Relación Nº 1: Ejercicios 1 y 2

Empezamos con una serie de ejercicios para aquellos que están comenzando desde cero con el lenguaje Java y tampoco tienen experiencia con ningún otro lenguaje de programación.
En estos ejercicios básicos inicales se realizan las siguientes instrucciones: declarar variables, asignarles un valor, operar con ellas y mostrar resultados por pantalla.
Estos ejercicios son todos de estructura secuencial, es decir, no hay condiciones ni bucles. Tampoco se lee nada por teclado. El objetivo es familiarizarse con la declaración de variables y practicar la salida por consola utilizando los métodos print y println.

Ejercicio básico inicial 1:

Escribe un programa Java que realice lo siguiente: declarar una variable N de tipo int, una variable A de tipo double y una variable C de tipo char y asigna a cada una un valor. A continuación muestra por pantalla:
El valor de cada variable.
La suma de N + A
La diferencia de A - N
El valor numérico correspondiente al carácter que contiene la variable C.
Si por ejemplo le hemos dado a N el valor 5, a A el valor 4.56 y a C el valor ‘a’, se debe mostrar por pantalla:

Variable N = 5
Variable A = 4.56
Variable C = a
5 + 4.56 = 9.559999999999999
4.56 - 5 = -0.4400000000000004
Valor numérico del carácter a = 97

```
/*
 * Solución Ejercicio Básico Inicial 1
 */
package bi1;

public class Main {

    public static void main(String[] args) {
        int N = 5;
        double A = 4.56;
        char C = 'a';
        System.out.println("Variable N = " + N);
```

```
        System.out.println("Variable A = " + A);
        System.out.println("Variable C = " + C);
        System.out.println(N + " + " + A + " = " + (N+A));
        System.out.println(A + " - " + N + " = " + (A-N));
        System.out.println("Valor numérico del carácter " + C + " = " + (int)C);
    }
}
```

Ejercicio básico inicial 2:

Escribe un programa Java que realice lo siguiente: declarar dos variables X e Y de tipo int, dos variables N y M de tipo double y asigna a cada una un valor. A continuación muestra por pantalla:
El valor de cada variable.
La suma X + Y
La diferencia X – Y
El producto X * Y
El cociente X / Y
El resto X % Y
La suma N + M
La diferencia N – M
El producto N * M
El cociente N / M
El resto N % M
La suma X + N
El cociente Y / M
El resto Y % M
El doble de cada variable
La suma de todas las variables
El producto de todas las variables
Si por ejemplo le hemos dado a X el valor 1, a Y el valor 2, a M el valor 3.2 y a N el valor 4.7 se debe mostrar por pantalla:
Variable X = 1
Variable Y = 2
Variable M = 3.2
Variable N = 4.7
1 + 2 = 3

```
1 - 2 = -1
1 * 2 = 2
1 / 2 = 0
1 % 2 = 1
4.7 + 3.2 = 7.9
4.7 - 3.2 = 1.5
4.7 * 3.2 = 15.040000000000001
4.7 / 3.2 = 1.46875
4.7 % 3.2 = 1.5
1 + 4.7 = 5.7
2 / 3.2 = 0.625
2 % 3.2 = 2.0
Variable X = 1 el doble es 2
Variable Y = 2 el doble es 4
Variable M = 3.2 el doble es 6.4
Variable N = 4.7 el doble es 9.4
1 + 2 + 4.7 + 3.2 = 10.9
1 * 2 * 4.7 * 3.2 = 30.080000000000002

/*
 * Solución Ejercicio Básico Inicial 2
 */

package bi2;

public class Main {

    public static void main(String[] args) {
        int X = 1, Y = 2;
        double M = 3.2, N = 4.7;
        System.out.println("Variable X = " + X);
        System.out.println("Variable Y = " + Y);
        System.out.println("Variable M = " + M);
        System.out.println("Variable N = " + N);
        System.out.println(X + " + " + Y + " = " + (X+Y));
        System.out.println(X + " - " + Y + " = " + (X-Y));
        System.out.println(X + " * " + Y + " = " + X*Y);
        System.out.println(X + " / " + Y + " = " + X/Y);
        System.out.println(X + " % " + Y + " = " + X%Y);
        System.out.println(N + " + " + M + " = " + (N+M));
        System.out.println(N + " - " + M + " = " + (N-M));
```

```
        System.out.println(N + " * " + M + " = " + N*M);
        System.out.println(N + " / " + M + " = " + N/M);
        System.out.println(N + " % " + M + " = " + N%M);
        System.out.println(X + " + " + N + " = " + (X+N));
        System.out.println(Y + " / " + M + " = " + Y/M);
        System.out.println(Y + " % " + M + " = " + Y%M);
        System.out.println("Variable X = " + X + " el doble es " + 2*X);
        System.out.println("Variable Y = " + Y + " el doble es " + 2*Y);
        System.out.println("Variable M = " + M + " el doble es " + 2*M);
        System.out.println("Variable N = " + N + " el doble es " + 2*N);
        System.out.println(X + " + " + Y + " + " + N + " + " + M + " = " + (X+Y+M+N));
        System.out.println(X + " * " + Y + " * " + N + " * " + M + " = " + (X*Y*M*N));
    }
}
```

Segunda entrega de ejercicios java básicos iniciales resueltos.
Relación Nº 2: Ejercicios 3 y 4

Ejercicio básico inicial 3
Escribe un programa Java que declare una variable entera N y asígnale un valor. A continuación escribe las instrucciones que realicen los siguientes:
Incrementar N en 77.
Decrementarla en 3.
Duplicar su valor.
Si por ejemplo N = 1 la salida del programa será:
Valor inicial de N = 1
N + 77 = 78
N - 3 = 75
N * 2 = 150

```
/*
 * Solución Ejercicio Básico Inicial 3
 */
package bi3;
```

```
public class Main {
   public static void main(String[] args) {
      int N = 1;
      System.out.println("Valor inicial de N = " + N);
      N+=77;
      System.out.println("N + 77 = " + N);
      N-=3;
      System.out.println("N - 3 = " + N);
      N*=2;
      System.out.println("N * 2 = " + N);
   }
}
```

Ejercicio básico inicial 4
Programa java que declare cuatro variables enteras A, B, C y D y asígnale un valor a cada una. A continuación realiza las instrucciones necesarias para que:
B tome el valor de C
C tome el valor de A
A tome el valor de D
D tome el valor de B
Si por ejemplo A = 1, B = 2, C = 3 y D = 4 el programa debe mostrar:
Valores iniciales
A = 1
B = 2
C = 3
D = 4
Valores finales
B toma el valor de C -> B = 3
C toma el valor de A -> C = 1
A toma el valor de D -> A = 4
D toma el valor de B -> D = 2

```
/*
 * Solución Ejercicio Básico Inicial 4
 */
package bi6;
public class Main {
   public static void main(String[] args) {
      int A = 1, B = 2, C = 3, D = 4, AUX;
      System.out.println("Valores iniciales");
      System.out.println("A = " + A);
```

```
        System.out.println("B = " + B);
        System.out.println("C = " + C);
        System.out.println("D = " + D);
        AUX = B;
        B = C;
        C = A;
        A = D;
        D = AUX;
        System.out.println("Valores finales");
        System.out.println("B toma el valor de C -> B = " + B);
        System.out.println("C toma el valor de A -> C = " + C);
        System.out.println("A toma el valor de D -> A = " + A);
        System.out.println("D toma el valor de B -> D = " + D);
    }
}
```

Relación Nº 3: Ejercicios 5, 6 y 7

En esta entrada vamos a ver tres ejemplos de utilización del operador condicional ? :
Se trata de usar el operador condicional en lugar de la instrucción condicional if para mostrar por pantalla un mensaje u otro dependiendo de una condición.

Ejercicio básico inicial 5
Escribe un programa java que declare una variable A de tipo entero y asígnale un valor. A continuación muestra un mensaje indicando si A es par o impar. Utiliza el operador condicional (? :) dentro del println para resolverlo.
Si por ejemplo A = 14 la salida será
14 es par
Si fuese por ejemplo A = 15 la salida será:
15 es impar

```
/*
 * Solución Ejercicio Básico Inicial 5
 */
package bi7;
```

```
public class Main {
   public static void main(String[] args) {
      int A = 15;
      System.out.println(A + (A%2==0 ? " es par " : " es impar "));
   }
}
```

Ejercicio básico inicial 6
Escribe un programa java que declare una variable B de tipo entero y asígnale un valor. A continuación muestra un mensaje indicando si el valor de B es positivo o negativo. Consideraremos el 0 como positivo. Utiliza el operador condicional (? :) dentro del println para resolverlo.
Si por ejemplo B = 1 la salida será
1 es positivo
Si fuese por ejemplo B = -1 la salida será:
-1 es negativo

```
/*
 * Solución Ejercicio Básico Inicial 6
 */
package bi6;
public class Main {
   public static void main(String[] args) {
      int B = -1;
      System.out.println(B + (B >= 0 ? " es positivo " : " es negativo "));
   }
}
```

Ejercicio básico inicial 7
Escribe un programa java que declare una variable C de tipo entero y asígnale un valor. A continuación muestra un mensaje indicando si el valor de C es positivo o negativo, si es par o impar, si es múltiplo de 5, si es múltiplo de 10 y si es mayor o menor que 100.
Consideraremos el 0 como positivo. Utiliza el operador condicional (? :) dentro del println para resolverlo.
Si por ejemplo C = 55 la salida será
55 es positivo
55 es impar
55 es múltiplo de 5
55 no es múltiplo de 10
55 es menor que 100

```
/*
 * Solución Ejercicio Básico Inicial 7
 */
package bi7;
public class Main {
    public static void main(String[] args) {
        int C = 55;
        System.out.println(C + (C >= 0 ? " es positivo " : " es negativo "));
        System.out.println(C + (C%2== 0 ? " es par " : " es impar "));
        System.out.println(C + (C%5== 0 ? " es múltiplo de 5 " : " no es múltiplo de 5 "));
        System.out.println(C + (C%10== 0 ? " es múltiplo de 10 " : " no es múltiplo de 10 "));
        System.out.println(C + (C>100 ? " es mayor que 100 " : " es menor que 100 "));
    }
}
```

Java Ejercicios Básicos Resueltos 1

RELACIÓN Nº 1: EJERCICIOS 1, 2 Y 3

Empezaremos por unos ejercicios básicos de programas Java con estructura secuencial, es decir, en estos programas no hay instrucciones condicionales ni repetitivas. En la mayoría de ellos las operaciones a realizar son: lectura de datos por teclado, realizar alguna operación con esos datos y mostrar resultados por pantalla.

EJERCICIOS BÁSICOS RESUELTOS CON ESTRUCTURA SECUENCIAL

1. Programa Java que lea dos números enteros por teclado y los muestre por pantalla.

```
 import java.util.*;
public class Main {
    public static void main(String[] args){
        //declaración de variables
        int n1, n2;
        Scanner sc = new Scanner(System.in);
        //leer el primer número
        System.out.println("Introduce un número entero: ");
```

```
        n1 = sc.nextInt();      //lee un entero por teclado
        //leer el segundo número
        System.out.println("Introduce otro número entero: ");
        n2 = sc.nextInt();      //lee un entero por teclado
        //mostrar resultado
        System.out.println("Ha introducido los números: " + n1 + " y " + n2);

    }
}
```

2. Programa Java que lea un nombre y muestre por pantalla:
“Buenos dias nombre_introducido”

```
 import java.util.*;
 public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String cadena;
        System.out.println("Introduce un nombre: ");
        cadena = sc.nextLine();
        System.out.println("Buenos Días " + cadena);
    }
}
```

3. Escribe un programa Java que lee un número entero por teclado y obtiene y muestra por pantalla el doble y el triple de ese número.

```
import java.util.*;
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int numero;
        System.out.println("Introduce un número entero:");
        numero = sc.nextInt();
        System.out.println("Número introducido: " + numero);
        System.out.println("Doble de " + numero + " -> "+ 2*numero);
        System.out.println("Triple de " + numero + " -> "+ 3*numero);

    }
}
```

Java Ejercicios Básicos estructura secuencial
Relación Nº 2: Ejercicios 4, 5, 6 y 7

Ejercicio 4:
Programa que lea una cantidad de grados centígrados y la pase a grados Fahrenheit.
La fórmula correspondiente para pasar de grados centígrados a fahrenheit es:
F = 32 + (9 * C / 5)

```
import java.util.*;
public class Main {
  public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      double gradosC, gradosF;
      System.out.println("Introduce grados Centígrados:");
      gradosC = sc.nextDouble();
      gradosF = 32 + (9 * gradosC / 5);
      System.out.println(gradosC +" ºC = " + gradosF + " ºF");
  }
}
```

Ejercicio 5. Programa que lee por teclado el valor del radio de una circunferencia y calcula y muestra por pantalla la longitud y el área de la circunferencia.

Longitud de la circunferencia = 2*PI*Radio, Area de la circunferencia = PI*Radio^2

```
import java.util.*;
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        double radio, longitud, area;
        System.out.println("Introduce radio de la circunferencia:");
        radio = sc.nextDouble();
        longitud = 2 * Math.PI * radio;
        area = Math.PI * Math.pow(radio, 2);
        System.out.println("Longitud de la circunferencia -> " + longitud);
        System.out.println("Area de la circunferencia -> " + area);
    }
}
```

Ejercicio 6. Programa que pase una velocidad en Km/h a m/s. La velocidad se lee por teclado.

```
import java.util.*;
public class Main {
  public static void main(String[] args) {

       Scanner sc = new Scanner(System.in);
       double velocidad;
       System.out.println("Introduzca velocidad en Km/h: ");
       velocidad = sc.nextDouble();
     System.out.println(velocidad + " Km/h -> " + velocidad*1000/3600 + " m/s");
}
```

Ejercicio 7. Programa lea la longitud de los catetos de un triángulo rectángulo y calcule la longitud de la hipotenusa según el teorema de Pitágoras.

```
import java.util.*;
public class Main {
   public static void main(String[] args) {
       Scanner sc = new Scanner(System.in);
       double cateto1, cateto2;
       System.out.print("Introduzca longitud del primer cateto: ");
       cateto1 = sc.nextDouble();
       System.out.print("Introduzca longitud del segundo cateto: ");
       cateto2 = sc.nextDouble();
     System.out.println("Hipotenusa -> " +  Math.sqrt(Math.pow(cateto1,2)+ Math.pow
(cateto2, 2)));
   }
}
```

Ejercicio 9:
Programa Java que calcule el área de un triángulo en función de las longitudes de sus lados (a, b, c), según la siguiente fórmula:
Area = RaizCuadrada(p*(p-a)*(p-b)*(p-c))
donde p = (a+b+c)/2
Para calcular la raíz cuadrada se utiliza el método Math.sqrt()

```
/*
 * Programa que calcule el área de un triángulo en función de las longitudes de sus lados (a, b, c)
 * según la siguiente fórmula: area=raiz2(p(p-a)(p-b)(p-c)) donde p = (a+b+c)/2
 */
import java.util.*;
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        double a,b,c,p;
        System.out.print("Introduzca longitud del primer lado del triángulo: ");
        a = sc.nextDouble();
        System.out.print("Introduzca longitud del segundo lado del triángulo: ");
        b = sc.nextDouble();
        System.out.print("Introduzca longitud del tercer lado del triángulo: ");
        c = sc.nextDouble();
        p = (a+b+c)/2;
    System.out.println("Area -> " +  Math.sqrt(p*(p-a)*(p-b)*(p-c)));
    }
}
```

Ejercicio 10:
Programa Java que lea un número entero de 3 cifras y muestre por separado las cifras del número.

```
/*
 * Programa que lea un número de 3 cifras y muestre por pantalla las cifras del número
 */
import java.util.*;
public class Main{
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int N;
```

```
        System.out.print("Introduzca valor de N: ");
        N = sc.nextInt(); //supondremos que el número introducido tiene 3 cifras
        System.out.println("Primera cifra de " + N + " -> " + (N/100));
        System.out.println("Cifra central de " + N + " -> " + (N/10)%10);
        System.out.println("Última cifra  de " + N + " -> " + (N%10));

    }
}
```

Ejercicios básicos resueltos con estructura secuencial
Relación Nº 4: Ejercicios 11, 12 y 13
Ejercicio 11:
Programa que lea un número entero N de 5 cifras y muestre sus cifras igual que en el ejemplo.
Por ejemplo para un número N = 12345 La salida debe ser:
1
12
123
1234
12345

```
/*
 *
 * N = 12345    La salida debe ser:
 * 1
 * 12
 * 123
 * 1234
 * 12345
 */
import java.util.*;
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int N;
        System.out.print("Introduzca valor de N: ");
        N = sc.nextInt(); //supondremos que el número introducido tiene 5 cifras
        System.out.println(N/10000);
        System.out.println(N/1000);
```

```
        System.out.println(N/100);
        System.out.println(N/10);
        System.out.println(N);
    }
}
```

