OBJECT ORIENTED PROGRAMMING UNIT1: COMPONENTS FROM A COMPUTER PROGRAM

CFGS - DAW 1

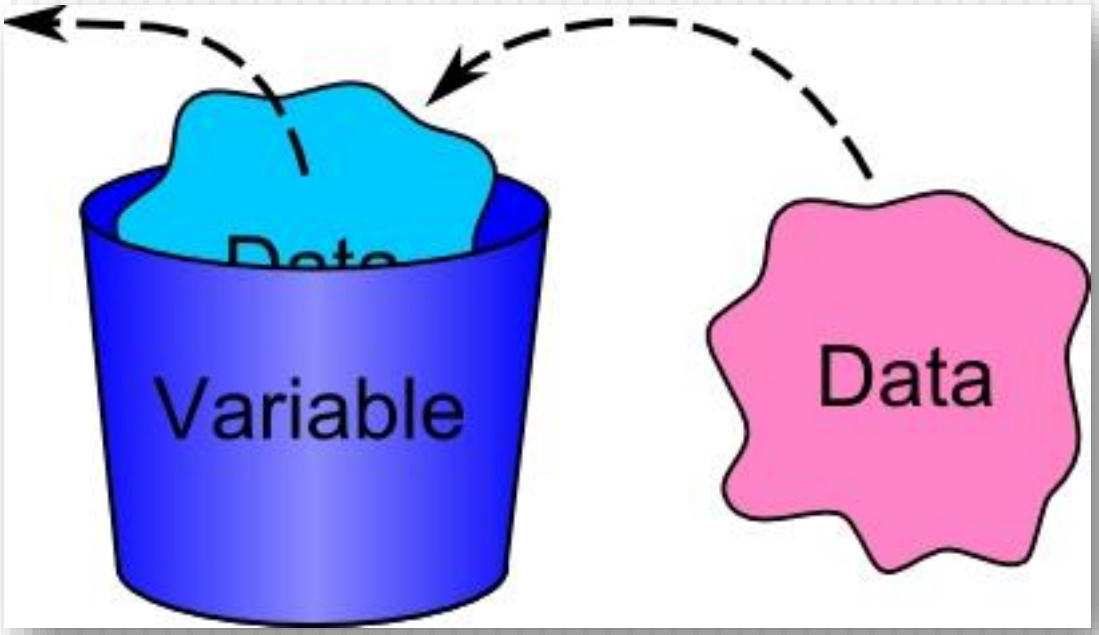
JESUITAS

Sagrado Corazón
Jesuitas · Logroño

Index

Assigning variable types

- *Variables* are the main way that a computer remembers something as it runs a program:
  - A place in the memory where you can save information.



Assigning variable types

- In the HelloWorld program, we used the variable called “a” to hold “Hello World”.
  - String a= "Hello world!";
- In that case the program needed to hold the information to use it later.
  - System.out.println(a);

Assigning variable types

- In a Java program, variables are created with a statement that must include two things:
  - The <u>name</u> of the variable
  - The <u>type</u> of information the variable will store

Data types

Java Primitive Data Types				
Type	Values	Default	Size	Range
byte	signed integers	0	8 bits	-128 to 127
short	signed integers	0	16 bits	-32768 to 32767
int	signed integers	0	32 bits	-2147483648 to 2147483647
long	signed integers	0	64 bits	-9223372036854775808 to 9223372036854775807
float	IEEE 754 floating point	0.0	32 bits	+/-1.4E-45 to +/-3.4028235E+38, +/-infinity, +/-0, NAN
double	IEEE 754 floating point	0.0	64 bits	+/-4.9E-324 to +/-1.7976931348623157E+308, +/-infinity, +/-0, NaN
char	Unicode character	\u0000	16 bits	\u0000 to \uFFFF
boolean	`true, false`	`false`	1 bit used in 32 bit integer	NA

Data types

- Examples:

```
boolean result = true;
char capitalC = 'C';
byte b = 100;
short s = 10000;
int i = 100000;
```

Naming variables

- Variable names in Java can begin with a letter, underscore character (_), or a dollar sign ($). The rest of the name can be any letters or numbers, but <u>you cannot use blank spaces</u>.
- You can give your variables <u>any names</u> you like under those rules.
- Java is <u>case-sensitive</u> when it comes to variable names.
- <u>Convention for naming variables</u>. Examples: <u>g</u>ame<u>O</u>ver, capitalLetters.

