

Métodos y arreglos



Métodos

- Similares a las funciones de C
- Forma general

[modificador] tipo identificador (parámetros)

- `main()` es un método

```
public class Ejemplo {  
    public static void main (String args[ ]) {  
        System.out.println("Un simple programa Java");  
    }  
}
```

- Devolución valores (métodos como funciones)
 - Se utiliza la sentencia `return`
 - `return dato_primitivo/objeto;`

Generando números aleatorios

```
public class NumAleatorios {  
    public static int aleatorio(int min, int max) {  
        return (int) (Math.random() * (max-min+1)) + min;  
    }  
    public static void main(String[] args) {  
        System.out.println("En rango 0-6: "+aleatorio(0,6));  
        System.out.println("En rango 1-10: "+aleatorio(1,10));  
        System.out.println("En rango 5-20: "+aleatorio(5,20));  
    }  
}
```

run:

En rango 0-6: 5

En rango 1-10: 8

En rango 5-20: 18

BUILD SUCCESSFUL (total time: 0 seconds)

Generando letras aleatorias

```
public class LetrasAleatoria {  
    public static int aleatorio(int min, int max) {  
        return (int) (Math.random() * (max-min+1)) + min;  
    }  
    public static void main(String[] args) {  
        char letra;  
        letra=(char)aleatorio(97,122);  
        while(letra!='a' && letra!='e' && letra!='i' && letra!='o' && letra!='u') {  
            System.out.print(letra+" ");  
            letra=(char)aleatorio(97,122);  
        }  
        System.out.println();  
    }  
}
```

run:
BUILD SUCCESSFUL (total time: 0 seconds)

run:
c f w h f
BUILD SUCCESSFUL (total time: 0 seconds)

run:
g d z h v m x
BUILD SUCCESSFUL (total time: 0 seconds)

Ámbitos o alcance de variables

□ Ámbito de clase

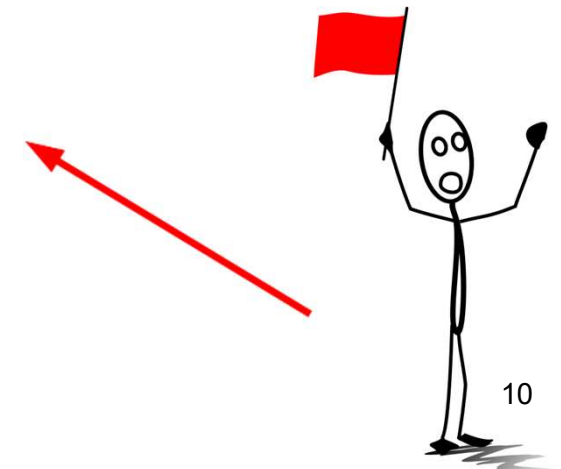
- Variables de instancia o atributos
- “Variable globales” de la clase
 - Pueden ser usadas por los métodos de la clase

□ Ámbito de método

- Variable locales del método y parámetros
- Si una variable de método se llama igual que una de instancia prevalece la del método

□ Ámbito del bloque

- Sólo existe dentro del bloque { ... }



Ejemplo alcance

```
public class Tiempo
```

```
{
```

```
    int minutos;
```

```
    . . .
```

```
    public int sumaTiempo( int hora, int min){
```

```
        int aux;
```

```
        aux = hora * 60 + min;
```

```
        minutos = minutos + aux;
```

```
        if( minutos > 300) {
```

```
            int exceso;
```

```
            exceso = exceso - 300;
```

```
            System.out.print("Min excedidos"+exceso);
```

```
        }
```

```
        else System.out.print("Min actuales"+minutos);
```

```
    . . .
```

```
}
```

Variable
de instancia
o atributo

Argumentos

Variable local

Variable de bloque

Ejercicio: alcance de variables.

```
- public class Confuso {  
    public static int uno=1, dos=2;  
    public static void numeros(int a, int uno){  
        a = a + uno;  
        imprime(uno*dos);  
        if (uno > 0) {  
            int dos;  
            dos = 5;  
            a = a * dos;  
            imprime(dos);  
        }  
        imprime(a);  
    }  
    public static void imprime (int a) {  
        System.out.print(a+" ");  
        System.out.println(uno);  
    }  
    public static void main(String[] args) {  
        numeros(uno, dos);  
    }  
}
```

Tipos de datos

- El tipo de dato de un objeto, determina el **espacio** que éste ocupa en **memoria** y **cómo** usarlo.