Ejercicio 12:
Programa Java que lea un número entero N de 5 cifras y muestre sus cifras igual que en el ejemplo.
Por ejemplo para un número N = 12345 La salida debe ser:

5
45
345
2345
12345

```
/*
 *
 * N = 12345    La salida debe ser:
 * 5
 * 45
 * 345
 * 2345
 * 12345
 */
import java.util.*;
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int N;
        System.out.print("Introduzca valor de N: ");
        N = sc.nextInt();  //supondremos que el número introducido tiene 5 cifras
        System.out.println(N%10);
        System.out.printf("%02d %n",N%100);
        System.out.printf("%03d %n",N%1000);
        System.out.printf("%04d %n",N%10000);
        System.out.printf("%05d %n",N);
```

```
    }
}
```

Ejercicio 13:
Programa que pida por teclado la fecha de nacimiento de una persona (dia, mes, año) y calcule su número de la suerte.
El número de la suerte se calcula sumando el día, mes y año de la fecha de nacimiento y a continuación sumando las cifras obtenidas en la suma.
Por ejemplo:
Si la fecha de nacimiento es 12/07/1980
Calculamos el número de la suerte así: 12+7+1980 = 1999 1+9+9+9 = 28
Número de la suerte: 28

```
/*
 * Programa que calcula el número de la suerte
 */
import java.util.*;
public class Secuenciales2_13 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int dia, mes, año, suerte, suma, cifra1, cifra2, cifra3, cifra4;
        System.out.println("Introduzca fecha de nacimiento");
        System.out.print("día: ");
        dia = sc.nextInt();
        System.out.print("mes: ");
        mes = sc.nextInt();
        System.out.print("año: ");
        año = sc.nextInt();
        suma = dia + mes + año;
        cifra1 = suma/1000;      //obtiene la primera cifra
        cifra2 = suma/100%10;  //obtiene la segunda cifra
        cifra3 = suma/10%10;   //obtiene la tercera cifra
        cifra4 = suma%10;        //obtiene la última cifra
        suerte = cifra1 + cifra2 + cifra3 + cifra4;
        System.out.println("Su número de la suerte es: " + suerte);
    }
}
```

Ejercicios Básicos con Estructura Condicional

Relación Nº 1: Ejercicios 1, 2 y 3

1. Programa Java que lea un número entero por teclado y calcule si es par o impar.

Podemos saber si un número es par si el resto de dividir el número entre 2 es igual a cero. En caso contrario el número es impar

El operador Java que calcula el resto de la división entre dos números enteros o no es el operador %

El programa que calcula si un número entero es par o impar es el siguiente:

```
import java.util.*;
public class Condicional1_1 {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      int N;
      System.out.print("Introduzca Número entero: ");
      N = sc.nextInt();
      if(N%2==0)
        System.out.println("Par");
      else
         System.out.println("Impar");
   }
}
```

2. Programa que lea un número entero y muestre si el número es múltiplo de 10.

Podemos comprobar si un número entero es múltiplo de 10 si al dividirlo por 10 es resto de

esta división es cero.

```
/* Programa que lea un número entero y muestre si el número es múltiplo de 10 */
import java.util.*;
public class Condicional1_2 {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      int N;
      System.out.print("Número entero: ");
      N = sc.nextInt();
      if(N%10==0)
        System.out.println("Es múltiplo de 10");
      else
       System.out.println("No es múltiplo de 10");
   }
}
```

3. Programa que lea un carácter por teclado y compruebe si es una letra mayúscula

```
/* condicional1_3
 * Programa que lea un carácter por teclado y compruebe si es una letra mayúscula
 */
import java.io.*;
import java.util.*;
public class condicional1_3 {
   public static void main(String[] args) throws IOException{
        Scanner sc = new Scanner(System.in);
        char car, car1;
        System.out.print("Introduzca un carácter: ");
        car = (char)System.in.read(); //lee un solo caracter

        if(Character.isUpperCase(car)) //utilizamos el método isUpperCase de la clase
Character
          System.out.println("Es una letra mayúscula");
        else
           System.out.println("No es una letra mayúscula");
    }
```

Forma alternativa de comprobar si un carácter es una letra mayúscula sin utilizar el método isUpperCase, en este caso comparando directamente con los caracteres A y Z

```
      if(car>='A' && car <='Z')
        System.out.println("Es una letra mayúscula");
      else
        System.out.println("No es una letra mayúscula");
```

Java Ejercicios Básicos Resueltos Estructura Condicional 2

Continuamos con más ejercicios básicos que usan la estructura condicional.
Relación Nº 2: Ejercicios 4 y 5
Ejercicio 4: Programa que lea dos caracteres y compruebe si son iguales.

```
/*
 *  Ejemplo básico java
 * Programa con estructura condicional
 * Programa que lea dos caracteres y compruebe
 * si son iguales.
 */
import java.io.*;
public class condicional1_5 {
   public static void main(String[] args) throws IOException {
      char car1, car2;
      System.out.print("Introduzca primer carácter: ");
      car1 = (char)System.in.read(); //lee un carácter
      System.in.read();  //saltar el intro que ha quedado en el buffer
      System.out.print("Introduzca segundo carácter: ");
      car2 = (char)System.in.read(); //lee el segundo carácter

      if(car1 == car2)
        System.out.println("Son iguales");
      else
        System.out.println("No son iguales");
   }
}
```

Una forma alternativa de hacer este programa es creando dos objetos Character a partir de los

caracteres que se han leído y compararlos utilizando el método equals

```
import java.io.*;
public class condicional1_5 {
   public static void main(String[] args) throws IOException {
      char car1, car2;
      System.out.print("Introduzca primer carácter: ");
      car1 = (char)System.in.read(); //lee un carácter
      System.in.read();  //saltar el intro que ha quedado en el buffer
      System.out.print("Introduzca segundo carácter: ");
      car2 = (char)System.in.read(); //lee el segundo carácter
      Character c1 = new Character(car1);
      Character c2 = new Character(car2);
      if(c1.equals(c2))  //comparamos dos objetos Character mediante el método equals
         System.out.println("Son iguales");
      else
         System.out.println("No son iguales");
   }
}
```

Ejercicio 5: Programa java que lea dos caracteres por teclado y compruebe si los dos son letras minúsculas

```
/* Ejemplo básico java de programa con estructura condicional
 * Programa que lea dos caracteres y compruebe si los dos son letras minúsculas
 */
import java.io.*;
public class condicional1_6 {
public static void main(String[] args) throws IOException {
      char car1, car2;
      System.out.println("Introduzca primer carácter: ");
      car1 = (char)System.in.read(); //lee un carácter
      System.in.read();  //saltar el intro que ha quedado
      System.out.println("Introduzca segundo carácter: ");
      car2 = (char)System.in.read(); //lee el segundo carácter
      if(Character.isLowerCase(car1)){ //utilizamos el método isLowerCase de la clase
Character
        if(Character.isLowerCase(car2))
```

```
            System.out.println("Los dos son letras minúsculas");
          else
            System.out.println("El primero es una letra minúscula pero el segundo no");
        }
        else{
           if(Character.isLowerCase(car2))
            System.out.println("El segundo es una letra minúscula pero el primero no");
          else
            System.out.println("Ninguno es una letra minúscula");
        }
    }
}
```

Una forma alternativa de resolver este ejercicio es comparando directamente las dos variables con las letras minúsculas

```
        if(car1>='a' && car1<='z'){
           if(car2>='a' && car2<='z')
            System.out.println("Los dos son letras minúsculas");
          else
            System.out.println("El primero es una letra minúscula pero el segundo no");
        }
        else{
           if(car2>='a' && car2<='z')
            System.out.println("El segundo es una letra minúscula pero el primero no");
          else
            System.out.println("Ninguno es una letra minúscula");
        }
```

Java Ejercicios Estructura Condicional 3

Aquí teneis otra entrega de ejercicios básicos con estructura condicional.

Relación Nº 3: Ejercicios 6 y 7

Ejercicio 6: Programa java que lea un carácter por teclado y compruebe si es un dígito numérico (cifra entre 0 y 9).

Vamos a escribir dos soluciones a este ejercicio.

La primera consiste en comprobar si el carácter es un dígito mediante el método isDigit de la clase Character. Este método devuelve true si el carácter que se le pasa como parámetro es una cifra entre 0 y 9:

```
/*
  * Ejemplo de programa con estructura condicional
 * Programa que lea un carácter por teclado y compruebe si es un número
 */
import java.io.*;
public class condicional1_7 {
   public static void main(String[] args) throws IOException {
       char car1;
       System.out.print("Introduzca carácter: ");
       car1 = (char)System.in.read(); //lee un carácter
       if(Character.isDigit(car1)) //utilizamos el método isDigit de la clase Character
          System.out.println("Es un número");
       else
          System.out.println("No es un número");
  }
}
```

La otra solución es directa y consiste en comprobar si el carácter que se ha leído por teclado es mayor o igual que el carácter 0 y menor o igual que el carácter 9.

```
// Versión alternativa comparando con
// los caracteres '0' ... '9'

    if(car1>='0' && car1<='9')
       System.out.println("Es un número");
       else
          System.out.println("No es un número");
```

Ejercicio 7: Programa que lea dos números por teclado y muestre el resultado de la división del primer número por el segundo. Se debe comprobar que el divisor no puede ser cero.

```
/*
 * Ejemplo de programa con estructura condicional
```

```
 * Programa que lea dos números por teclado y muestre el resultado
 * de la división del primero por el segundo.
 * Se comprueba que el divisor es distinto de cero.
 */
import java.util.*;
public class condicional1_8 {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      double dividendo, divisor;
      System.out.print("Introduzca el dividendo: ");
      dividendo = sc.nextDouble();
      System.out.print("Introduzca el divisor: ");
      divisor = sc.nextDouble();
      if(divisor==0)
        System.out.println("No se puede dividir por cero");
      else{
         System.out.println(dividendo + " / " + divisor + " = " + dividendo/divisor);
         System.out.printf("%.2f / %.2f = %.2f %n" , dividendo, divisor , dividendo/divisor);
      }
   }
}
```

Mayor de tres numeros

Calcular el mayor de tres números enteros en Java.

El programa lee por teclado tres números enteros y calcula y muestra el mayor de los tres.

```
/*
 * Programa que lee tres números distintos
 * y nos dice cuál de ellos es el mayor
 */
import java.util.*;
public class MayorDeTres {
 public static void main(String[] args) {
```

```
        Scanner sc = new Scanner(System.in);
        int n1, n2, n3;
        System.out.print("Introduzca primer número: ");
        n1 = sc.nextInt();
        System.out.print("Introduzca segundo número: ");
        n2 = sc.nextInt();
        System.out.print("Introduzca tercer número: ");
        n3 = sc.nextInt();
        if (n1 > n2) {
            if (n1 > n3) {
                System.out.println("El mayor es: " + n1);
            } else {
                System.out.println("el mayor es: " + n3);
            }
        } else if (n2 > n3) {
            System.out.println("el mayor es: " + n2);
        } else {
            System.out.println("el mayor es: " + n3);
        }
    }
}
```

Java Ejercicios Básicos Condicional 4

Relación Nº 4: Ejercicios 9 y 10

Ejercicio 9:

Programa que lea por teclado tres números enteros H, M, S correspondientes a hora, minutos y segundos respectivamente, y comprueba si la hora que indican es una hora válida.

Supondremos que se leemos una hora en modo 24 Horas, es decir, el valor válido para las horas será mayor o igual que cero y menor que 24.

El valor válido para los minutos y segundos estará comprendido entre 0 y 59 ambos incluidos

```
/*
 * Programa java que lea tres números enteros H, M, S que contienen hora, minutos y
 * segundos respectivamente, y comprueba si la hora que indican es una hora válida.
 */
import java.util.*;
public class condicional1_16 {
   public static void main(String[] args) {
      int H,M,S;
      Scanner sc = new Scanner(System.in);
      System.out.print("Introduzca hora: ");
      H =  sc.nextInt();
      System.out.print("Introduzca minutos: ");
      M =  sc.nextInt();
      System.out.print("Introduzca segundos: ");
      S =  sc.nextInt();
      if(H>=0 && H<24 && M>=0 && M<60 && S>=0 && S<60)
        System.out.println("Hora correcta");
      else
         System.out.println("Hora incorrecta");
   }
}
```

La instrucción que comprueba si la hora leída por teclado es correcta es:

```
if(H>=0 && H<24 && M>=0 && M<60 && S>=0 && S<60)
```

Esta condición da como resultado true cuando la hora es mayor o igual a 0 y menor que 24, los minutos son mayores o iguales a 0 y menores que 60 y los segundos son mayores 0 iguales a cero y menores a 60.

Ejercicio 10:

Programa que lea una variable entera mes y compruebe si el valor corresponde a un mes de

30 días, de 31 o de 28. Supondremos que febrero tiene 28 días. Se mostrará además el nombre del mes. Se debe comprobar que el valor introducido esté comprendido entre 1 y 12.

```
/*
 * Programa java que lea una variable entera mes y compruebe si el valor corresponde
 * a un mes de 30 días. Se debe comprobar que el valor introducido esté
 * comprendido entre 1 y 12
 */
import java.util.*;
public class condicional1_17 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int mes;
        System.out.print("Introduzca número de mes: ");
        mes =  sc.nextInt();
        if(mes < 1 || mes > 12) //se comprueba que el valor del mes es correcto
            System.out.println("Mes incorrecto");
        else{  //si el mes es correcto
            switch(mes){  //se muestra el nombre mediante una instrucción switch
                case 1: System.out.print("Enero");
                        break;
                case 2: System.out.print("Febrero");
                        break;
                case 3: System.out.print("Marzo");
                        break;
                case 4: System.out.print("Abril");
                        break;
                case 5: System.out.print("Mayo");
                        break;
                case 6: System.out.print("Junio");
                        break;
                case 7: System.out.print("Julio");
                        break;
                case 8: System.out.print("Agosto");
                        break;
                case 9: System.out.print("Septiembre");
                        break;
                case 10: System.out.print("Octubre");
                        break;
                case 11: System.out.print("Noviembre");
                        break;
```

```
            case 12: System.out.print("Diciembre");
                    break;
          }
          // mostrar si es un mes de 30, 31 0 28 días
          if(mes == 4 || mes == 6 || mes == 9 || mes == 11)
            System.out.println(" es un mes de 30 días");
          else if(mes == 2)
                 System.out.println(" es un mes de 28 días");
               else
                 System.out.println(" es un mes de 31 días");
       }
    }
}
```

Estructura Repetitiva en Java. Ejercicios Básicos Resueltos 1

Relación Nº 1: Ejercicios 1, 2, 3, 4, 5 y 6
Ejercicios básicos que utilizan la estructura repetitiva.

Se trata de mostrar los números desde el 1 hasta el 100 utilizando las instrucciones repetitivas while, do while y for.

1. Ejemplo de uso de while: Programa Java que muestre los números del 1 al 100 utilizando la instrucción while.

```
/*
 * Ejercicios básicos java con estructura iterativa o repetitiva
 * Mostrar los números del 1 al 100
 * utilizando un bucle while
 */
public class Main {
    public static void main(String[] args) {
```

```
        System.out.println("Numeros del 1 al 100: ");
        int i=1;
        while(i<=100) {
                System.out.println(i);
                i++;
        }
    }
}
```

2. Ejemplo de uso de do-while. Programa Java que muestre los números del 1 al 100 utilizando la instrucción do..while.

```
/*
 * Ejercicios básicos java con estructura iterativa o repetitiva
 * Mostrar los números del 1 al 100 utilizando un bucle do while
 */
public class Main {
    public static void main(String[] args) {
        int i = 1;
        System.out.println("Numeros del 1 al 100: ");
        do{
                System.out.println(i);
                i++;
        }while(i<=100);
    }
}
```

3. Ejemplo de uso de for. Programa Java que muestre los números del 1 al 100 utilizando la instrucción for.

```
/*
 * Ejercicios básicos java con estructura iterativa o repetitiva
 * Mostrar los números del 1 al 100 utilizando un bucle for
 */
public class Repetitiva1_3 {
    public static void main(String[] args) {
        System.out.println("Numeros del 1 al 100: ");
        for(int i = 1; i<=100;i++)
            System.out.println(i);
    }
}
```

4. Ejemplo de uso de while. Programa Java que muestre los números del 100 al 1 utilizando la instrucción while.

```
/*
 * Ejercicios básicos java con estructura iterativa o repetitiva
 * Mostrar los números del 100 al 1 utilizando un bucle while
 */
public class Main {
    public static void main(String[] args) {
        System.out.println("Numeros del 100 al 1: ");
         int i=100;
         while(i>=1)
         {
                 System.out.println(i);
                 i--;
         }
    }
}
```

5. Ejemplo de uso de do-while. Programa Java que muestre los números del 100 al 1 utilizando la instrucción do..while.

```
/*
 * Ejercicios básicos java con estructura iterativa o repetitiva
 * Mostrar los números del 100 al 1 utilizando un bucle do while
 */
public class Main {
    public static void main(String[] args) {
        int i = 100;
        System.out.println("Numeros del 100 al 1: ");
         do{
                 System.out.println(i);
                 i--;
        }while(i>=1);
    }
}
```

6. Ejemplo de for. Programa Java que muestre los números del 100 al 1 utilizando la instrucción for.

```
/*
```

```
 * Ejercicios básicos java con estructura iterativa o repetitiva
 * Mostrar los números del 100 al 1 utilizando un bucle for
 */
public class Repetitiva1_6 {
   public static void main(String[] args) {
      System.out.println("Numeros del 100 al 1: ");
      for(int i=100;i>=1;i--)
         System.out.println(i);
   }
}
```

Intercambiar el contenido de dos variables en Java

En esta entrada vamos a explicar dos métodos para realizar el intercambio de valores entre dos variables.
El primer método de intercambio utiliza una variable auxiliar y el segundo método realiza el intercambio de valores sin utilizar variable auxiliar.
Intercambio de valores entre dos variables utilizando una variable auxiliar.