Storing Information in Variables

- You can put a value into a variable <u>at the same</u> time that you create the variable in a Java program, <u>or</u> you can also put a value in the variable <u>at any time later</u> in the program.
- Example:
  - double pi = 3.14;

```
double pi;
// There could be more statements in the middle
pi=3.14;
```

- You can also set one variable equal to the value of another variable if they both are of the same type:

```
double c=3.15;
double b=c;
```

Constants

- Definition: "Variables that <u>will not change in value</u>".
- Naming convention: capitalize the names.

```
final static double PI=3.14;
```

Indicates that it is a constant

Static: if you declare as static, there will just be one copy of that variable in the program

- What is the use of constants:
  - The value cannot change.
  - If we use a constant once (**Ex.: IVA**) and it changes, you only have to change it in one statement and not everytime you use it.

Literals

- Definition: “A notation for representing a fixed value in source code”.
- Example:

```
pi=3.14;
```

```
saludo="Hello world!";
```

Keywords

- There is a list of words used by Java compiler, that you cannot use anyway, because they already hava a meaning for Java language.
- Example: class, double…
- Oracle documentation about that, here.

Variables visibility/scope


```
public class Addition {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        int a=150;
        int b=50;
        int c=a+b;
        System.out.print("La suma de " + a + " y "+ b + " es:" + c);
    }

}
```

Local variable

```
public class Addition {
    static int a=150;
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        int b=50;
        int c=sum(b);
        System.out.print("La suma de " + a + " y "+ b + " es:"
    }
    static int sum(int y){
        int z;
        z=a+y;
        return z;
    }
}
```

Member variable of class Addition

Method or function of the class Addition

All About Operators

- Arithmetic operators:

Operator	Description
+	Additive operator (also used for String concatenation)
–	Subtraction operator
*	Multiplication operator
/	Division operator
%	Remainder operator

```
int n1=2, n2;
n2=n1*n1;    //n2=4
n2=n2-n1;    //n2=2
n2=n2+n1+15;     //n2=19
n2=n2/n1;    //n2=9
n2=n2%n1;    //n2=1
```

Writing your first program

A1.4: In Eclipse, make a class called "Root", where you have to calculate the square value of 225 and print it in the command line.
Steps:

- Declare a variable called *a*, which value is going to be 225.
- Declare a variable called *b*, which value is going to be the square value of *a*. *Clue: There is a class called "Math" with a method that you can use it.*

A1.5: In Eclipse, make a class called "Calculator", where you have to calculate the addition of two numbers that you like and print the result.

All About Operators

- Relational operators:

```
==      equal to
!=      not equal to
>       greater than
>=      greater than or equal to
<       less than
<=      less than or equal to
```

```
int n1=2, n2=5;
boolean res;
res=n1>n2;      //res=false
res=n1<n2;      //res=true
res=n1>=n2;     //res=false
res=n1<=n2;     //res=true
res=n1==n2;     //res=false
res=n1!=n2;     //res=true
```

All About Operators

- Unitary operators:

Operator	Description
+	Unary plus operator; indicates positive value (numbers are positive without this, however)
-	Unary minus operator; negates an expression
++	Increment operator; increments a value by 1
--	Decrement operator; decrements a value by 1
!	Logical complement operator; inverts the value of a boolean

```java
int result = +1;
// result is now 1
System.out.println(result);

result--;
// result is now 0
System.out.println(result);

result++;
// result is now 1
System.out.println(result);

result = -result;
// result is now -1
System.out.println(result);

boolean success = false;
// false
System.out.println(success);
// true
System.out.println(!success);
```

All About Operators

- Logic operators:

Operator	Name	Type	Description
!	Not	Unary	Returns `true` if the operand to the right evaluates to `false`. Returns `false` if the operand to the right is `true`.
&	And	Binary	Returns `true` if both of the operands evaluate to `true`. Both operands are evaluated before the And operator is applied.
|	Or	Binary	Returns `true` if at least one of the operands evaluates to `true`. Both operands are evaluated before the Or operator is applied.
^	Xor	Binary	Returns `true` if one — and only one — of the operands evaluates to `true`. Returns `false` if both operands evaluate to `true` or if both operands evaluate to `false`.
&&	Conditional And	Binary	Same as `&`, but if the operand on the left returns `false`, it returns `false` without evaluating the operand on the right.
||	Conditional Or	Binary	Same as `|`, but if the operand on the left returns `true`, it returns `true` without evaluating the operand on the `right`.

```
int m=2, n=5;
boolean res;
res=m>n && m>=n; //res=false
res=!(m<n ||m!=n); //res=false
```

All About Operators

- Assignment operators:

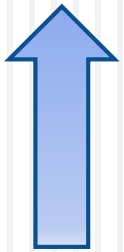
Operador	Uso	Operación
=	A = B	Asignación. Operador ya visto.
*=	A *= B	Multiplicación y asignación. La operación A*=B equivale a A=A*B.
/=	A /= B	División y asignación. La operación A/=B equivale a A=A/B.
%=	A %= B	Módulo y asignación. La operación A%=B equivale a A=A%B.
+=	A += B	Suma y asignación. La operación A+=B equivale a A=A+B.
-=	A -= B	Resta y asignación. La operación A-=B equivale a A=A-B.

```
int num =5;
num += 5;    // num=10. It is the same as doing: num=num + 5;
```

