = 0;
        age = age + 5;
        System.out.println("La edad del estudiante es : " + age);
    }

    public static void main(String args[])
    {
        StudentDetails obj = new StudentDetails();
        obj.StudentAge();
    }
}
```

Salida:

```
La edad del estudiante es : 5
```

En el programa anterior, la variable age és una variable local. Si intentem usar la variable age fora del mètode es mostrarà un error.

Variables de instancia

Las variables de instancia són variables no estàtiques i es declaren en una clase fora de qualsevol mètode, constructor o bloc.

Com les variables d'instància es declaren en una clase, aquestes variables es creen quan un objecte de la clase es crea i es destrueix quan es destrueix l'objecte.

A diferencia de les variables locals, podem usar especificadors d'accés per a variables de instancia. Si no especifiquem cap especificador d'accés, s'utilitzarà l'especificador d'accés predeterminat.

```
import java.io.*;
class Points
{
    //Estas variables son variables de instancia.
    //Estas variables están en una clase y no están dentro de ninguna función/método
    int engPoints;
    int mathsPoints;
    int phyPoints;
}

class PointsDemo
{
    public static void main(String args[])
    {   //primer objeto
        Points obj1 = new Points();
        obj1.engPoints = 50;
        obj1.mathsPoints = 80;
        obj1.phyPoints = 90;

        //segundo objeto
        Points obj2 = new Points();
        obj2.engPoints = 80;
        obj2.mathsPoints = 60;
        obj2.phyPoints = 85;

        //mostrando puntos para el primer objeto
        System.out.println("Puntos para el primer objeto:");
        System.out.println(obj1.engPoints);
        System.out.println(obj1.mathsPoints);
        System.out.println(obj1.phyPoints);

        //mostrando puntos para el segundo objeto
        System.out.println("Puntos para el segundo objeto:");
        System.out.println(obj2.engPoints);
        System.out.println(obj2.mathsPoints);
        System.out.println(obj2.phyPoints);
    }
}
```

Salida:

```
Puntos para el primer objeto:
50
80
90
Puntos para el segundo objeto:
80
60
85
```

Com podem veure al programa anterior, les variables engPoints, mathsPoints, phyPoints; són variables de instancia. En cas de que tinguem varios objectes com al programa anterior, cada objecte tindrà les seves pròpies copies de variables de instancia.

Variables estàtiques

Les variables estàtiques també es coneixen com a variables de clase.

Aquestes variables es declaren de forma similar a las variables de instancia, la diferencia es que les variables estàtiques se declaren utilitzant la paraula clau dins de una clase fora de qualsevol constructor o bloc de mètodes.

A diferència de las variables de instancia, **només podem tenir una copia de una variable estàtica per classe**, independentment de quants objectes creem.

Les variables estàtiques es creen a l'inici de la execució del programa y se destrueixen automàticament quan finalitza la execució.

```
import java.io.*;
class Emp {

   // salario como variable estatica
   public static double salary;
   public static String name = "Alex";
}

public class EmpDemo
{
     public static void main(String args[]) {

      //acceder a la variable estatica sin objeto
      Emp.salary = 1000;
      System.out.println(Emp.name + " tiene un salario promedio de: " + Emp.salary);
   }

}
```

Salida:

```
Alex tiene un salario promedio de: 1000.0
```

2. Conversions de tipus de dades.

En Java és possible transformar el tipus d'una variable o objecte en un altre de diferent a l'original. Aquest procés es denomina "conversió" i és algo que hem d'usar amb cura ja que un mal ús de la conversió de tipus és freqüent que doni errors.

Una forma de realitzar conversions consisteix en col·locar el tipus destí entre parèntesis, a l'esquerra de valor que volem convertir de la forma següent: Tipo VariableNova = (NouTipus) VariableAntiga;

Per exemple: int miNumero = (int) ObjecteInteger; char c = (char)System.in.read();

En el primer exemple, hem extret com a tipus primitiu int el valor sencer contingut en un camp de l'objecte integer. En el segon cas, la funció read torna un valor int, que es converteix en un char a causa de la conversió (char), i el valor resultant s'almacena a la variable de tipus caràcter c.

La mida dels tipus que volem convertir és molt important. No tots els tipus és convertiran de forma segura. Per exemple, al convertir un long en un int, el compilador talla els 32 bits superiors del long (de 64 bits), de forma que encaixen amb els 32 bits del int, pe tant si contenen informació útil, aquesta es perdrà. Aquest tipus de conversions que suposen una pèrdua de informació es denominen "conversions no segures" i en general es tracta d'evitar-les, encara que de forma controlada poden usar-se puntualment.

De forma general tractarem de seguir la norma de que "en les conversions s'ha d'evitar la pèrdua d'informació". En la següent taula veiem les conversions que són segures per a no tenir una pèrdua d'informació.

TIPO ORIGEN	TIPO DESTINO
byte	double, float, long, int, char, short
short	double, float, long, int
char	double, float, long, int
int	double, float, long
long	double, float
float	Double

MÈTODE VALUEOF PER A LA CONVERSIÓ DE TIPUS

El mètode valueOf es un mètode sobrecarregat aplicable a nombroses clases de Java i que permet realitzar conversions de tipus. Per exemple:

EXPRESIÓN	INTERPRETACIÓN aprenderaprogramar.com
miInteger = miInteger.valueOf (i)	Con i entero primitivo que se transforma en Integer
miInteger = miInteger.valueOf (miString)	El valor del String se transforma en Integer
miString = miString.valueOf (miBooleano)	El booleano se transforma en String "true" o "false"
miString = miString.valueOf (miChar)	El carácter (char) se transforma en String
miString = miString.valueOf (miDouble)	El double se transforma en String. Igualmente aplicable a float, int, long.

No totes les conversions són possibles. Moltes vegades per despiste els programadors escriuen instruccions de conversions incoherents com miInteger = (int) miString;. El resultat en aquest cas es un error de tipus "Inconvertible types". Un ús típic de valueOf es per a convertir tipus primitius en objectes.

3. Constants. Tipus i utilitat.

Una constant es declara igual que una variable, posant la paraula final abans del tipo (enlloc del tipo podem usar var). La declaració té lloc al principi del mètode main(), igual com fem en les variables.
El nom de les constants s'escriu en majúscules. Si és paraula composta les separarem en '_'.
El valor de les constants no pot canviar.

Es pot fer una divisió de les constants en tres classes:

- constants literals (sense nom)
- constants declarades (amb nom)
- constants d'expressió

Constants literals

Són valors de qualsevol tipus que s'utilitzen directament, no es declaren ja que no tenen nom. En el seguent exemple hi han un parell de constants literals (el 3, el 4, y el 3.1416):
VolumEsfera := 4/3 * 3.1416 * Radi * Radi * Radi;
Constants declarades
També anomenades constants amb nom, són les que es declaren a la seccó const asignan-los-ho un valor directament. Per exemple:

```
const
Pi = 3.141592; (* valor real *)
Min = 0; (* entero *)
Max = 99; (* entero *)
Saludo = 'Hola'; (* cadena caract. *)
```

Constantes d'expressió

També es declaren a la secció constant, pero a aquestes no se'ls hi assigna un valor directament, sinó que se'ls hi assigna una expressió. Aquesta expressió es evalua en temps de compilació i el resultat se li'n assigna a la constant. Exemple:

```
const
Min = 0;
Max = 100;
Interval = 10;
N = (Max - Min) div Interval;
Centre = (Max - Min) div 2;
```

4. Operadors del llenguatge de programació.

Un operador realitza una funció, pren un o més arguments i retorna un resultat. Quan hem vist les variables, hem definit un tipus de dada amb un conjunt de operacions associades. Es de esperar que podem realitzar operacions aritmètiques amb nombres i lògiques amb els booleans. Aquestes operacions es representen mitjançant operadors.
Els operadors, al igual que els mètodes, ses poden sobrecarregar, es a dir es pot redefinir la seva funcionalitat depenent dels tipus de dades dels operands que recibeix. Així sí, podem indicar que l'operador (+) realitza una suma aritmètica si els operands que rep són dos enters i una concatenació si rep una cadena i un altre objecte.

Tipus d'operadors

- **Operadors aritmètics**

Realitzen les operacions aritmètiques bàsiques; suma (+), resta (-), multiplicació (*), divisió (/) i mòdul (%) per a dades de tipus numèriques, tant enters com reals. Aquestes operacions són binàries perquè admeten dos operands.

- **Operadors relacionals**

Revisant algunes definicions matemàtiques, ens adonem que els nombres conformen un conjunt ordenat. Cada un té una posició relativa. Sabem que el 2 “és menor que” el 4 i que el 6 “és major que” el 1. Al comparar dos nombres, realitzem una funció de relació.
En java disposem dels operadors relacionals per a verificar si es compleix una relació. Per exemple el operador de equivalència (==) ens retorna un valor de vertader si els operadors són iguals. Aquestes operacions comparen dos valors numèrics i retornen un valor booleà.

Operador	Utilización	Resultado
>	A > B	verdadero si A es mayor que B
>=	A >= B	verdadero si A es mayor o igual que B
<	A < B	verdadero si A es menor que B
<=	A <= B	verdadero si A es menor o igual que B
==	A == B	verdadero si A es igual a B
!=	A != B	verdadero si A es distinto de B

- **Operadors booleans**

Com podem suposar, treballen amb operands booleans. Realitzen les operacions lògiques de conjunció (AND), disjunció (OR), negació (NOT) i la disjunció exclusiva (XOR).

Nombre	Operador	Utilización	Resultado
AND	&&	A && B	verdadero cuando A y B son verdaderos. Evaluación condicional.
OR	||	A || B	verdadero cuando A o B son verdaderos. Evaluación condicional.
NOT	!	!A	verdadero si A es falso.
AND	&	A & B	verdadero cuando A y B son verdaderos. Siempre evalúa ambos operandos.
OR	|	A | B	verdadero cuando A o B son verdaderos. Siempre evalúa ambos operandos.
XOR	^	A ^ B	verdadero cuando A y B son diferentes

Cada una de les operacions té associada una taula de veritat. Això ens permet veure el resultat de un operador aplicat a les diferents combinacions de valors que poden tenir els operands.

- **Operadors de bits**

Les dades en un ordinador internament es representen en codi binari. El microprocessador només entén zeros i uns. Després, gràcies a una serie de processos, nosaltres veiem aquest codi ja transformat en nombres, caràcters, imatges i sons. Pero en realitat tot segueix sent binari.

- **Operadors d'assignació**

És l'operador de assignació. Aquest apareix amb un signe d'igual (=). Canvia el valor de la variable que està a l'esquerra per un literal o el resultat de la expressió que es troba a la dreta.

Operadores de asignación

Operación	Operador	Utilización	Operación equivalente
Suma	+=	A += B	A = A + B
Resta	-=	A -= B	A = A - B
Multiplicación	*=	A *= B	A = A * B
División	/=	A /= B	A = A / B
Resto de división	%=	A %= B	A = A % B
Desplazamiento a la izquierda	<<=	A <<= B	A = A << B
Desplazamiento a la derecha	>>=	A >>= B	A = A >> B
Desplazamiento a la derecha sin signo	>>>=	A >>>= B	A = A >>> B
AND de bits	&=	A &= B	A = A & B
OR de bits	|=	A |= B	A = A | B
XOR de bits	^=	A ^= B	A = A ^ B

- **Operadors cast**

En java es pot forçar una dada, una variable o una expressió a convertir-se o canviar-se a un nou tipus de dada.
L'operador cast realitza aquest procés, es a dir converteix dades, variables o expressions a un nou tipus de dada.

5. Tipus de dades simples

Java és un llenguatge de tipat estàtic, es a dir, es defineix el tipus de dada de la variable a la hora de definir aquesta. És per això que totes les variables tindran un tipus de dada assignada.
El llenguatge Java dona de base una serie de dades primitives.

- byte
- short
- int
- long
- float
- double
- boolean
- char

És important saber que aquests són tipus de dades del llenguatge i que no representen objectes. Cosa que sí que succeeix amb la resta d'elements del llenguatge Java.

byte

Representa un tipus de dada de 8 bits amb signe. De tal manera que pot guardar els valors numèrics de -128 a 127(ambdós inclosos).

short

Representa un tipus de dada de 16 bits amb signe. De aquesta manera guarda valors numèrics de -32.768 a 32.767.

int

És un tipus de dada de 32 bits amb signe per a guardar valors numèrics. El valor mínimo és -231 y el valor màxim 231-1.

long

Es un tipus de dada de 64 bits amb signe que guarda valors numèrics entre -263 i 263-1

float

És un tipus de dada per a guardar nombres amb coma flotant amb precisió simple de 32 bits.

double

És un tipus de dada per a guardar nombres amb coma flotant amb doble precisió de 64 bits.

boolean

Serveix per a definir tipus de dades booleanes. Es a dir, aquelles que tenen un valor de true o false. Ocupa un bit de informació.

char

És un tipus de dada que representa un caràcter Unicode senzill de 16 bits.

Dato Primitivo	Valor por Defecto
byte	0
short	0
int	0
long	0L
float	0.0f
double	0.0d
char	'u0000'
String (o cualquier objeto)	null
boolean	false