Programa para intercambiar el valor de dos variables. Los valores iniciales se leen por teclado.

Por ejemplo, suponiendo que las variables se llaman A y B, si A contiene 3 y B contiene 5, después del intercambio A contendrá 5 y B 3.

En este ejemplo, para intercambiar el valor entre dos variables utilizaremos una avariable auxiliar donde guardar el valor de una de ellas. Después veremos la forma de hacerlo sin usar una variable auxiliar para el intercambio.

Las instrucciones a realizar son:

```
AUX = A;
A = B;
B = AUX;
```

Programa completo:

```
/*
 * Programa que lea dos variables
 * numéricas A y B e
 * intercambie sus contenidos.
 */
import java.util.*;
public class Secuenciales2_5 {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      int A, B, AUX;
      System.out.print("Introduzca valor de A: ");
      A = sc.nextInt();
      System.out.print("Introduzca Valor de B: ");
      B = sc.nextInt();
      System.out.println("Valores iniciales: A = " + A + "   B = " + B);
      //instrucciones para hacer el intercambio
      //se utiliza una variable auxiliar
      AUX = A;
      A = B;
      B = AUX;
      System.out.println("Valores intercambiados: A = " + A + "   B = " + B);
   }
}
```

Intercambio de valores entre dos variables sin utilizar variable auxiliar.
También se puede intercambiar el valor de dos variables sin utilizar una variable auxiliar.
En ese caso se resuelve utilizando aritmética básica:

```
A = A + B;
B = A - B;
A = A - B;
```

Si por ejemplo A = 5 y B = 3

A = 5 + 3 = 8
B = 8 - 3 = 5

A = 8 - 5 = 3

Programa completo:

```
/*
 * Programa que intercambie dos variables sin auxiliar
 */
import java.util.*;
public class Secuenciales2_5 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int A, B, AUX;
        System.out.print("Introduzca valor de A: ");
        A = sc.nextInt();
        System.out.print("Introduzca Valor de B: ");
        B = sc.nextInt();
        System.out.println("Valores iniciales: A = " + A + "   B = " + B);
        //instrucciones para hacer el intercambio
        //sin utilizar una variable auxiliar
        A = A + B;
        B = A - B;
        A = A - B;
        System.out.println("Valores intercambiados: A = " + A + "   B = " + B);
    }
}
```

Contar las cifras de un número entero en Java

Programa Java que pide un número entero por teclado y calcula y muestra el número de cifras que tiene.

Por ejemplo si se introduce el número 54391 el programa mostrará el mensaje:

El número tiene 5 cifras

Si se introduce el número 101 se mostrará el mensaje:

El número tiene 3 cifras

El proceso leer un número y contar sus cifras se repetirá hasta que se conteste ‘N’ a la pregunta Continuar? (S/N)
Si se responde 'S' se volverá a pedir otro número.

Para saber cuántas cifras tiene un número entero haremos lo siguiente:

Dividiremos el número sucesivamente entre 10. En cada división tomaremos la parte entera y volvemos a dividir. Este proceso se repite hasta que se obtenga un cero como cociente en la división.

Por ejemplo, si el número leído es 1234 haremos las siguientes operaciones:

1234 / 10 = 123
123 / 10 = 12
12 / 10 = 1
1 / 10 = 0

hemos realizado 4 divisiones hasta obtener un cero como cociente, por lo tanto el número tiene 4 cifras.

```
// Programa Java que calcula el número de cifras que tiene un número entero
import java.util.*;
import java.io.*;

public class Main {
   public static void main(String[] args) throws IOException{
      Scanner sc = new Scanner(System.in);
      int n, cifras;
      char car;
      do{
         System.out.print("Introduce un número entero: ");
         n = sc.nextInt();
```

```
            cifras= 0;    //esta variable es el contador de cifras
            while(n!=0){            //mientras a n le queden cifras
                  n = n/10;         //le quitamos el último dígito
                 cifras++;          //sumamos 1 al contador de cifras
            }
            System.out.println("El número tiene " + cifras+ " cifras");
            System.out.print("Continuar? ");
            car = (char)System.in.read();
         }while(car!='n' && car != 'N');
      }
}
```

Pasar de grados centígrados a kelvin en Java

Programa Java que lee una temperatura expresada en grados centígrados y la convierte a grados kelvin.

El proceso de leer grados centígrados se debe repetir mientras que se responda ‘S’ a la pregunta: Repetir proceso? (S/N)

Para hacer la conversión de grados Centígrados a grados Kelvin hay que utilizar la fórmula:

ºK = ºC + 273

El programa java para realizar la conversión de temperaturas es el siguiente:

```
import java.util.*;
import java.io.*;
/**
 * Programa que lee una temperatura expresada en grados centígrados y los pasa a grados kelvin.
 * Repetir el proceso mientras que se responda ‘S’ a la pregunta:
 * Repetir proceso? (S/N)
```

```
 * @author Enrique
 */
public class CentigradosAKelvin {
    public static void main(String[] args) throws IOException{
        Scanner sc = new Scanner(System.in);
        double temperatura;
        char car;
        do{
            System.out.print("Introduce temperatura en ºC: ");
            temperatura = sc.nextDouble();
            System.out.println("Grados Kelvin ..: " + (temperatura+273));
            System.out.print("Repetir proceso? (S/N): " );
            car = (char)System.in.read();
        }while(car =='S' || car == 's');
    }
}
```

Mostrar la tabla de multiplicar de un número en Java

Programa Java que lea un número entero N y muestre la tabla de multiplicar de ese número. Por ejemplo, si se lee el valor 7 se mostrará por pantalla:

```
Tabla del 7
--------------
7 * 1  =  7
7 * 2  = 14
7 * 3  = 21
7 * 4  = 28
7 * 5  = 35
7 * 6  = 42
```

```
7 * 7  = 49
7 * 8  = 56
7 * 9  = 63
7 * 10 = 70
```

```
import java.util.*;
/**
 * Programa que lea un número entero N y muestre la tabla de multiplicar de ese número.
 * @author Enrique
 */
public class TablaMultiplicar {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n;
        System.out.print("Introduce un número entero: ");
        n = sc.nextInt();
        System.out.println("Tabla del " + n);
        for(int i = 1; i<=10; i++){
            System.out.println(n + " * " + i + " = " + n*i);
        }
    }
}
```

Contar números acabados en 2

Programa que lea una serie de números por teclado hasta que se lea un número negativo. El programa indicará cuántos números acabados en 2 se han leído.

Para saber si un número acaba en dos o en general para saber en qué dígito termina un número entero se divide el número entre 10 y se obtiene el resto de esta división.
En Java el operador que obtiene el resto de una división es el operador %
En este caso para saber si el número acaba en 2 escribiremos la instrucción:
if(n%10==2)

```
import java.util.*;
/**
```

```
 * Programa que lea una serie de números por teclado hasta que
 * se lea un número negativo. El programa indicará cuántos números
 * acabados en 2 se han leído.
 * @author Enrique
 */
public class AcabadosEn2 {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      int n, contador=0;
      System.out.print("Introduce un número entero: ");
      n = sc.nextInt();
      while(n>=0){
          if(n%10==2)  //Si el número acaba en dos
             contador++;       //esta variable contendrá cuántos números acabados en 2 se han
leído.
          System.out.print("Introduce un número entero: ");
          n = sc.nextInt();
   }
      System.out.println("Se han introducido " + contador + " números acabados en 2");
   }
}
```

Más ejercicios básicos resueltos

Si te ha sido útil compártelo

11 comentarios:

Daniel Arocha26 de febrero de 2014, 21:07
amigo falta el comando "break;" al final para que no se cree un bucle infinito

Responder
Respuestas

Enrique27 de febrero de 2014, 0:43
Daniel no es necesario poner un break, el while se repite hasta que se introduzca un número negativo

Número perfecto en java

Qué es un número perfecto?

Un número es perfecto si es igual a la suma de todos sus divisores positivos sin incluir el propio número.

Por ejemplo, el número 6 es perfecto.

El 6 tiene como divisores: 1, 2, 3 y 6 pero el 6 no se cuenta como divisor para comprobar si es perfecto.

Si sumamos 1 + 2 + 3 = 6

Los siguientes números perfectos después del 6 son 28, 496, 8128, 33550336, 8589869056.

En esta entrada vamos a desarrollar el algoritmo para comprobar si un número es perfecto.
El programa pide por teclado un número y muestra si es perfecto o no. mediante un bucle for sumaremos los divisores del número. Al final si esta suma es igual al número mostraremos el mensaje correspondiente.

Programa java para calcular si un número es perfecto:

```
import java.util.Scanner;
public class NumeroPerfecto {
   public static void main(String[] args) {
      int i, suma = 0, n;
      Scanner sc = new Scanner(System.in);
      System.out.println("Introduce un número: ");
      n = sc.nextInt();
      for (i = 1; i < n; i++) {  // i son los divisores. Se divide desde 1 hasta n-1
         if (n % i == 0) {
            suma = suma + i;     // si es divisor se suma
```

```
            }
        }
        if (suma == n) {  // si el numero es igual a la suma de sus divisores es perfecto
            System.out.println("Perfecto");
        } else {
            System.out.println("No es perfecto");

        }
    }
}
```

Utilizando el algoritmo anterior vamos a escribir ahora el programa Java que muestre los números perfectos entre 1 y 1000

```
public class NumerosPerfectos1a1000 {
    public static void main(String[] args) {
        int i, j, suma;
        System.out.println("Números perfectos entre 1 y 1000: ");
        for(i=1;i<=1000;i++){    // i es el número que vamos a comprobar
            suma=0;
            for(j=1;j<i;j++){  // j son los divisores. Se divide desde 1 hasta i-1
                if(i%j==0){
                    suma=suma+j;     // si es divisor se suma
                }
            }
          if(i==suma){          // si el numero es igual a la suma de sus divisores es perfecto
                System.out.println(i);
             }
        }
    }
}
```

Si te ha sido útil compártelo

Números amigos en Java

COMPROBAR SI DOS NÚMEROS SON AMIGOS

Dos números enteros positivos A y B son números amigos si la suma de los divisores propios de A es igual a B y la suma de los divisores propios de B es igual a A.

Los divisores propios de un número incluyen la unidad pero no el propio número.

Un ejemplo de números amigos son los números 220 y 284.
Los divisores propios de 220 son 1, 2, 4, 5, 10, 11, 20, 22, 44, 55 y 110.
La suma de los divisores propios de 220 da como resultado 284
Los divisores propios de 284 son 1, 2, 4, 71 y 142.
La suma de los divisores propios de 284 da como resultado 220.
Por lo tanto 220 y 284 son amigos.

Otras parejas de números amigos son:

1184, 1210
2620, 2924
5020, 5564
6232, 6368
10744, 10856
12285, 14595
17296, 18416

Vamos a escribir el programa que calcula si dos números son amigos:

```
import java.util.*;
/**
 * Programa Java que determina si dos números son amigos.
 * Dos números son amigos si la suma de los divisores propios del primero
 * es igual al segundo y viceversa.
```

```
 */
public class NumerosAmigos {
   public static void main(String[] args) {
      int i,suma=0, n1, n2;
      Scanner sc = new Scanner(System.in);
      System.out.print("Introduce primer número: ");
      n1 = sc.nextInt();
      System.out.print("Introduce segundo número: ");
      n2 = sc.nextInt();
      for(i=1;i<n1;i++){  // for para sumar todos los divisores propios de n1
          if(n1%i==0){
            suma=suma+i;
          }
      }
      // si la suma de los divisores de n1 es igual a n2
      if(suma==n2){
        suma=0;
        for(i=1;i<n2;i++){  // sumo los divisores propios de n2
            if(n2%i==0){
              suma=suma+i;
            }
        }
      // si la suma de los divisores de n2 es igual a n1
        if(suma==n1){
          System.out.println("Son Amigos");
        }else{
              System.out.println("No son amigos");
        }
      }
      else{
            System.out.println("No son amigos");
      }
   }
}
```

Fibonacci en java

La serie de fibonacci la forman una serie de números tales que:

El primer término de la serie es el número 1
El segundo término de la serie es el número 1
Los siguientes términos de la serie de fibonacci se obtienen de la suma de los dos anteriores:

1, 1, 2, 3, 5, 8, 13, ...

Vamos a escribir el programa java que muestra los N primeros números de la serie. El valor de N se lee por teclado.

```
import java.util.*;
/**
 * Serie de Fibonacci en Java
 * Programa que imprima los N primeros números de la serie de Fibonacci.
 * El primer número de la serie es 1, el segundo número es 1 y cada uno de los
 * siguientes es la suma de los dos anteriores.
 * 1, 1, 2, 3, 5, 8, 13,  ..... , N
 * @author Enrique
 */
public class Main{
   public static void main(String[] args){
      Scanner sc = new Scanner(System.in);
      int numero,fibo1,fibo2,i;
      do{
         System.out.print("Introduce numero mayor que 1: ");
         numero = sc.nextInt();
      }while(numero<=1);
      System.out.println("Los " + numero + " primeros términos de la serie de Fibonacci
son:");

      fibo1=1;
      fibo2=1;
```

```
        System.out.print(fibo1 + " ");
        for(i=2;i<=numero;i++){
            System.out.print(fibo2 + " ");
            fibo2 = fibo1 + fibo2;
            fibo1 = fibo2 - fibo1;
        }
        System.out.println();
    }
}
```

Decimal a binario en java

En esta entrada vamos a escribir el programa java para convertir un número de decimal a binario.

Para escribir el programa nos vamos a basar en la forma clásica de pasar de decimal a binario, o sea, dividir el número entre 2 y quedarnos con el resto de la división. Esta cifra, que será un cero o un uno, es el dígito de menos peso (más a la derecha) del número binario. A continuación volvemos a dividir el cociente que hemos obtenido entre 2 y nos quedamos con el resto de la división. Esta cifra será la segunda por la derecha del número binario. Esta operación se repite hasta que obtengamos un cero como cociente.

De forma gráfica lo vamos a ver mucho más claro:

Si queremo convertir el número 12 en binario haremos las siguientes operaciones:

El número 12 en decimal es el 1100 en binario. El número binario se obtiene tomando los

restos en orden inverso a como se han obtenido.

Los que ya sabéis algo de Java podeis pensar que para qué quiero hacer ese programa si simplemente escribiendo la instrucción:

System.out.println(Integer.toBinaryString(numero));

se mostrará el número en binario.
El método toBinaryString de la clase Integer ya me hace el trabajo, pero se trata de que seamos capaces de desarrollar por nosotros mismos el algoritmo que realiza la conversión de decimal a binario.

Este ejercicio se suele plantear cuando se está comenzando a aprender las estructuras repetitivas (while, for, do while) y aún no se conocen los arrays por lo que la solución que se plantea no utiliza arrays y por tanto esta solución aunque es correcta solo es válida para números enteros relativamente pequeños.

```
/**
 * Programa que pasa un número
 * de decimal a binario
 * @author Enrique García
 */
public class Main{

    public static void main(String[] args) {

        int numero, exp, digito;
        double binario;
        Scanner sc = new Scanner(System.in);

        do{
            System.out.print("Introduce un numero entero >= 0: ");
            numero = sc.nextInt();
        }while(numero<0);

        exp=0;
        binario=0;
        while(numero!=0){
```

```
            digito = numero % 2;
            binario = binario + digito * Math.pow(10, exp);
            exp++;
            numero = numero/2;
        }
        System.out.printf("Binario: %.0f %n", binario);
    }
}
```

Pasar un número de binario a decimal en Java

CONVERTIR UN NÚMERO DE BINARIO A DECIMAL EN JAVA

El programa para pasar un número expresado en binario a decimal se basa en la forma tradicional de hacerlo. Los dígitos del número binario ocupan una posición que se numera de derecha a izquierda empezando por cero. La posición del dígito más a la derecha es la 0.