- Primitivos
 - Enteros
 - Flotantes
 - Caracter
 - Lógicos
- Referencias a objetos
 - Arreglos o vectores
 - Cadenas de caracteres
 - Entrada/Salida
 - Excepciones
 - Archivos



... veamos arreglos y cadenas

```
String.class.getName().toString()
return "java.lang.String"
byte.class.getName()
return "byte"
(new Object[3]).getClass().getName()
return "[Ljava.lang.Object
(new int[3][4][5][6][7][8][9]).getClass()
return "[[[[[[[I"
```


Arreglos

- ❑ Mecanismo básico para almacenar una colección de entidades del mismo tipo

- Pueden ser datos primitivos u objetos

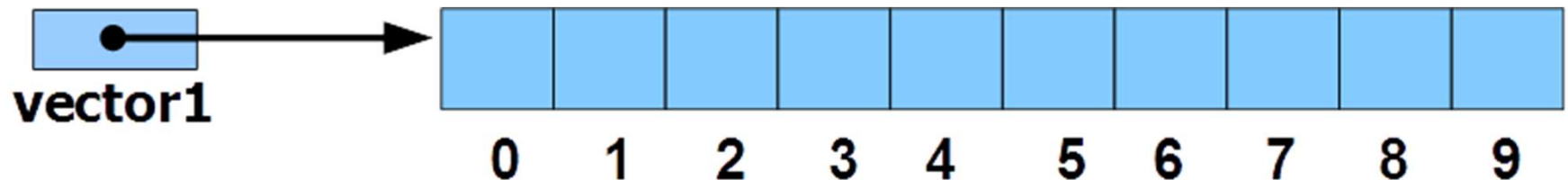
- ❑ Declaración

- Tipo [] identificador;
int [] vector1;

- ❑ Asignación de memoria

- identificador = new tipo [número_elementos];
vector1 = new int [10];

- ❑ El índice inicia en 0



Carmen Villar / René Martínez

Longitud de un arreglo

□ Otras declaraciones y asignaciones

- `int [] vector2 = new int[100];`
- `int [] vector3 = { 3, 4, 2, 6 };`
- `int [][] vector4 = new int[2][3];`
- `int [][] vector5 = {{1,2,3},{0,1,4}};`

□ Número de elementos

- `length`
 - `double [] a = { 10., 2., 4.};`
 - `int longitud = a.length;`

		Columns		
		0	1	2
Filas	0	1	2	3
	1	0	1	4

Imprimir en forma de matriz

```
public class ImprimeMatriz {  
    public static void main(String[] args) {  
        double [][] mat = {{1.2, 2.3, 7.4},{3.1, 9.1, 4.8}};  
        for(int i=0; i<2; i++){  
            for(int j=0; j<3; j++){  
                mat[i][j] *= Math.random();  
                System.out.printf(" %5.3f ",mat[i][j]);  
            }  
            System.out.println();  
        }  
    }  
}
```

run:

0.438 2.208 0.234

2.045 2.137 3.137

BUILD SUCCESSFUL (total time: 0 seconds)


For ampliado en arreglos

□ Sintaxis

```
for ( tipo variable : arreglo )
```

□ Ejemplo

```
int num[] = {2,4,8,16,32};  
for (int i=0; i<num.length; i++)  
    System.out.println("Potencias: " + num[i]);
```



```
int[] num = {2,4,8,16,32};  
for (int pp : num)  
    System.out.println("Potencias: "+pp);
```