All About Operators

- Operators precedence:

More priority



Less priority

```
( ) [ ] .
-- ~ ! ++ --
new (tipo)expresión
* / %
+ -
<< >> >>>
< <= > >= instanceof
== !=
&
^
|
&&
||
?:
= *= /= %= += -= <<= >>= >>>= &= |= ^=
```

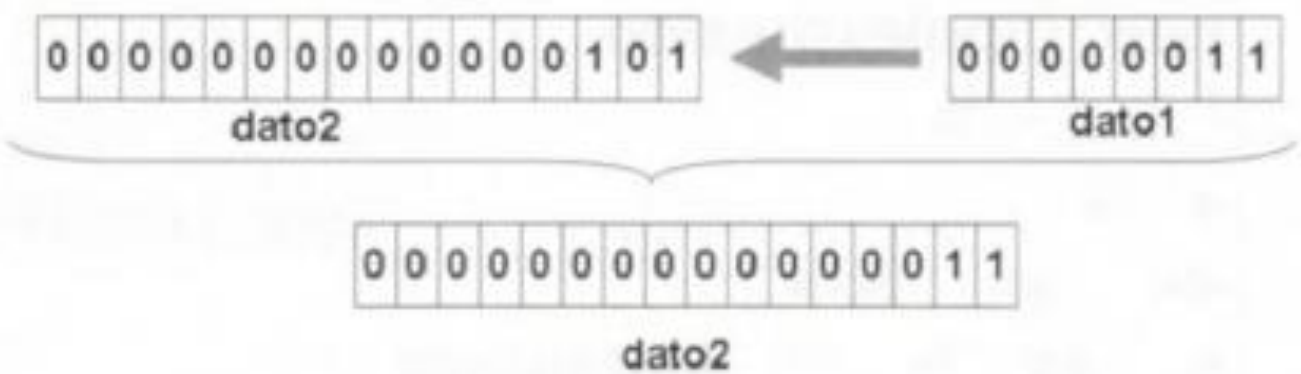
```
int a = 4;
a= 5 * a + 3;
```

Suggestion: Use parenthesis ()

All About Operators

- Type Casting: Conversion between types. Two types of conversions:
  - *Implicit conversion* (also called *widening* conversion): requires the target variable to have more precision than the right one.

```
byte dato1 = 3;      //Entero de 8 bits
short dato2 = 5;     //Entero de 16 bits
dato2 = dato1;
```



All About Operators

- Type Casting:
  - *Explicit conversion* (also named *narrowing* conversion): the target variable have less precision. You are loosing precision, son <u>be careful if you use it</u>.

```
int idato=5;
byte bdato;
bdato=(byte)idato;
System.out.println("bdato es: " + bdato); /*sin problemas,
                                            porque el rango del tipo byte*/

idato=2014;
bdato=(byte)idato;
System.out.println("bdato es: " + bdato); /*resultado incongruente,
                                            porque queda fuera del rango byte*/
```

Type boolean is *incompatible* for conversion

Some exercises

A1.6: Write a program where having two variables of type integer of values 5 and 8, you exchange its values. Print them at the beginning and at the end, as follows:

The initial values are: 5 AND 8
The final values are: 8 AND 5

A1.7: Write a program where you calculate the length of a circumference of radius 3 meters. Print the result as follows:

The length is: 18.84 meters

Some exercises

A1.8: What can you do in this code in order it to work?

```
class suma
{
    static int n1=50;
    public static void main(String [] args)
    {
    int  n2=30, suma=0, n3;
    suma=n1+n2;
    System.out.println("LA SUMA ES: " + suma);
    suma=suma+n3;
    System.out.println(suma);
    }
}
```

Some exercises

A1.9: What erros can you find in this code?

```
class suma
{
    public static void main(String [] args)
    {
    int  n1=50,n2=30,
    boolean suma=0;
    suma=n1+n2;
    System.out.println("LA SUMA ES: " + suma);
    }
}
```

Some exercises

A1.10: Write a program that simulates rolling a pair of dice. You can simulate rolling one die by choosing one of the integers 1, 2, 3, 4, 5, or 6 at random. The number you pick represents the number on the die after it is rolled. You can assign this value to a variable to represent one of the dice that are being rolled.

Do this twice and add the results together to get the total score. Your program should report the number showing on each die as well as the total roll. For example:

The first die: 3

The second die: 5

Your total score: 8

Hint: Remenber the Math.random() function.