Numero Binario:
1
1
0
1
0
1

Posición que ocupa cada dígito
5
4
3
2
1
0

Para pasar el número a decimal se multiplica cada dígito binario por 2 elevado a la posición que ocupa. La suma de todos los productos es el equivalente en decimal.

```
/*
```

```
* Programa Java que convierte un número binario a decimal
*/
import java.util.Scanner;

public class BinarioDecimal {

  public static void main(String[] args) {
    long numero, aux, digito, decimal;
     int exponente;
     boolean esBinario;
     Scanner sc = new Scanner(System.in);

    //Leer un número por teclado y comprobar que es binario
     do {
         System.out.print("Introduce un numero binario: ");
         numero = sc.nextLong();
         //comprobamos que sea un número binario es decir
         //que este formado solo por ceros y unos
         esBinario = true;
         aux = numero;
         while (aux != 0) {
                 digito = aux % 10; //última cifra del números
                 if (digito != 0 && digito != 1) { //si no es 0 ó 1
                     esBinario = false; //no es un número binario
                 }
                 aux = aux / 10; //quitamos la última cifra para repetir el proceso
          }
     } while (!esBinario); //se vuelve a pedir si no es binario

     //proceso para pasar de binario a decimal
     exponente = 0;
     decimal = 0; //será el equivalente en base decimal
     while (numero != 0) {
             //se toma la última cifra
             digito = numero % 10;
             //se multiplica por la potencia de 2 correspondiente y se suma al número
             decimal = decimal + digito * (int) Math.pow(2, exponente);
             //se aumenta el exponente
             exponente++;
             //se quita la última cifra para repetir el proceso
             numero = numero / 10;
```

```
    }
    System.out.println("Decimal: " + decimal);
  }
}
```

Convertir a números romanos en Java

Programa Java para convertir un número entero a números romanos.
El programa pide un número entre 1 y 3999 y calcula su equivalente en números romanos. Se utiliza un método llamado convertirANumerosRomanos que recibe el número N a convertir de tipo int y devuelve un String con el equivalente en números romanos.
Para convertirlo se obtiene por separado cada cifra del número y se muestran las combinaciones de letras del número romano equivalentes a cada cifra del número original.
Este método no utiliza arrays de modo que este programa se puede resolver sin haber estudiado aún los arrays.

```
//Programa Java que lee un número y lo convierte a números romanos
import java.util.Scanner;

public class NumerosRomanos {

  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    int N;
    do {
        System.out.print("Introduce un número entre 1 y 3999: ");
        N = sc.nextInt();
    } while (N < 1 || N > 3999);
    System.out.println(N + " en numeros romanos -> " + convertirANumerosRomanos(N));
  }

  //método para pasar a números romanos
  public static String convertirANumerosRomanos(int numero) {
    int i, miles, centenas, decenas, unidades;
    String romano = "";
```

```
//obtenemos cada cifra del número
 miles = numero / 1000;
 centenas = numero / 100 % 10;
 decenas = numero / 10 % 10;
 unidades = numero % 10;

//millar
 for (i = 1; i <= miles; i++) {
       romano = romano + "M";
 }

//centenas
 if (centenas == 9) {
    romano = romano + "CM";
 } else if (centenas >= 5) {
              romano = romano + "D";
              for (i = 6; i <= centenas; i++) {
                    romano = romano + "C";
              }
 } else if (centenas == 4) {
               romano = romano + "CD";
 } else {
           for (i = 1; i <= centenas; i++) {
                romano = romano + "C";
           }
 }

//decenas
 if (decenas == 9) {
     romano = romano + "XC";
 } else if (decenas >= 5) {
               romano = romano + "L";
               for (i = 6; i <= decenas; i++) {
                    romano = romano + "X";
               }
 } else if (decenas == 4) {
               romano = romano + "XL";
 } else {
             for (i = 1; i <= decenas; i++) {
                   romano = romano + "X";
             }
```

```
    }

    //unidades
    if (unidades == 9) {
        romano = romano + "IX";
    } else if (unidades >= 5) {
                romano = romano + "V";
                for (i = 6; i <= unidades; i++) {
                      romano = romano + "I";
                }
    } else if (unidades == 4) {
                romano = romano + "IV";
    } else {
              for (i = 1; i <= unidades; i++) {
                   romano = romano + "I";
              }
    }
    return romano;
  }
}
```

Cifrado César en Java

Programa para codificar o decodificar un texto utilizando el método de cifrado de César. Supondremos que el texto solo contiene letras mayúsculas o minúsculas. La letras serán las correspondientes al alfabeto inglés (26 caracteres, excluimos la ñ y Ñ).
En este método de cifrado cada letra del texto se sustituye por otra letra que se encuentra N posiciones adelante en el alfabeto. Se considera que el alfabeto es circular, es decir, la letra siguiente a la ‘z’ es la ‘a’.
Por ejemplo, si N es 3, la ‘a’ se transformaría en ‘d’, la ‘b’ en ‘e’, la ‘c’ en ‘f’, etc.

Ejemplo de cifrado César: si el texto es “casa” y N = 3 el texto cifrado es “fdvd”

Para descifrar un texto se realiza la operación contraria. Se calcula la letra que está N posiciones por detrás en el alfabeto. Como el alfabeto es circular, la letra anterior a la ‘a’ es la ‘z’.

El programa pedirá por teclado un texto, a continuación el valor de N y si queremos codificar o decodificar el texto. Finalmente se mostrará el texto resultante.

Programa resuelto: Cifrado César en Java

```
import java.io.IOException;
import java.util.Scanner;

public class CifradoCesar {

    public static void main(String[] args) throws IOException {
        Scanner sc = new Scanner(System.in);
        String texto;
        int codigo;
        char opcion;
        //Introducir un texto
        do {
            System.out.print("Introduce un texto: ");
            texto = sc.nextLine();
        } while (texto.isEmpty());
        //Introducir el valor del desplazamiento
        do {
            System.out.print("Introduce el código: ");
            codigo = sc.nextInt();
        } while (codigo < 1);
        //Introducir la operación a realizar: cifrar o descifrar
        do {
            sc.nextLine();
            System.out.print("(C) cifrar o (D) descifrar?: ");
            opcion = (char) System.in.read();
        } while (Character.toUpperCase(opcion) != 'C' && Character.toUpperCase(opcion) != 
'D');
        if (Character.toUpperCase(opcion) == 'C') {
            System.out.println("Texto cifrado: " + cifradoCesar(texto, codigo));
        } else {
```

```
        System.out.println("Texto descifrado: " + descifradoCesar(texto, codigo));
    }
}

//método para cifrar el texto
public static String cifradoCesar(String texto, int codigo) {
    StringBuilder cifrado = new StringBuilder();
    codigo = codigo % 26;
    for (int i = 0; i < texto.length(); i++) {
        if (texto.charAt(i) >= 'a' && texto.charAt(i) <= 'z') {
            if ((texto.charAt(i) + codigo) > 'z') {
                cifrado.append((char) (texto.charAt(i) + codigo - 26));
            } else {
                cifrado.append((char) (texto.charAt(i) + codigo));
            }
        } else if (texto.charAt(i) >= 'A' && texto.charAt(i) <= 'Z') {
            if ((texto.charAt(i) + codigo) > 'Z') {
                cifrado.append((char) (texto.charAt(i) + codigo - 26));
            } else {
                cifrado.append((char) (texto.charAt(i) + codigo));
            }
        }
    }
    return cifrado.toString();
}

//método para descifrar el texto
public static String descifradoCesar(String texto, int codigo) {
    StringBuilder cifrado = new StringBuilder();
    codigo = codigo % 26;
    for (int i = 0; i < texto.length(); i++) {
        if (texto.charAt(i) >= 'a' && texto.charAt(i) <= 'z') {
            if ((texto.charAt(i) - codigo) < 'a') {
                cifrado.append((char) (texto.charAt(i) - codigo + 26));
            } else {
                cifrado.append((char) (texto.charAt(i) - codigo));
            }
        } else if (texto.charAt(i) >= 'A' && texto.charAt(i) <= 'Z') {
            if ((texto.charAt(i) - codigo) < 'A') {
                cifrado.append((char) (texto.charAt(i) - codigo + 26));
            } else {
```

```
                cifrado.append((char) (texto.charAt(i) - codigo));
            }
        }
    }
    return cifrado.toString();
  }
} //Fin cifrado Cesar
```

Ejemplos de ejecución:

Introduce un texto: Tengo el examen resuelto
Introduce el código: 4
(C) cifrar o (D) descifrar?: C
Texto cifrado: Xirksipibeqirviwyipxs

Introduce un texto: glgtekekqutguwgnvqu
Introduce el código: 2
(C) cifrar o (D) descifrar?: D
Texto descifrado: ejerciciosresueltos

Número capicúa en Java

COMPROBAR SI UN NÚMERO ES CAPICÚA EN JAVA

Un número es capicúa si se puede leer igual de derecha a izquierda que de izquierda a derecha. Ejemplos de números capicúas: 121, 3003, 1234321, 33, 445544, etc.

Vamos a escribir un programa Java que pida por teclado un número entero N de más de una cifra y verifique si es capicúa.

```
import java.util.Scanner;

public class Numero_Capicua {

   public static void main(String[] args) {
      int N, aux, inverso = 0, cifra;
      Scanner sc = new Scanner(System.in);
      do {
         System.out.print("Introduce un número >= 10: ");
```

```
            N = sc.nextInt();
        } while (N < 10);

        //le damos la vuelta al número
        aux = N;
        while (aux!=0){
            cifra = aux % 10;
            inverso = inverso * 10 + cifra;
            aux = aux / 10;
        }

        if(N == inverso){
            System.out.println("Es capicua");
        }else{
            System.out.println("No es capicua");
        }
    }
}
```

Para resolverlo, lo que hemos hecho es darle la vuelta al número y comprobar si el número invertido es igual al original.
El procedimiento para obtener el número invertido, al que hemos llamado inverso, es el siguiente:
Si por ejemplo N = 121
inverso = 0
Repetimos estos pasos hasta que N valga 0:
Obtenemos la última cifra de N: cifra = N % 10 -> cifra = 121 % 10 -> cifra = 1
Se lo sumamos al contenido de inverso multiplicado por 10
inverso = inverso * 10 + cifra -> inverso = 0 * 10 + 1 -> inverso = 1
Quitamos la última cifra a N: N = N / 10 -> N = 121 / 10 -> N = 12
Repetimos el proceso:
Obtenemos la última cifra de N: cifra = N % 10 -> cifra = 12 % 10 -> cifra = 2
Se lo sumamos al contenido de inverso multiplicado por 10
inverso = inverso * 10 + cifra -> inverso = 1 * 10 + 2 -> inverso = 12
Quitamos la última cifra a N: N = N / 10 -> N = 12 / 10 -> N = 1
Repetimos el proceso:
Obtenemos la última cifra de N: cifra = N % 10 -> cifra = 1 % 10 -> cifra = 1
Se lo sumamos al contenido de inverso multiplicado por 10
inverso = inverso * 10 + cifra -> inverso = 12 * 10 + 1 -> inverso = 121

Quitamos la última cifra a N: N = N / 10 -> N = 1 / 10 -> N = 0

El proceso para invertir el número finaliza cuando N = 0. Ahora comprobamos si el número invertido es igual al original. En ese caso el número es capicúa.

Número perfecto en java

Qué es un número perfecto?

Un número es perfecto si es igual a la suma de todos sus divisores positivos sin incluir el propio número.

Por ejemplo, el número 6 es perfecto.

El 6 tiene como divisores: 1, 2, 3 y 6 pero el 6 no se cuenta como divisor para comprobar si es perfecto.

Si sumamos 1 + 2 + 3 = 6

Los siguientes números perfectos después del 6 son 28, 496, 8128, 33550336, 8589869056.

En esta entrada vamos a desarrollar el algoritmo para comprobar si un número es perfecto.
El programa pide por teclado un número y muestra si es perfecto o no. mediante un bucle for sumaremos los divisores del número. Al final si esta suma es igual al número mostraremos el mensaje correspondiente.

Programa java para calcular si un número es perfecto:

```
import java.util.Scanner;
public class NumeroPerfecto {
   public static void main(String[] args) {
      int i, suma = 0, n;
      Scanner sc = new Scanner(System.in);
      System.out.println("Introduce un número: ");
      n = sc.nextInt();
      for (i = 1; i < n; i++) {  // i son los divisores. Se divide desde 1 hasta n-1
         if (n % i == 0) {
            suma = suma + i;     // si es divisor se suma
         }
      }
      if (suma == n) {  // si el numero es igual a la suma de sus divisores es perfecto
         System.out.println("Perfecto");
      } else {
         System.out.println("No es perfecto");

      }
   }
}
```

Utilizando el algoritmo anterior vamos a escribir ahora el programa Java que muestre los números perfectos entre 1 y 1000

```
public class NumerosPerfectos1a1000 {
   public static void main(String[] args) {
      int i, j, suma;
      System.out.println("Números perfectos entre 1 y 1000: ");
      for(i=1;i<=1000;i++){    // i es el número que vamos a comprobar
         suma=0;
         for(j=1;j<i;j++){  // j son los divisores. Se divide desde 1 hasta i-1
            if(i%j==0){
               suma=suma+j;     // si es divisor se suma
            }
         }
        if(i==suma){          // si el numero es igual a la suma de sus divisores es perfecto
```

```
                System.out.println(i);
            }
        }
    }
}
```

Ejercicios Resueltos de Bucles Anidados en Java.

BUCLES ANIDADOS EN JAVA

Primera relación de ejercicios para practicar con los bucles anidados en Java. En este caso todos los ejercicios propuestos se resuelven anidando dos bucles for aunque también se podrían resolver mediante dos bucles while o do..while anidados o mediante combinaciones de los tres tipos: for, while, do..while. Te animo a que los resuelvas utilizando bucles distintos al for utilizado aquí.

1. Leer por teclado un número entero N no negativo y mostrar el factorial de todos los números desde 0 hasta N.

El factorial de un número entero se expresa mediante el símbolo '!' y se define de la siguiente forma:

Si n = 0 entonces 0! = 1

Si n > 0 entonces

n! = n * (n – 1) * (n – 2) * …. * 3 * 2 * 1

Por ejemplo, n = 5 entonces

5! = 5 * 4 * 3 * 2 * 1 = 120

Solución:

```
import java.util.Scanner;

public class BucleAnidados1 {
```

```
    public static void main(String[] args) {
        int N;
        double factorial;
        Scanner sc = new Scanner(System.in);

        //Leer un número entero >= 0
        do{
            System.out.print("Introduce un número > 0: ");
            N = sc.nextInt();
        }while(N<0);

        for(int i = 0; i <= N ; i++){ //para cada número desde 1 hasta N

            //se calcula su factorial
            factorial = 1;
            for(int j = 1; j <= i; j++){
                factorial = factorial * j;
            }

            //se muestra el factorial
            System.out.printf("%2d! = %.0f %n", i, factorial);

        }
    }

}
```

El primer bucle for representa todos los números desde 0 hasta el valor N que se ha introducido por teclado. La variable i toma todos los valores desde 0 hasta N. Para cada uno de estos valores se calcula su factorial mediante el seguno bucle for que se encuentra dentro del primero. Para calcular el factorial se multiplican todos los números desde 1 hasta el número i por el que vamos en cada iteración del for exterior. El resultado se guarda en la variable factorial. Esta variable se ha declarado de tipo double porque el valor del factorial de un número puede fácilmente sobrepasar el rango de los int.
Cuando acaba este bucle interior tendremos el factorial del número i. Mediante una instrucción System.out.printf se muestra el resultado por pantalla y comenzará una nueva iteración del primer for.
Si no tienes muy claro cómo funciona el printf puedes ver ejemplos en esta entrada: java printf.
Ejemplo de ejecución:

```
Introduce un número > 0: 10
 0! = 1
 1! = 1
 2! = 2
 3! = 6
 4! = 24
 5! = 120
 6! = 720
 7! = 5040
 8! = 40320
 9! = 362880
10! = 3628800
```

2. Leer por teclado un número entero N no negativo y mostrar la suma de los factoriales de todos los números desde 0 hasta N.
Este ejercicio es una variante del anterior. Para resolverlo basta con añadir otra variable suma que actúe como acumulador donde sumaremos el factorial obtenido de cada número.
La variable suma también se ha declarado de tipo double igual que la variable factorial.
Solución:

```
import java.util.Scanner;
public class BucleAnidados2 {

    public static void main(String[] args) {
        int N;
        double factorial, suma = 0;
        Scanner sc = new Scanner(System.in);

        //Leer un número entero >= 0
        do {
            System.out.print("Introduce un número > 0: ");
            N = sc.nextInt();
        } while (N < 0);

        for (int i = 0; i <= N; i++) { //para cada número desde 1 hasta N

            //se calcula su factorial
            factorial = 1;
```

```
            for (int j = 1; j <= i; j++) {
                factorial = factorial * j;
            }

            //se muestra el factorial
            System.out.printf("%n%2d! = %.0f %n", i, factorial);

            //se suma el factorial o
            suma = suma + factorial;
        }

        //Al final del proceso se muestra la suma de todos los factoriales
        System.out.printf("Suma de los factoriales desde 0 hasta %d: %.0f%n", N, suma);
    }
}
```

Ejemplo de ejecución:

```
Introduce un número > 0: 12
 0! = 1
 1! = 1
 2! = 2
 3! = 6
 4! = 24
 5! = 120
 6! = 720
 7! = 5040
 8! = 40320
 9! = 362880
10! = 3628800
11! = 39916800
12! = 479001600

Suma de los factoriales desde 0 hasta 12: 522956314
```

3. Programa que muestre lo siguiente por pantalla:

```
ZYWXVUTSRQPONMLKJIHGFEDCBA
YWXVUTSRQPONMLKJIHGFEDCBA
WXVUTSRQPONMLKJIHGFEDCBA
```

XVUTSRQPONMLKJIHGFEDCBA
VUTSRQPONMLKJIHGFEDCBA
UTSRQPONMLKJIHGFEDCBA
TSRQPONMLKJIHGFEDCBA
SRQPONMLKJIHGFEDCBA
RQPONMLKJIHGFEDCBA
QPONMLKJIHGFEDCBA
PONMLKJIHGFEDCBA
ONMLKJIHGFEDCBA
NMLKJIHGFEDCBA
MLKJIHGFEDCBA
LKJIHGFEDCBA
KJIHGFEDCBA
JIHGFEDCBA
IHGFEDCBA
HGFEDCBA
GFEDCBA
FEDCBA
EDCBA
DCBA
CBA
BA
A

Como podemos ver, en este caso se trata de mostrar las letras del abecedario (sin la Ñ) en mayúsculas y en orden inverso. A continuación en cada fila mostrar una letra menos hasta llegar a mostrar solamente la A.
Solución:

```
public class BucleAnidados3 {

    public static void main(String[] args) {
        for (char x = 'Z'; x >= 'A'; x--) {
            for (char y = x; y >= 'A'; y--) {
                System.out.print(y);
            }
            System.out.println();
        }
    }
}
```

Para resolverlo hemos anidado dos bucles for. En este caso las variables de control del bucle son de tipo char. La variable del primer for la vamos a llamar x y tomará los valores desde la

'Z' hasta la 'A'. La variable del for interior la vamos a llamar y. Los valores que tomará esta variable irán desde el valor de la variable x hasta 'A'.