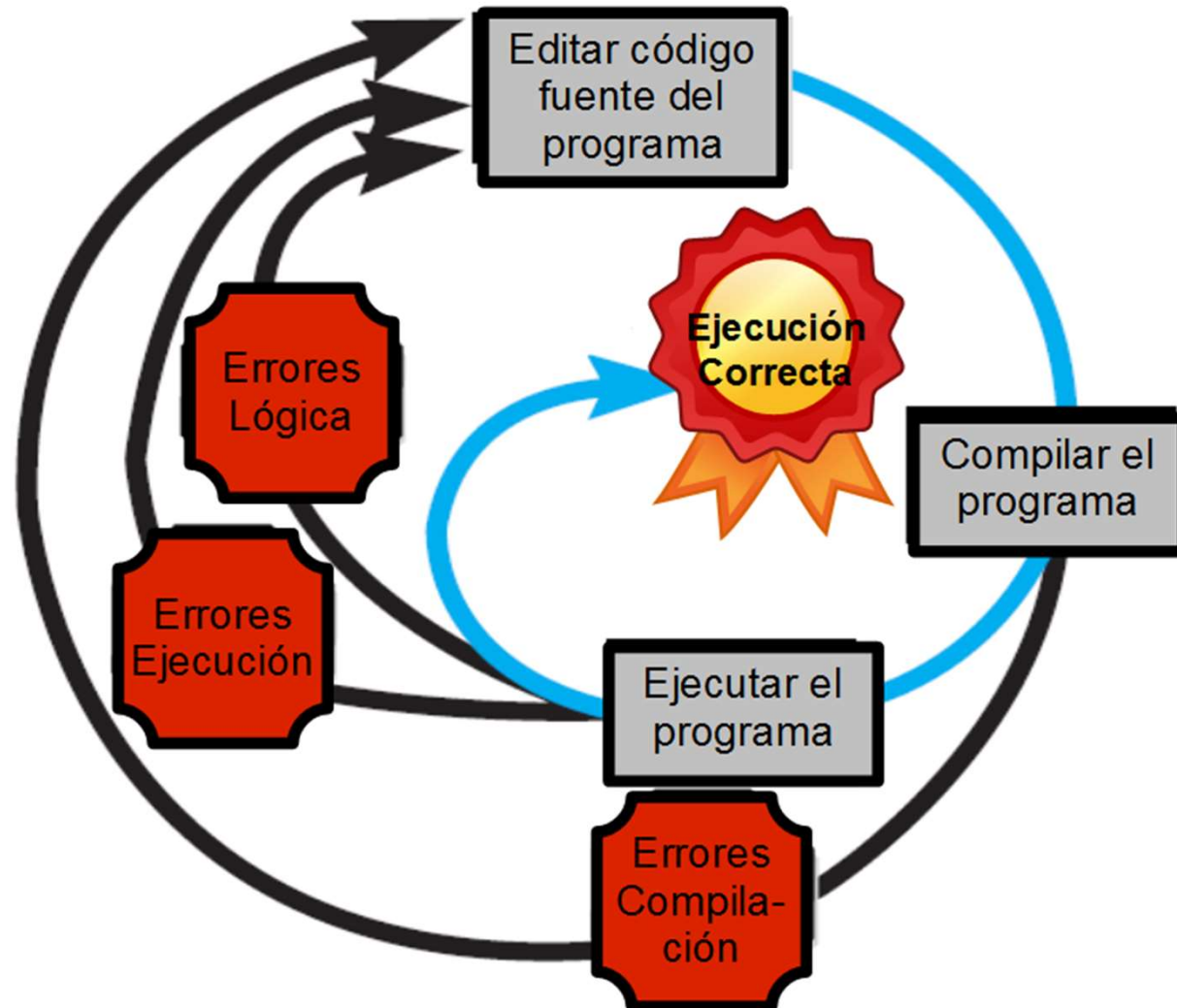
Usando arreglos

```
public class Arreglos {  
    public static void main(String[] args) {  
        char abc[];  
        abc = new char[26];  
        abc = creaArreglo();  
        for ( char i : abc )  
            System.out.print(i+" ");  
        System.out.println();  
    }  
  
    public static char[] creaArreglo(){  
        char[] s = new char[26];  
        for ( int i=0; i<26; i++ )  
            s[i] = (char) ('A' + i);  
        return s;  
    }  
}
```

run:

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
BUILD SUCCESSFUL (total time: 1 second)

EL ciclo: Editar – Compilar - Ejecutar



Tipos de errores

□ Compilación

- Cuando el código se traduce a bytecode
- Ayuda: IDE

```
8      public class Arreglos {  
9          +      public static void main(String[] args)  
17         -      public static char[] creaArreglo(){  
18                 char[] s = new char[26];  
19                 for ( int i=0; i<26; i++ )  
20                     s[i] = (char) ('A' + i);  
21                 return s;  
22             }  
23     }
```

□ Ejecución (Run-time)

- Cuando se ejecuta el programa y se intenta hacer algo no permitido
- Ayuda: Mensajes de error de Java

run:

```
- Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 26  
  |       at presTema2.Arreglos.creaArreglo(Arreglos.java:20)  
    |       at presTema2.Arreglos.main(Arreglos.java:11)  
Java Result: 1
```

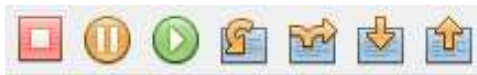
Tipos de errores (cont)

□ Lógica

- Cuando el programa no hace lo que se desea
- Ayuda: Debugger

Debugger o depurador

- Ayuda en la corrección de programas
- Permite ver los valores de las variables en cada paso
- Se debe establecer un break point (punto de interrupción)
- Netbeans
- Debug > Debug file



```
11  * @author maria.villar
12  */
13  public class Precision {
14      public static void main(String[] args) {
15          float f = 0.0f;
16          double d = 0.0;
17          for(int i=0; i<100_000_000; i++){
18              f += 3.0f;
19              d += 3.0;
20          }
21          System.out.println(f);
22          System.out.println(d);
23          System.out.printf("%.1f\n", f);
24          System.out.printf("%.1f\n", d);
25      }
26  }
27
```

Precision > main > for (inti = 0; i < 100000000; i++) >

Output	Variables %	Breakpoints
Name	Type	Value
<Enter new watch>		
Static		
args	String[]	#55(length=0)
f	float	177.0
d	double	174.0
i	int	58