De esta forma, cuando comienza la ejecución del for exterior el valor que toma la variable x es 'Z' y comienza la ejecución del for interior. El valor inicial de y será 'Z' y este bucle interior mostrará todas las letras desde 'Z' hasta 'A'. Cuando este bucle termina se hace un salto de línea y comienza una nueva iteración del bucle exterior. Ahora el valor de x será 'Y' y comienza la ejecución del for interior. El valor inicial de la variable y será 'Y' y este bucle mostrará todas las letras desde la 'Y' hasta la 'A'. Este proceso se repite hasta que finalmente se muestre solamente la letra 'A'.
De esta forma, cuando comienza la ejecución del for exterior el valor que toma la variable x es 'Z' y comienza la ejecución del for interior. El valor inicial de y será 'Z' y este bucle interior mostrará todas las letras desde 'Z' hasta 'A'. Cuando este bucle termina se hace un salto de línea y comienza una nueva iteración del bucle exterior. Ahora el valor de x será 'Y' y comienza la ejecución del for interior. El valor inicial de la variable y será 'Y' y este bucle mostrará todas las letras desde la 'Y' hasta la 'A'. Este proceso se repite hasta que finalmente se muestre solamente la letra 'A'.

Mostrar un contador de 5 digitos

Programa Java que muestre todos los valores de un contador de 5 dígitos empezando por 00000 y acabando en 99999 con la particularidad que cada vez que se deba mostrar un 3 se muestre E.
Solución: para resolverlo anidamos 5 bucles for, uno para dígito del contador.

```
public class MostrarContador5Digitos {

    public static void main(String[] args) {
        for (int i = 0; i <= 9; i++) {
            for (int j = 0; j <= 9; j++) {
                for (int k = 0; k <= 9; k++) {
                    for (int l = 0; l <= 9; l++) {
                        for (int m = 0; m <= 9; m++) {
                            System.out.print(i != 3 ? i : "E");
                            System.out.print(j != 3 ? j : "E");
                            System.out.print(k != 3 ? k : "E");
                            System.out.print(l != 3 ? l : "E");
                            System.out.println(m != 3 ? m : "E");
```

```
                }
            }
        }
    }
  }
}
```

La salida del programa (de forma abreviada) será esta:

```
00000
00001
00002
0000E
00004
.....
29998
29999
E0000
E0001
E0002
E000E
.....
04EE8
04EE9
04E40
.....
99998
99999
```

Relación Nº 1: Ejercicios 1 y 2

1. Calcular la media de una serie de números que se leen por teclado.

Programa Java que lea por teclado 10 números enteros y los guarde en un array. A

continuación calcula y muestra por separado la media de los valores positivos y la de los valores negativos.

```
/*
 * Programa que lea por teclado 10 números enteros y los guarde en un array.
 * A continuación calcula y muestra la media de los valores positivos y la  de los valores negativos.
 */

import java.util.*;
public class Media1 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int i;
        int pos = 0, neg = 0; //contadores de los números positivos y negativos
        int[] numeros = new int[10]; //array que contendrá los números leídos por teclado
        double sumaPos = 0, sumaNeg = 0; //acumuladores para las sumas de positivos y negativos

        //lectura de datos y llenar el array
        System.out.println("Lectura de los elementos del array: ");
        for (i = 0; i < 10; i++) {
            System.out.print("numeros[" + i + "]= ");
            numeros[i]=sc.nextInt();
        }

        //recorrer el array para sumar por separado los números positivos
         // y los negativos
        for (i = 0; i < 10; i++) {
            if (numeros[i] > 0){ //sumar positivos
                sumaPos += numeros[i];
                pos++;
            } else if (numeros[i] < 0){ //sumar negativos
                sumaNeg += numeros[i];
                neg++;
            }
        }

        //Calcular y mostrar las medias
        if (pos != 0) {
            System.out.println("Media de los valores positivos: " + sumaPos / pos);
```

```
        } else {
            System.out.println("No ha introducido numeros positivos");
        }
        if (neg != 0) {
            System.out.println("Media de los valores negativos: " + sumaNeg / neg);
        } else {
            System.out.println("No ha introducido numeros negativos");
        }
    }
}
```

2. Programa Java que lea 10 números enteros por teclado y los guarde en un array. Calcula y muestra la media de los números que estén en las posiciones pares del array.

Considera la primera posición del array (posición 0) como par.

```
/*
 * Leer 10 números enteros y guardarlos
 * en un array. Calcular la media de los
 * que estén en las posiciones pares.
 */
import java.util.*;

public class Arrays1_2 {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);
        int i;
        int[] numeros = new int[10];
        double media = 0;

        //lectura de datos y llenar el array
        System.out.println("Lectura de los elementos del array: ");
        for (i = 0; i < 10; i++) {
            System.out.print("numeros[" + i + "]= ");
            numeros[i] = sc.nextInt();
```

```
        }

        //Recorrer el array y calcular la media
        for (i = 0; i < 10; i++) {
           if (i % 2 == 0){ //si la posición actual es par
              media = media + numeros[i]; //se suma el valor de esa posición
           }
        }
        //Calcular y mostrar la media
        System.out.println("Media de los valores que se encuentran en posiciones pares: "+
media/5);
    }
}
```

Relación Nº 2: Ejercicios 3 y 4

3. Programa que lee por teclado la nota de los alumnos de una clase y calcula la nota media del grupo. También muestra los alumnos con notas superiores a la media. El número de alumnos se lee por teclado.
Este programa utiliza un array de elementos de tipo double que contendrá las notas de los alumnos.
El tamaño del array será el número de alumnos de la clase, por lo tanto primero se pedirá por teclado el número de alumnos y a continuación se creará el array.
Se realizan 3 recorridos sobre el array, el primero para asignar a cada elemento las notas introducidas por teclado, el segundo para sumarlas y calcular la media y el tercero para mostrar los alumnos con notas superiores a la media.

```
import java.util.*;
public class MediaDeLaClase {
    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);
        int numAlum, i;
        double suma = 0, media;
```

```
        //Lectura del número de alumnos. Debe ser un valor positivo
        do {
            System.out.print("Número de alumnos de la clase: ");
            numAlum = sc.nextInt();
        } while (numAlum <= 0);

        //se crea un array llamado notas de numAlumn elementos de tipo double
        double[] notas = new double[numAlum];

        // Entrada de datos. Se lee la nota de cada alummo y se guarda
        // en cada elemento del array
        for (i = 0; i < notas.length; i++) {
            System.out.print("Alumno " + (i + 1) + " Nota final: ");
            notas[i] = sc.nextDouble();
        }

        // Sumar todas las notas
        for (i = 0; i < notas.length; i++) {
            suma = suma + notas[i];
        }

        // Calcular la media
        media = suma / notas.length;

        // Mostrar la media
        System.out.printf("Nota media del curso: %.2f %n", media);

        // Mostrar los valores superiores a la media
        System.out.println("Listado de notas superiores a la media: ");
        for (i = 0; i < notas.length; i++) {
            if (notas[i] > media) {
                System.out.println("Alumno numero " + (i + 1)+ " Nota final: " + notas[i]);
            }
        }
    }
}
```

4. Programa que crea un array de 20 elementos llamado Pares y guarde los 20 primeros números pares. Mostrar por pantalla el contenido del array creado.

```
/*
 * Programa que crea un array de 20 elementos
 * llamado Pares y guarde los 20 primeros
 * números pares.
 * Mostrar por pantalla el contenido
 * del array creado
 */
import java.util.*;
public class ArrayPares {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);
        int i, cont = 2;

        int[] pares = new int[20];

            //Llenamos el array con números pares. Utilizamos un contador
        //con valor inicial 2 y le sumamos dos en cada iteración.
        for (i = 0; i < pares.length; i++) {
            pares[i] = cont;
            cont += 2;
        }

            //Mostrar el array
        for (i = 0; i < pares.length; i++) {
            System.out.println(pares[i]);
        }
    }
}
```

Relación Nº 3: Ejercicios 5, 6 y 7

5. Programa Java que guarda en un array 10 números enteros que se leen por teclado. A continuación se recorre el array y calcula cuántos números son positivos, cuántos negativos y cuántos ceros.

```
 // Contar el número de elementos positivos, negativos y ceros de un array de 10 elementos.
import java.util.*;
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int[] numeros = new int[10];
        int pos = 0, neg = 0, cero = 0; //contadores
        int i;

//Leemos los valores por teclado y los guardamos en el array
        System.out.println("Lectura de los elementos del array: ");
        for (i = 0; i < numeros.length; i++) {
            System.out.print("numeros[" + i + "]= ");
            numeros[i] = sc.nextInt();
        }
        //se recorre el array para contar positivos, negativos y ceros
        for (i = 0; i < numeros.length; i++) {
            if (numeros[i] > 0) {
                pos++;
            } else if (numeros[i] < 0) {
                neg++;
            } else {
                cero++;
            }
        }
        //mostrar resultados
        System.out.println("Positivos: " + pos);
        System.out.println("Negativos: " + neg);
        System.out.println("Ceros: " + cero);
    }
}
```

6. Programa Java que llene un array con 10 números enteros que se leen por teclado. A continuación calcula y muestra la media de los valores positivos y la de los valores negativos del array.

```
/*
 * Leer 10 números enteros y mostrar la media de los valores positivos y la de los valores negativos.
 */
import java.util.*;
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int[] numeros = new int[10];
        int pos = 0, neg = 0;  //contadores para positivos y negativos
        int i;
        double sumaPos = 0, sumaNeg = 0;  //suma de positivos y negativos
        //Leemos los valores por teclado y los guardamos en el array
        System.out.println("Lectura de los elementos del array: ");
        for (i = 0; i < numeros.length; i++) {
            System.out.print("numeros[" + i + "]= ");
            numeros[i]=sc.nextInt();
        }
        //se recorre el array para sumar positivos y negativos
        for (i = 0; i < numeros.length; i++) {
            if (numeros[i] > 0){ //sumar positivos
                sumaPos += numeros[i];
                pos++;
            } else if (numeros[i] < 0){ //sumar negativos
                sumaNeg += numeros[i];
                neg++;
            }
        }
        //mostrar resultados
        if (pos != 0) {
            System.out.println("Media de los valores positivos: " + sumaPos / pos);
        } else {
            System.out.println("No ha introducido números positivos");
        }
        if (neg != 0) {
            System.out.println("Media de los valores negativos: " + sumaNeg / neg);
```

```
        } else {
            System.out.println("No ha introducido números negativos");
        }
    }
}
```

7. Programa Java para leer la altura de N personas y calcular la altura media. Calcular cuántas personas tienen una altura superior a la media y cuántas tienen una altura inferior a la media. El valor de N se pide por teclado y debe ser entero positivo.

```
/*
 * Leer la altura de N personas y calcular la altura media
 * Mostra cuántos hay superiores a la media.
 */
import java.util.*;
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int i, N;
        int contMas = 0, contMenos = 0;
        double media = 0;
        //Leer el número de personas
        do{
            System.out.print("Número de personas: ");
            N = sc.nextInt();
        }while(N<=0);
        //Se crea el array de tamaño N
        double[] alto = new double[N];
        //Leer alturas
        System.out.println("Lectura de la altura de las personas: ");
        for (i = 0; i < N; i++) {
            System.out.print("persona " + (i+1) + " = ");
            alto[i] = sc.nextDouble();
            media = media + alto[i]; //se suma la estatura leída para calcular la media
        }
        //Calcular la media
        media = media / N;
        //recorremos el array para ver cuantos hay más altos
        //que la media y cuantos más bajos
        for (i = 0; i < alto.length; i++) {
            if (alto[i] > media){ //si la estatura es mayor que la media
```

```
            contMas++;
          } else if (alto[i] < media){ //si es menor
            contMenos++;
          }
        }
        //Mostrar resultados
        System.out.println("Estatura media: " + media);
        System.out.println("Personas con estatura superior a la media: " + contMas);
        System.out.println("Personas con estatura inferior a la media: " + contMenos);
    }
}
```

Programa Java que lea el nombre y el sueldo de 20 empleados y muestre el nombre y el sueldo del empleado que más gana.

Para hacerlo utilizaremos dos arrays:
Un array de String para los nombres de los empleados
Un array de tipo double para los sueldos de cada empleado.

Al mismo tiempo que leemos los datos de los empleados iremos comprobando cuál es el que tiene el mayor sueldo. Para ello tomamos el sueldo del primer empleado que se lee como mayor sueldo y después vamos comprobando el resto de sueldos. Cuando encontramos alguno mayor que el mayor actual este sueldo se convierte en el nuevo mayor.

En general para calcular el mayor de una serie de números tomamos el primero como mayor y después comparamos el resto de números.

```
//programa que muestra el nombre y el sueldo del empleado que más gana
import java.util.*;
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
```

```
        //creamos los arrays
        String[] empleados = new String[20];
        double[] sueldos = new double[20];

        //variables donde guardar el nombre y sueldo del empleado que más gana
        String nombreMayor;
        double mayorSueldo;

        int i = 0;

        //se lee el primer empleado
        System.out.println("Lectura de nombres y sueldos de empleados: ");
        System.out.print("Empleado " + (i + 1) + ": ");
        empleados[i] = sc.nextLine();
        System.out.print("Sueldo: ");
        sueldos[i] = sc.nextDouble();

        //se toma el primero como mayor
        mayorSueldo = sueldos[i];
        nombreMayor = empleados[i];

        //se leen el resto de empleados
        for (i = 1; i < empleados.length; i++) {
            sc.nextLine(); //limpiar el buffer
            System.out.print("Empleado " + (i + 1) + ": ");
            empleados[i] = sc.nextLine();
            System.out.print("Sueldo: ");
            sueldos[i] = sc.nextDouble();
            //se compara el sueldo leído con el mayor
            if (sueldos[i] > mayorSueldo) {
                mayorSueldo = sueldos[i];
                nombreMayor = empleados[i];
            }
        }

        //mostrar resultados
        System.out.println("Empleado con mayor sueldo: " + nombreMayor );
        System.out.println("Sueldo: " + mayorSueldo);
    }
}
```

Llenar un array con números aleatorios

En esta entrada vamos a escribir un método Java que llene un array de enteros con números aleatorios.
Los números aleatorios deberán estar comprendidos entre dos límites (desde, hasta) ambos incluidos.
El método recibe como parámetros los valores desde, hasta y el tamaño del array.
El método devuelve mediante return el array de enteros creado.
Para obtener números enteros aleatorios usaremos el método nextInt() de la clase Random.
Para que los números aleatorios obtenidos estén dentro de los limites utilizaremos el método nextInt() de la siguiente fórma: nextInt(hasta - desde + 1) + desde

```
public static int [] llenarArrayAleatorio(int desde, int hasta, int tam){
      int[] numeros = new int[tam];
      Random rnd = new Random();
      for (int i = 0; i < numeros.length; i++) {
         numeros[i] = rnd.nextInt(hasta - desde + 1) + desde;
      }
      return numeros;
}
```

Si los números no se pueden repetir debemos complicar un poco más el código.
En este caso cada vez que se obtiene un número aleatorio se debe comprobar si ya está en el array.
Para hacer este trabajo vamos a escribir un método llamado comprobarSiContiene. A este método se le envía el array, la posición del elemento actual y el número aleatorio a insertar y devuelve true si el número ya existe en el array. En ese caso se vuelve a sacar otro número aleatorio.

```
public static int[] llenarArrayAleatorio(int desde, int hasta, int tam) {
      int[] numeros = new int[tam];
      Random rnd = new Random();
      int n;
```

```
        for (int i = 0; i < numeros.length; i++) {
            do {
                n = rnd.nextInt(hasta - desde + 1) + desde;
            } while (comprobarSiContiene(numeros, i, n));
            numeros[i] = n;
        }
        return numeros;
}

public static boolean comprobarSiContiene(int[] numeros, int indice, int valor) {
        for (int i = 0; i < indice; i++) {
            if (numeros[i] == valor) {
                return true;
            }
        }
        return false;
}
```

Esta sería la solución para aquellos que están empezando a programar. Pero en este caso nos podemos ahorrar trabajo utilizando la API de Java. En concreto utilizando un ArrayList y el método shuffle de Collections.

La idea es llenar un ArrayList con todos los números comprendidos entre los límites desde/ hasta. A continuación se llama al método shuffle para desordenarlos y después se pasan al array de enteros los tam primeros elementos, donde tam es el tamaño del array a devolver.

El método utilizando el ArrayList y Collections.shuffle quedaría así:

```
private static int[] llenarArrayAleatorio(int desde, int hasta, int tam) {
        ArrayList<Integer> array = new ArrayList<Integer>();
        for (int i = desde; i <= hasta; i++) {
            array.add(i);
        }
        Collections.shuffle(array);
        int [] numeros = new int[tam];
        for(int i = 0; i<numeros.length; i++){
            numeros[i]=array.get(i);
        }
        return numeros;
}
```

Java Ejercicios Básicos Resueltos Estructura Condicional 1

Los siguiente ejercicios utilizan la estructura condicional también llamada estructura alternativa o selectiva.

Relación Nº 1: Ejercicios 1, 2 y 3
1. Programa Java que lea un número entero por teclado y calcule si es par o impar.

Podemos saber si un número es par si el resto de dividir el número entre 2 es igual a cero. En caso contrario el número es impar

El operador Java que calcula el resto de la división entre dos números enteros o no es el operador %

El programa que calcula si un número entero es par o impar es el siguiente:

```
import java.util.*;
public class Condicional1_1 {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      int N;
      System.out.print("Introduzca Número entero: ");
      N = sc.nextInt();
      if(N%2==0)
        System.out.println("Par");
      else
         System.out.println("Impar");
   }
}
```

2. Programa que lea un número entero y muestre si el número es múltiplo de 10.

Podemos comprobar si un número entero es múltiplo de 10 si al dividirlo por 10 es resto de esta división es cero.

```
/* Programa que lea un número entero y muestre si el número es múltiplo de 10 */
import java.util.*;
public class Condicional1_2 {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      int N;
      System.out.print("Número entero: ");
      N = sc.nextInt();
      if(N%10==0)
        System.out.println("Es múltiplo de 10");
      else
       System.out.println("No es múltiplo de 10");
   }
}
```

3. Programa que lea un carácter por teclado y compruebe si es una letra mayúscula

```
/* condicional1_3
 * Programa que lea un carácter por teclado y compruebe si es una letra mayúscula
 */
import java.io.*;
import java.util.*;
public class condicional1_3 {
   public static void main(String[] args) throws IOException{
        Scanner sc = new Scanner(System.in);
        char car, car1;
        System.out.print("Introduzca un carácter: ");
        car = (char)System.in.read(); //lee un solo caracter

        if(Character.isUpperCase(car)) //utilizamos el método isUpperCase de la clase
Character
           System.out.println("Es una letra mayúscula");
        else
           System.out.println("No es una letra mayúscula");
    }
```

Forma alternativa de comprobar si un carácter es una letra mayúscula sin utilizar el método isUpperCase, en este caso comparando directamente con los caracteres A y Z

```
if(car>='A' && car <='Z')
  System.out.println("Es una letra mayúscula");
else
  System.out.println("No es una letra mayúscula");
```

Java Ejercicios Básicos Resueltos Estructura Condicional 2

Continuamos con más ejercicios básicos que usan la estructura condicional.
Relación Nº 2: Ejercicios 4 y 5
Ejercicio 4: Programa que lea dos caracteres y compruebe si son iguales.

```
/*
 *  Ejemplo básico java
 * Programa con estructura condicional
 * Programa que lea dos caracteres y compruebe
 * si son iguales.
 */
import java.io.*;
public class condicional1_5 {
   public static void main(String[] args) throws IOException {
      char car1, car2;
      System.out.print("Introduzca primer carácter: ");
      car1 = (char)System.in.read(); //lee un carácter
      System.in.read();  //saltar el intro que ha quedado en el buffer
      System.out.print("Introduzca segundo carácter: ");
      car2 = (char)System.in.read(); //lee el segundo carácter

      if(car1 == car2)
        System.out.println("Son iguales");
      else
        System.out.println("No son iguales");
```

```
    }
}
```

Una forma alternativa de hacer este programa es creando dos objetos Character a partir de los caracteres que se han leído y compararlos utilizando el método equals

```
import java.io.*;
public class condicional1_5 {
    public static void main(String[] args) throws IOException {
        char car1, car2;
        System.out.print("Introduzca primer carácter: ");
        car1 = (char)System.in.read(); //lee un carácter
        System.in.read();  //saltar el intro que ha quedado en el buffer
        System.out.print("Introduzca segundo carácter: ");
        car2 = (char)System.in.read(); //lee el segundo carácter
        Character c1 = new Character(car1);
        Character c2 = new Character(car2);
        if(c1.equals(c2))  //comparamos dos objetos Character mediante el método equals
            System.out.println("Son iguales");
        else
            System.out.println("No son iguales");
    }
}
```

Ejercicio 5: Programa java que lea dos caracteres por teclado y compruebe si los dos son letras minúsculas

```
/* Ejemplo básico java de programa con estructura condicional
 * Programa que lea dos caracteres y compruebe si los dos son letras minúsculas
 */
import java.io.*;
public class condicional1_6 {
public static void main(String[] args) throws IOException {
        char car1, car2;
        System.out.println("Introduzca primer carácter: ");
        car1 = (char)System.in.read(); //lee un carácter
        System.in.read();  //saltar el intro que ha quedado
```

```
        System.out.println("Introduzca segundo carácter: ");
        car2 = (char)System.in.read(); //lee el segundo carácter
        if(Character.isLowerCase(car1)){ //utilizamos el método isLowerCase de la clase
Character
          if(Character.isLowerCase(car2))
             System.out.println("Los dos son letras minúsculas");
          else
             System.out.println("El primero es una letra minúscula pero el segundo no");
        }
        else{
           if(Character.isLowerCase(car2))
             System.out.println("El segundo es una letra minúscula pero el primero no");
          else
             System.out.println("Ninguno es una letra minúscula");
        }
    }
}
```

Una forma alternativa de resolver este ejercicio es comparando directamente las dos variables con las letras minúsculas

```
        if(car1>='a' && car1<='z'){
           if(car2>='a' && car2<='z')
             System.out.println("Los dos son letras minúsculas");
          else
             System.out.println("El primero es una letra minúscula pero el segundo no");
        }
        else{
           if(car2>='a' && car2<='z')
             System.out.println("El segundo es una letra minúscula pero el primero no");
          else
             System.out.println("Ninguno es una letra minúscula");
        }
```

Aquí teneis otra entrega de ejercicios básicos con estructura condicional.

Relación Nº 3: Ejercicios 6 y 7

Ejercicio 6: Programa java que lea un carácter por teclado y compruebe si es un dígito numérico (cifra entre 0 y 9).

Vamos a escribir dos soluciones a este ejercicio.

La primera consiste en comprobar si el carácter es un dígito mediante el método isDigit de la clase Character. Este método devuelve true si el carácter que se le pasa como parámetro es una cifra entre 0 y 9:

```
/*
  * Ejemplo de programa con estructura condicional
 * Programa que lea un carácter por teclado y compruebe si es un número
 */
import java.io.*;
public class condicional1_7 {
   public static void main(String[] args) throws IOException {
       char car1;
       System.out.print("Introduzca carácter: ");
       car1 = (char)System.in.read(); //lee un carácter
       if(Character.isDigit(car1)) //utilizamos el método isDigit de la clase Character
          System.out.println("Es un número");
       else
          System.out.println("No es un número");
  }
}
```

La otra solución es directa y consiste en comprobar si el carácter que se ha leído por teclado es mayor o igual que el carácter 0 y menor o igual que el carácter 9.

```
// Versión alternativa comparando con
// los caracteres '0' ... '9'

    if(car1>='0' && car1<='9')
       System.out.println("Es un número");
       else
          System.out.println("No es un número");
```

Ejercicio 7: Programa que lea dos números por teclado y muestre el resultado de la división del primer número por el segundo. Se debe comprobar que el divisor no puede ser cero.

```
/*
 * Ejemplo de programa con estructura condicional
 * Programa que lea dos números por teclado y muestre el resultado
 * de la división del primero por el segundo.
 * Se comprueba que el divisor es distinto de cero.
 */
import java.util.*;
public class condicional1_8 {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      double dividendo, divisor;
      System.out.print("Introduzca el dividendo: ");
      dividendo = sc.nextDouble();
      System.out.print("Introduzca el divisor: ");
      divisor = sc.nextDouble();
      if(divisor==0)
        System.out.println("No se puede dividir por cero");
      else{
         System.out.println(dividendo + " / " + divisor + " = " + dividendo/divisor);
         System.out.printf("%.2f / %.2f = %.2f %n" , dividendo, divisor , dividendo/divisor);
      }
   }
}
```

Mayor de tres numeros

Calcular el mayor de tres números enteros en Java.

El programa lee por teclado tres números enteros y calcula y muestra el mayor de los tres.

```
/*
```

```
 * Programa que lee tres números distintos
 * y nos dice cuál de ellos es el mayor
 */
import java.util.*;
public class MayorDeTres {
 public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n1, n2, n3;
        System.out.print("Introduzca primer número: ");
        n1 = sc.nextInt();
        System.out.print("Introduzca segundo número: ");
        n2 = sc.nextInt();
        System.out.print("Introduzca tercer número: ");
        n3 = sc.nextInt();
        if (n1 > n2) {
           if (n1 > n3) {
              System.out.println("El mayor es: " + n1);
           } else {
              System.out.println("el mayor es: " + n3);
           }
        } else if (n2 > n3) {
           System.out.println("el mayor es: " + n2);
        } else {
           System.out.println("el mayor es: " + n3);
        }
    }
}
```

Java Ejercicios Básicos Condicional 4

Relación Nº 4: Ejercicios 9 y 10

Ejercicio 9:

Programa que lea por teclado tres números enteros H, M, S correspondientes a hora, minutos y segundos respectivamente, y comprueba si la hora que indican es una hora válida.

Supondremos que se leemos una hora en modo 24 Horas, es decir, el valor válido para las horas será mayor o igual que cero y menor que 24.

El valor válido para los minutos y segundos estará comprendido entre 0 y 59 ambos incluidos

```
/*
 * Programa java que lea tres números enteros H, M, S que contienen hora, minutos y
 * segundos respectivamente, y comprueba si la hora que indican es una hora válida.
 */
import java.util.*;
public class condicional1_16 {
    public static void main(String[] args) {
        int H,M,S;
        Scanner sc = new Scanner(System.in);
        System.out.print("Introduzca hora: ");
        H =  sc.nextInt();
        System.out.print("Introduzca minutos: ");
        M =  sc.nextInt();
        System.out.print("Introduzca segundos: ");
        S =  sc.nextInt();
        if(H>=0 && H<24 && M>=0 && M<60 && S>=0 && S<60)
           System.out.println("Hora correcta");
        else
            System.out.println("Hora incorrecta");
    }
}
```

La instrucción que comprueba si la hora leída por teclado es correcta es:

```
if(H>=0 && H<24 && M>=0 && M<60 && S>=0 && S<60)
```

Esta condición da como resultado true cuando la hora es mayor o igual a 0 y menor que 24, los minutos son mayores o iguales a 0 y menores que 60 y los segundos son mayores 0 iguales a cero y menores a 60.

Ejercicio 10:

Programa que lea una variable entera mes y compruebe si el valor corresponde a un mes de 30 días, de 31 o de 28. Supondremos que febrero tiene 28 días. Se mostrará además el nombre

del mes. Se debe comprobar que el valor introducido esté comprendido entre 1 y 12.

```
/*
 * Programa java que lea una variable entera mes y compruebe si el valor corresponde
 * a un mes de 30 días. Se debe comprobar que el valor introducido esté
 * comprendido entre 1 y 12
 */
import java.util.*;
public class condicional1_17 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int mes;
        System.out.print("Introduzca número de mes: ");
        mes =  sc.nextInt();
        if(mes < 1 || mes > 12) //se comprueba que el valor del mes es correcto
            System.out.println("Mes incorrecto");
        else{  //si el mes es correcto
            switch(mes){  //se muestra el nombre mediante una instrucción switch
                case 1: System.out.print("Enero");
                        break;
                case 2: System.out.print("Febrero");
                        break;
                case 3: System.out.print("Marzo");
                        break;
                case 4: System.out.print("Abril");
                        break;
                case 5: System.out.print("Mayo");
                        break;
                case 6: System.out.print("Junio");
                        break;
                case 7: System.out.print("Julio");
                        break;
                case 8: System.out.print("Agosto");
                        break;
                case 9: System.out.print("Septiembre");
                        break;
                case 10: System.out.print("Octubre");
                        break;
                case 11: System.out.print("Noviembre");
                        break;
                case 12: System.out.print("Diciembre");
```

```
                    break;
            }
            // mostrar si es un mes de 30, 31 0 28 días
            if(mes == 4 || mes == 6 || mes == 9 || mes == 11)
              System.out.println(" es un mes de 30 días");
            else if(mes == 2)
                    System.out.println(" es un mes de 28 días");
                  else
                    System.out.println(" es un mes de 31 días");
        }
    }
}
```

Estructura Repetitiva en Java. Ejercicios Básicos Resueltos 1

Relación Nº 1: Ejercicios 1, 2, 3, 4, 5 y 6
Ejercicios básicos que utilizan la estructura repetitiva.

Se trata de mostrar los números desde el 1 hasta el 100 utilizando las instrucciones repetitivas while, do while y for.

1. Ejemplo de uso de while: Programa Java que muestre los números del 1 al 100 utilizando la instrucción while.

```
/*
 * Ejercicios básicos java con estructura iterativa o repetitiva
 * Mostrar los números del 1 al 100
 * utilizando un bucle while
 */
public class Main {
    public static void main(String[] args) {
        System.out.println("Numeros del 1 al 100: ");
        int i=1;
        while(i<=100) {
                System.out.println(i);
                i++;
        }
    }
```

```
}
```

2. Ejemplo de uso de do-while. Programa Java que muestre los números del 1 al 100 utilizando la instrucción do..while.

```
/*
 * Ejercicios básicos java con estructura iterativa o repetitiva
 * Mostrar los números del 1 al 100 utilizando un bucle do while
 */
public class Main {
    public static void main(String[] args) {
        int i = 1;
        System.out.println("Numeros del 1 al 100: ");
        do{
                System.out.println(i);
                i++;
        }while(i<=100);
    }
}
```

3. Ejemplo de uso de for. Programa Java que muestre los números del 1 al 100 utilizando la instrucción for.

```
/*
 * Ejercicios básicos java con estructura iterativa o repetitiva
 * Mostrar los números del 1 al 100 utilizando un bucle for
 */
public class Repetitiva1_3 {
    public static void main(String[] args) {
        System.out.println("Numeros del 1 al 100: ");
        for(int i = 1; i<=100;i++)
            System.out.println(i);
    }
}
```

4. Ejemplo de uso de while. Programa Java que muestre los números del 100 al 1 utilizando la instrucción while.

```
/*
 * Ejercicios básicos java con estructura iterativa o repetitiva
 * Mostrar los números del 100 al 1 utilizando un bucle while
 */
```

```
public class Main {
    public static void main(String[] args) {
        System.out.println("Numeros del 100 al 1: ");
         int i=100;
         while(i>=1)
         {
                 System.out.println(i);
                 i--;
         }
    }
}
```

5. Ejemplo de uso de do-while. Programa Java que muestre los números del 100 al 1 utilizando la instrucción do..while.

```
/*
 * Ejercicios básicos java con estructura iterativa o repetitiva
 * Mostrar los números del 100 al 1 utilizando un bucle do while
 */
public class Main {
    public static void main(String[] args) {
        int i = 100;
        System.out.println("Numeros del 100 al 1: ");
         do{
                 System.out.println(i);
                 i--;
        }while(i>=1);
    }
}
```

6. Ejemplo de for. Programa Java que muestre los números del 100 al 1 utilizando la instrucción for.

```
/*
 * Ejercicios básicos java con estructura iterativa o repetitiva
 * Mostrar los números del 100 al 1 utilizando un bucle for
 */
public class Repetitiva1_6 {
    public static void main(String[] args) {
        System.out.println("Numeros del 100 al 1: ");
        for(int i=100;i>=1;i--)
```

```
            System.out.println(i);
    }
}
```

Intercambiar el contenido de dos variables en Java

En esta entrada vamos a explicar dos métodos para realizar el intercambio de valores entre dos variables.
El primer método de intercambio utiliza una variable auxiliar y el segundo método realiza el intercambio de valores sin utilizar variable auxiliar.
Intercambio de valores entre dos variables utilizando una variable auxiliar.

Programa para intercambiar el valor de dos variables. Los valores iniciales se leen por teclado.

Por ejemplo, suponiendo que las variables se llaman A y B, si A contiene 3 y B contiene 5, después del intercambio A contendrá 5 y B 3.

En este ejemplo, para intercambiar el valor entre dos variables utilizaremos una avariable auxiliar donde guardar el valor de una de ellas. Después veremos la forma de hacerlo sin usar una variable auxiliar para el intercambio.

Las instrucciones a realizar son:

```
AUX = A;
A = B;
B = AUX;
```

Programa completo:

```
/*
 * Programa que lea dos variables
```

```
 * numéricas A y B e
 * intercambie sus contenidos.
 */
import java.util.*;
public class Secuenciales2_5 {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      int A, B, AUX;
      System.out.print("Introduzca valor de A: ");
      A = sc.nextInt();
      System.out.print("Introduzca Valor de B: ");
      B = sc.nextInt();
      System.out.println("Valores iniciales: A = " + A + "   B = " + B);
      //instrucciones para hacer el intercambio
      //se utiliza una variable auxiliar
      AUX = A;
      A = B;
      B = AUX;
      System.out.println("Valores intercambiados: A = " + A + "   B = " + B);
   }
}
```

Intercambio de valores entre dos variables sin utilizar variable auxiliar.
También se puede intercambiar el valor de dos variables sin utilizar una variable auxiliar.
En ese caso se resuelve utilizando aritmética básica:

```
A = A + B;
B = A - B;
A = A - B;
```

Si por ejemplo A = 5 y B = 3

```
A = 5 + 3 = 8
B = 8 - 3 = 5
A = 8 - 5 = 3
```

Programa completo:

```
/*
 * Programa que intercambie dos variables sin auxiliar
 */
import java.util.*;
public class Secuenciales2_5 {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      int A, B, AUX;
      System.out.print("Introduzca valor de A: ");
      A = sc.nextInt();
      System.out.print("Introduzca Valor de B: ");
      B = sc.nextInt();
      System.out.println("Valores iniciales: A = " + A + "   B = " + B);
      //instrucciones para hacer el intercambio
      //sin utilizar una variable auxiliar
      A = A + B;
      B = A - B;
      A = A - B;
      System.out.println("Valores intercambiados: A = " + A + "   B = " + B);
   }
}
```

Contar las cifras de un número entero en Java

Programa Java que pide un número entero por teclado y calcula y muestra el número de cifras que tiene.

Por ejemplo si se introduce el número 54391 el programa mostrará el mensaje:

El número tiene 5 cifras

Si se introduce el número 101 se mostrará el mensaje:

El número tiene 3 cifras

El proceso leer un número y contar sus cifras se repetirá hasta que se conteste ‘N’ a la pregunta Continuar? (S/N)
Si se responde 'S' se volverá a pedir otro número.

Para saber cuántas cifras tiene un número entero haremos lo siguiente:

Dividiremos el número sucesivamente entre 10. En cada división tomaremos la parte entera y volvemos a dividir. Este proceso se repite hasta que se obtenga un cero como cociente en la división.

Por ejemplo, si el número leído es 1234 haremos las siguientes operaciones:

1234 / 10 = 123
123 / 10 = 12
12 / 10 = 1
1 / 10 = 0

hemos realizado 4 divisiones hasta obtener un cero como cociente, por lo tanto el número tiene 4 cifras.

```
// Programa Java que calcula el número de cifras que tiene un número entero
import java.util.*;
import java.io.*;

public class Main {
   public static void main(String[] args) throws IOException{
      Scanner sc = new Scanner(System.in);
      int n, cifras;
      char car;
      do{
         System.out.print("Introduce un número entero: ");
         n = sc.nextInt();
         cifras= 0;    //esta variable es el contador de cifras
         while(n!=0){            //mientras a n le queden cifras
               n = n/10;         //le quitamos el último dígito
               cifras++;         //sumamos 1 al contador de cifras
         }
         System.out.println("El número tiene " + cifras+ " cifras");
         System.out.print("Continuar? ");
```

```
            car = (char)System.in.read();
        }while(car!='n' && car != 'N');
    }
}
```

Pasar de grados centígrados a kelvin en Java

Programa Java que lee una temperatura expresada en grados centígrados y la convierte a grados kelvin.

El proceso de leer grados centígrados se debe repetir mientras que se responda ‘S’ a la pregunta: Repetir proceso? (S/N)

Para hacer la conversión de grados Centígrados a grados Kelvin hay que utilizar la fórmula:

ºK = ºC + 273

El programa java para realizar la conversión de temperaturas es el siguiente:

```
import java.util.*;
import java.io.*;
/**
 * Programa que lee una temperatura expresada en grados centígrados y los pasa a grados kelvin.
 * Repetir el proceso mientras que se responda ‘S’ a la pregunta:
 * Repetir proceso? (S/N)
 * @author Enrique
 */
public class CentigradosAKelvin {
    public static void main(String[] args) throws IOException{
        Scanner sc = new Scanner(System.in);
        double temperatura;
        char car;
        do{
            System.out.print("Introduce temperatura en ºC: ");
            temperatura = sc.nextDouble();
```

```
            System.out.println("Grados Kelvin ..: " + (temperatura+273));
            System.out.print("Repetir proceso? (S/N): " );
            car = (char)System.in.read();
        }while(car =='S' || car == 's');
    }
}
```

Mostrar la tabla de multiplicar de un número en Java

Programa Java que lea un número entero N y muestre la tabla de multiplicar de ese número. Por ejemplo, si se lee el valor 7 se mostrará por pantalla:

```
Tabla del 7
--------------
7 * 1  =  7
7 * 2  = 14
7 * 3  = 21
7 * 4  = 28
7 * 5  = 35
7 * 6  = 42
7 * 7  = 49
7 * 8  = 56
7 * 9  = 63
7 * 10 = 70
```

```
import java.util.*;
/**
 * Programa que lea un número entero N y muestre la tabla de multiplicar de ese número.
 * @author Enrique
 */
public class TablaMultiplicar {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n;
        System.out.print("Introduce un número entero: ");
```

```
        n = sc.nextInt();
        System.out.println("Tabla del " + n);
        for(int i = 1; i<=10; i++){
            System.out.println(n + " * " + i + " = " + n*i);
        }
    }
}
```

Contar números acabados en 2

Programa que lea una serie de números por teclado hasta que se lea un número negativo. El programa indicará cuántos números acabados en 2 se han leído.

Para saber si un número acaba en dos o en general para saber en qué dígito termina un número entero se divide el número entre 10 y se obtiene el resto de esta división.
En Java el operador que obtiene el resto de una división es el operador %
En este caso para saber si el número acaba en 2 escribiremos la instrucción:
if(n%10==2)

```
import java.util.*;
/**
 * Programa que lea una serie de números por teclado hasta que
 * se lea un número negativo. El programa indicará cuántos números
 * acabados en 2 se han leído.
 * @author Enrique
 */
public class AcabadosEn2 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n, contador=0;
        System.out.print("Introduce un número entero: ");
        n = sc.nextInt();
        while(n>=0){
            if(n%10==2)  //Si el número acaba en dos
              contador++;       //esta variable contendrá cuántos números acabados en 2 se han
leído.
            System.out.print("Introduce un número entero: ");
            n = sc.nextInt();
```

```
    }
        System.out.println("Se han introducido " + contador + " números acabados en 2");
    }
}
```

Número perfecto en java

Qué es un número perfecto?

Un número es perfecto si es igual a la suma de todos sus divisores positivos sin incluir el propio número.

Por ejemplo, el número 6 es perfecto.

El 6 tiene como divisores: 1, 2, 3 y 6 pero el 6 no se cuenta como divisor para comprobar si es perfecto.

Si sumamos 1 + 2 + 3 = 6

Los siguientes números perfectos después del 6 son 28, 496, 8128, 33550336, 8589869056.

En esta entrada vamos a desarrollar el algoritmo para comprobar si un número es perfecto.
El programa pide por teclado un número y muestra si es perfecto o no. mediante un bucle for sumaremos los divisores del número. Al final si esta suma es igual al número mostraremos el mensaje correspondiente.

Programa java para calcular si un número es perfecto:

```
import java.util.Scanner;
public class NumeroPerfecto {
    public static void main(String[] args) {
        int i, suma = 0, n;
        Scanner sc = new Scanner(System.in);
        System.out.println("Introduce un número: ");
        n = sc.nextInt();
        for (i = 1; i < n; i++) {  // i son los divisores. Se divide desde 1 hasta n-1
            if (n % i == 0) {
```

```
            suma = suma + i;     // si es divisor se suma
          }
       }
       if (suma == n) {  // si el numero es igual a la suma de sus divisores es perfecto
          System.out.println("Perfecto");
       } else {
          System.out.println("No es perfecto");

       }
    }
}
```

Utilizando el algoritmo anterior vamos a escribir ahora el programa Java que muestre los números perfectos entre 1 y 1000

```
public class NumerosPerfectos1a1000 {
   public static void main(String[] args) {
      int i, j, suma;
      System.out.println("Números perfectos entre 1 y 1000: ");
      for(i=1;i<=1000;i++){    // i es el número que vamos a comprobar
         suma=0;
         for(j=1;j<i;j++){  // j son los divisores. Se divide desde 1 hasta i-1
            if(i%j==0){
              suma=suma+j;     // si es divisor se suma
            }
         }
        if(i==suma){          // si el numero es igual a la suma de sus divisores es perfecto
            System.out.println(i);
          }
      }
   }
}
```

Números amigos en Java

COMPROBAR SI DOS NÚMEROS SON AMIGOS

Dos números enteros positivos A y B son números amigos si la suma de los divisores propios de A es igual a B y la suma de los divisores propios de B es igual a A.

Los divisores propios de un número incluyen la unidad pero no el propio número.

Un ejemplo de números amigos son los números 220 y 284.
Los divisores propios de 220 son 1, 2, 4, 5, 10, 11, 20, 22, 44, 55 y 110.
La suma de los divisores propios de 220 da como resultado 284
Los divisores propios de 284 son 1, 2, 4, 71 y 142.
La suma de los divisores propios de 284 da como resultado 220.
Por lo tanto 220 y 284 son amigos.

Otras parejas de números amigos son:

1184, 1210
2620, 2924
5020, 5564
6232, 6368
10744, 10856
12285, 14595
17296, 18416

Vamos a escribir el programa que calcula si dos números son amigos:

```
import java.util.*;
/**
 * Programa Java que determina si dos números son amigos.
 * Dos números son amigos si la suma de los divisores propios del primero
 * es igual al segundo y viceversa.
 */
public class NumerosAmigos {
   public static void main(String[] args) {
      int i,suma=0, n1, n2;
      Scanner sc = new Scanner(System.in);
      System.out.print("Introduce primer número: ");
      n1 = sc.nextInt();
      System.out.print("Introduce segundo número: ");
```

```
        n2 = sc.nextInt();
        for(i=1;i<n1;i++){  // for para sumar todos los divisores propios de n1
            if(n1%i==0){
              suma=suma+i;
            }
        }
        // si la suma de los divisores de n1 es igual a n2
        if(suma==n2){
          suma=0;
          for(i=1;i<n2;i++){  // sumo los divisores propios de n2
              if(n2%i==0){
                suma=suma+i;
              }
          }
        // si la suma de los divisores de n2 es igual a n1
          if(suma==n1){
            System.out.println("Son Amigos");
          }else{
                System.out.println("No son amigos");
          }
        }
        else{
              System.out.println("No son amigos");
        }
    }
}
```

Fibonacci en java

La serie de fibonacci la forman una serie de números tales que:

El primer término de la serie es el número 1
El segundo término de la serie es el número 1
Los siguientes términos de la serie de fibonacci se obtienen de la suma de los dos anteriores:

1, 1, 2, 3, 5, 8, 13, ...

Vamos a escribir el programa java que muestra los N primeros números de la serie. El valor de N se lee por teclado.

```
import java.util.*;
/**
 * Serie de Fibonacci en Java
 * Programa que imprima los N primeros números de la serie de Fibonacci.
 * El primer número de la serie es 1, el segundo número es 1 y cada uno de los
 * siguientes es la suma de los dos anteriores.
 * 1, 1, 2, 3, 5, 8, 13,  ..... , N
 * @author Enrique
 */
public class Main{
   public static void main(String[] args){
      Scanner sc = new Scanner(System.in);
      int numero,fibo1,fibo2,i;
      do{
         System.out.print("Introduce numero mayor que 1: ");
         numero = sc.nextInt();
      }while(numero<=1);
      System.out.println("Los " + numero + " primeros términos de la serie de Fibonacci
son:");

      fibo1=1;
      fibo2=1;

      System.out.print(fibo1 + " ");
      for(i=2;i<=numero;i++){
          System.out.print(fibo2 + " ");
          fibo2 = fibo1 + fibo2;
          fibo1 = fibo2 - fibo1;
      }
      System.out.println();
   }
}
```

Decimal a binario en java

En esta entrada vamos a escribir el programa java para convertir un número de decimal a binario.

Para escribir el programa nos vamos a basar en la forma clásica de pasar de decimal a binario, o sea, dividir el número entre 2 y quedarnos con el resto de la división. Esta cifra, que será un cero o un uno, es el dígito de menos peso (más a la derecha) del número binario. A continuación volvemos a dividir el cociente que hemos obtenido entre 2 y nos quedamos con el resto de la división. Esta cifra será la segunda por la derecha del número binario. Esta operación se repite hasta que obtengamos un cero como cociente.

De forma gráfica lo vamos a ver mucho más claro:

Si queremo convertir el número 12 en binario haremos las siguientes operaciones:

El número 12 en decimal es el 1100 en binario. El número binario se obtiene tomando los restos en orden inverso a como se han obtenido.

Los que ya sabéis algo de Java podeis pensar que para qué quiero hacer ese programa si simplemente escribiendo la instrucción:

```
System.out.println(Integer.toBinaryString(numero));
```

se mostrará el número en binario.
El método toBinaryString de la clase Integer ya me hace el trabajo, pero se trata de que seamos capaces de desarrollar por nosotros mismos el algoritmo que realiza la conversión de decimal a binario.

Este ejercicio se suele plantear cuando se está comenzando a aprender las estructuras repetitivas (while, for, do while) y aún no se conocen los arrays por lo que la solución que se plantea no utiliza arrays y por tanto esta solución aunque es correcta solo es válida para números enteros relativamente pequeños.

```
/**
 * Programa que pasa un número
 * de decimal a binario
 * @author Enrique García
 */
public class Main{

   public static void main(String[] args) {

      int numero, exp, digito;
      double binario;
      Scanner sc = new Scanner(System.in);

      do{
         System.out.print("Introduce un numero entero >= 0: ");
         numero = sc.nextInt();
      }while(numero<0);

      exp=0;
      binario=0;
      while(numero!=0){
            digito = numero % 2;
            binario = binario + digito * Math.pow(10, exp);
            exp++;
            numero = numero/2;
      }
      System.out.printf("Binario: %.0f %n", binario);
   }
}
```

Pasar un número de binario a decimal en Java

CONVERTIR UN NÚMERO DE BINARIO A DECIMAL EN JAVA

El programa para pasar un número expresado en binario a decimal se basa en la forma tradicional de hacerlo. Los dígitos del número binario ocupan una posición que se numera de derecha a izquierda empezando por cero. La posición del dígito más a la derecha es la 0.

Numero Binario:
1
1
0
1
0
1
Posición que ocupa cada dígito
5
4
3
2
1
0

Para pasar el número a decimal se multiplica cada dígito binario por 2 elevado a la posición que ocupa. La suma de todos los productos es el equivalente en decimal.

```
/*
* Programa Java que convierte un número binario a decimal
*/
import java.util.Scanner;

public class BinarioDecimal {

  public static void main(String[] args) {
    long numero, aux, digito, decimal;
     int exponente;
     boolean esBinario;
     Scanner sc = new Scanner(System.in);

    //Leer un número por teclado y comprobar que es binario
     do {
         System.out.print("Introduce un numero binario: ");
         numero = sc.nextLong();
         //comprobamos que sea un número binario es decir
```

```
            //que este formado solo por ceros y unos
            esBinario = true;
            aux = numero;
            while (aux != 0) {
                    digito = aux % 10; //última cifra del números
                    if (digito != 0 && digito != 1) { //si no es 0 ó 1
                        esBinario = false; //no es un número binario
                    }
                    aux = aux / 10; //quitamos la última cifra para repetir el proceso
             }
        } while (!esBinario); //se vuelve a pedir si no es binario

        //proceso para pasar de binario a decimal
        exponente = 0;
        decimal = 0; //será el equivalente en base decimal
        while (numero != 0) {
                //se toma la última cifra
                digito = numero % 10;
                //se multiplica por la potencia de 2 correspondiente y se suma al número
                decimal = decimal + digito * (int) Math.pow(2, exponente);
                //se aumenta el exponente
                exponente++;
                //se quita la última cifra para repetir el proceso
                numero = numero / 10;
        }
        System.out.println("Decimal: " + decimal);
    }
}
```

Convertir a números romanos en Java

Programa Java para convertir un número entero a números romanos.
El programa pide un número entre 1 y 3999 y calcula su equivalente en números romanos. Se utiliza un método llamado convertirANumerosRomanos que recibe el número N a convertir de tipo int y devuelve un String con el equivalente en números romanos.

Para convertirlo se obtiene por separado cada cifra del número y se muestran las combinaciones de letras del número romano equivalentes a cada cifra del número original. Este método no utiliza arrays de modo que este programa se puede resolver sin haber estudiado aún los arrays.

```
//Programa Java que lee un número y lo convierte a números romanos
import java.util.Scanner;

public class NumerosRomanos {

 public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    int N;
    do {
          System.out.print("Introduce un número entre 1 y 3999: ");
          N = sc.nextInt();
    } while (N < 1 || N > 3999);
    System.out.println(N + " en numeros romanos -> " + convertirANumerosRomanos(N));
 }

 //método para pasar a números romanos
 public static String convertirANumerosRomanos(int numero) {
    int i, miles, centenas, decenas, unidades;
    String romano = "";
   //obtenemos cada cifra del número
    miles = numero / 1000;
    centenas = numero / 100 % 10;
    decenas = numero / 10 % 10;
    unidades = numero % 10;

   //millar
    for (i = 1; i <= miles; i++) {
          romano = romano + "M";
    }

   //centenas
    if (centenas == 9) {
       romano = romano + "CM";
    } else if (centenas >= 5) {
                romano = romano + "D";
                for (i = 6; i <= centenas; i++) {
```

```
                romano = romano + "C";
            }
 } else if (centenas == 4) {
            romano = romano + "CD";
 } else {
          for (i = 1; i <= centenas; i++) {
              romano = romano + "C";
          }
 }

//decenas
 if (decenas == 9) {
    romano = romano + "XC";
 } else if (decenas >= 5) {
            romano = romano + "L";
            for (i = 6; i <= decenas; i++) {
                romano = romano + "X";
            }
 } else if (decenas == 4) {
            romano = romano + "XL";
 } else {
           for (i = 1; i <= decenas; i++) {
               romano = romano + "X";
           }
 }

//unidades
 if (unidades == 9) {
    romano = romano + "IX";
 } else if (unidades >= 5) {
            romano = romano + "V";
            for (i = 6; i <= unidades; i++) {
                romano = romano + "I";
            }
 } else if (unidades == 4) {
            romano = romano + "IV";
 } else {
          for (i = 1; i <= unidades; i++) {
              romano = romano + "I";
          }
 }
```

```
    return romano;
  }
}
```

Cifrado César en Java

Programa para codificar o decodificar un texto utilizando el método de cifrado de César. Supondremos que el texto solo contiene letras mayúsculas o minúsculas. La letras serán las correspondientes al alfabeto inglés (26 caracteres, excluimos la ñ y Ñ).
En este método de cifrado cada letra del texto se sustituye por otra letra que se encuentra N posiciones adelante en el alfabeto. Se considera que el alfabeto es circular, es decir, la letra siguiente a la ‘z’ es la ‘a’.
Por ejemplo, si N es 3, la ‘a’ se transformaría en ‘d’, la ‘b’ en ‘e’, la ‘c’ en ‘f’, etc.

Ejemplo de cifrado César: si el texto es “casa” y N = 3 el texto cifrado es “fdvd”

Para descifrar un texto se realiza la operación contraria. Se calcula la letra que está N posiciones por detrás en el alfabeto. Como el alfabeto es circular, la letra anterior a la ‘a’ es la ‘z’.

El programa pedirá por teclado un texto, a continuación el valor de N y si queremos codificar o decodificar el texto. Finalmente se mostrará el texto resultante.

Programa resuelto: Cifrado César en Java

```
import java.io.IOException;
import java.util.Scanner;

public class CifradoCesar {
```

```
    public static void main(String[] args) throws IOException {
        Scanner sc = new Scanner(System.in);
        String texto;
        int codigo;
        char opcion;
        //Introducir un texto
        do {
            System.out.print("Introduce un texto: ");
            texto = sc.nextLine();
        } while (texto.isEmpty());
        //Introducir el valor del desplazamiento
        do {
            System.out.print("Introduce el código: ");
            codigo = sc.nextInt();
        } while (codigo < 1);
        //Introducir la operación a realizar: cifrar o descifrar
        do {
            sc.nextLine();
            System.out.print("(C) cifrar o (D) descifrar?: ");
            opcion = (char) System.in.read();
        } while (Character.toUpperCase(opcion) != 'C' && Character.toUpperCase(opcion) !=
'D');
        if (Character.toUpperCase(opcion) == 'C') {
            System.out.println("Texto cifrado: " + cifradoCesar(texto, codigo));
        } else {
            System.out.println("Texto descifrado: " + descifradoCesar(texto, codigo));
        }
    }

    //método para cifrar el texto
    public static String cifradoCesar(String texto, int codigo) {
        StringBuilder cifrado = new StringBuilder();
        codigo = codigo % 26;
        for (int i = 0; i < texto.length(); i++) {
            if (texto.charAt(i) >= 'a' && texto.charAt(i) <= 'z') {
                if ((texto.charAt(i) + codigo) > 'z') {
                    cifrado.append((char) (texto.charAt(i) + codigo - 26));
                } else {
                    cifrado.append((char) (texto.charAt(i) + codigo));
                }
            } else if (texto.charAt(i) >= 'A' && texto.charAt(i) <= 'Z') {
```

```
            if ((texto.charAt(i) + codigo) > 'Z') {
                cifrado.append((char) (texto.charAt(i) + codigo - 26));
            } else {
                cifrado.append((char) (texto.charAt(i) + codigo));
            }
        }
    }
    return cifrado.toString();
}

//método para descifrar el texto
public static String descifradoCesar(String texto, int codigo) {
    StringBuilder cifrado = new StringBuilder();
    codigo = codigo % 26;
    for (int i = 0; i < texto.length(); i++) {
        if (texto.charAt(i) >= 'a' && texto.charAt(i) <= 'z') {
            if ((texto.charAt(i) - codigo) < 'a') {
                cifrado.append((char) (texto.charAt(i) - codigo + 26));
            } else {
                cifrado.append((char) (texto.charAt(i) - codigo));
            }
        } else if (texto.charAt(i) >= 'A' && texto.charAt(i) <= 'Z') {
            if ((texto.charAt(i) - codigo) < 'A') {
                cifrado.append((char) (texto.charAt(i) - codigo + 26));
            } else {
                cifrado.append((char) (texto.charAt(i) - codigo));
            }
        }
    }
    return cifrado.toString();
}
} //Fin cifrado Cesar
```

Ejemplos de ejecución:

Introduce un texto: Tengo el examen resuelto
Introduce el código: 4
(C) cifrar o (D) descifrar?: C
Texto cifrado: Xirksipibeqirviwyipxs

Introduce un texto: glgtekekqutguwgnvqu
Introduce el código: 2

(C) cifrar o (D) descifrar?: D
Texto descifrado: ejerciciosresueltos

Número capicúa en Java

COMPROBAR SI UN NÚMERO ES CAPICÚA EN JAVA

Un número es capicúa si se puede leer igual de derecha a izquierda que de izquierda a derecha. Ejemplos de números capicúas: 121, 3003, 1234321, 33, 445544, etc.

Vamos a escribir un programa Java que pida por teclado un número entero N de más de una cifra y verifique si es capicúa.

```
import java.util.Scanner;

public class Numero_Capicua {

   public static void main(String[] args) {
      int N, aux, inverso = 0, cifra;
      Scanner sc = new Scanner(System.in);
      do {
         System.out.print("Introduce un número >= 10: ");
         N = sc.nextInt();
      } while (N < 10);

      //le damos la vuelta al número
      aux = N;
      while (aux!=0){
         cifra = aux % 10;
         inverso = inverso * 10 + cifra;
         aux = aux / 10;
      }

      if(N == inverso){
         System.out.println("Es capicua");
      }else{
         System.out.println("No es capicua");
      }
   }
```

}

Para resolverlo, lo que hemos hecho es darle la vuelta al número y comprobar si el número invertido es igual al original.
El procedimiento para obtener el número invertido, al que hemos llamado inverso, es el siguiente:
Si por ejemplo N = 121
inverso = 0
Repetimos estos pasos hasta que N valga 0:
Obtenemos la última cifra de N: cifra = N % 10 -> cifra = 121 % 10 -> cifra = 1
Se lo sumamos al contenido de inverso multiplicado por 10
inverso = inverso * 10 + cifra -> inverso = 0 * 10 + 1 -> inverso = 1
Quitamos la última cifra a N: N = N / 10 -> N = 121 / 10 -> N = 12
Repetimos el proceso:
Obtenemos la última cifra de N: cifra = N % 10 -> cifra = 12 % 10 -> cifra = 2
Se lo sumamos al contenido de inverso multiplicado por 10
inverso = inverso * 10 + cifra -> inverso = 1 * 10 + 2 -> inverso = 12
Quitamos la última cifra a N: N = N / 10 -> N = 12 / 10 -> N = 1
Repetimos el proceso:
Obtenemos la última cifra de N: cifra = N % 10 -> cifra = 1 % 10 -> cifra = 1
Se lo sumamos al contenido de inverso multiplicado por 10
inverso = inverso * 10 + cifra -> inverso = 12 * 10 + 1 -> inverso = 121
Quitamos la última cifra a N: N = N / 10 -> N = 1 / 10 -> N = 0

El proceso para invertir el número finaliza cuando N = 0. Ahora comprobamos si el número invertido es igual al original. En ese caso el número es capicúa.

Ejemplo de ArrayList en Java

Ejemplo de uso de un ArrayList: Calcular la altura media de los alumnos de una clase
Programa Java que pida por teclado las alturas de N alumnos de una clase y las guarde en un

ArrayList de tipo Double. A continuación el programa calculará la altura media de todos los alumnos, cuantos alumnos hay más altos que la media y cuantos más bajos.
Para resolverlo vamos a utilizar 4 métodos además del método main:
Método numeroAlumnos(): este método pide por teclado el número de alumnos de la clase y devuelve dicho número al programa principal.
Método leerAlturas(): pide por teclado las alturas de los N alumnos y las almacena en el ArrayList. Este método recibe como parámetros el ArrayList inicialmente vacío y el número de alumnos a leer.
Método calcularMedias(): calcula y devuelve la media de los alumnos de la clase. Este método recibe como parámetro el ArrayList con las alturas de todos los alumnos.
Método mostrarResultados(): muestra por pantalla todas las alturas y calcula y muestra el número de alumnos con altura superior e inferior a la media. Recibe como parámetros el ArrayList con las alturas de todos los alumnos y la media calculada anteriormente.
Solución:

```
import java.util.ArrayList;
import java.util.Scanner;

public class Main {

  public static void main(String[] args) {
    //creamos el ArrayList que contendrá la altura de los alumnos
    ArrayList<Double> altura = new ArrayList();
    int N;
    double media;
    //obtenemos el número de alumnos de la clase
    N = numeroAlumnos();
    //leemos la altura de los N alumnos
    leerAlturas(altura, N);
    //calculamos la media
    media = calcularMedia(altura);
    //mostramos los resultados
    mostrarResultados(altura, media);
  }

  //Método para pedir por teclado el número de alumnos de la clase
  public static int numeroAlumnos() {
    Scanner sc = new Scanner(System.in);
    int n;
    do {
        System.out.print("Introduce número de alumnos: ");
        n = sc.nextInt();
```

```
    } while (n < 1);
    return n;
  }

  //Este método recibe el ArrayList y el número de alumnos de la clase
  //Pide por teclado la altura de todos los alumnos y las guarda en el ArrayList
  public static void leerAlturas(ArrayList<Double> a, int n) {
    Scanner sc = new Scanner(System.in);
    int i;
    double alto;
    for (i = 1; i <= n; i++) {
        do {
            System.out.print("Alumno " + i + " altura: ");
            alto = sc.nextDouble();
        } while (alto <= 0);
        a.add(alto); //añade la altura al final del ArrayList
    }
  }

  //Este método recibe el ArrayList con todas las alturas
  //calcula y devuelve la media
  public static double calcularMedia(ArrayList<Double> a) {
    double media = 0;
    for (Double d : a) {
         media = media + d;
    }
    return media / a.size();
  }

  //Muestra la altura de todos los alumnos, la media y calcula y muestra
  //cuantos alumnos hay con altura superior a la media
  //y cuántos con altura inferior
  public static void mostrarResultados(ArrayList<Double> a, double media) {
    int superior = 0, inferior = 0;
    System.out.println("alturas introducidas: ");
    System.out.println(a);
    for (Double d : a) {
        if (d > media)
           superior++;
        else if (d < media)
                inferior++;
```

```
    }
    System.out.printf("Media: %.2f %n", media);
    System.out.println("Hay " + superior + " alumnos más altos que la media");
    System.out.println("Hay " + inferior + " alumnos más bajos que la media");
  }
}
```

CALCULAR LA SUMA Y LA MEDIA ARITMÉTICA DE LOS VALORES CONTENIDOS EN UN ARRAYLIST.

Programa que lea una serie de valores numéricos enteros desde el teclado y los guarde en un ArrayList de tipo Integer. La lectura de números termina cuando se introduzca el valor -99. Este valor no se guarda en el ArrayList. A continuación el programa mostrará por pantalla el número de valores que se han leído, su suma, su media. Por último se mostrarán todos los valores leídos, indicando cuántos de ellos son mayores que la media.

Vamos a utilizar 3 métodos además del método main para resolverlo:

Método leerValores(): pide por teclado los números y los almacena en el ArrayList. La lectura acaba cuando se introduce el valor -99. El método devuelve mediante return el ArrayList con los valores introducidos.

Método calcularSuma(): Recibe como parámetro el ArrayList con los valores numéricos y calcula y devuelve su suma. En este método se utiliza un Iterator para recorrer el ArrayList.

Método mostrarResultados(): Recibe como parámetro el ArrayList, la suma y la media aritmética. Muestra por pantalla todos los valores, su suma y su media y calcula y muestra cuantos números son superiores a la media. En este método se utiliza un for para colecciones para recorrer el ArrayList.

```
import java.util.ArrayList;
import java.util.Iterator;
import java.util.Scanner;

public class Main {

  public static void main(String[] args) {
      ArrayList<Integer> array = leerValores();
      double suma = calcularSuma(array);
      double media = suma / array.size();
      mostrarResultados(array, suma, media);
  }

  //pedir por teclado los valores y guardarlos en el ArrayList
```

```
  //la lectura acaba cuando se introduzca -99
  public static ArrayList<Integer> leerValores() {
      ArrayList<Integer> valores = new ArrayList();
      Scanner sc = new Scanner(System.in);
      int n;
      System.out.print("Introduce entero. -99 para acabar: ");
      n = sc.nextInt();
      while (n != -99) {
            valores.add(n);
            System.out.print("Introduce entero. -99 para acabar: ");
            n = sc.nextInt();
      }
      return valores;
  }

  //recorrer el ArrayList y sumar todos sus elementos
  public static double calcularSuma(ArrayList<Integer> valores) {
       double suma = 0;
       Iterator it = valores.iterator();
       while (it.hasNext()) {
             suma = suma + (Integer) it.next();
            //next() devuelve un dato de tipo Object. Hay que convertirlo a Integer
       }
       return suma;
  }

  //Mostrar valores, su suma y su media aritmética
  //y cuántos hay superiores a la media
  public static void mostrarResultados(ArrayList<Integer> valores, double suma, double
media) {
       int cont = 0;
       System.out.println("Valores introducidos: ");
       System.out.println(valores);
       System.out.println("Suma: " + suma);
       System.out.printf("Media: %.2f %n", media);
       for (Integer i : valores) {
            if (i > media) {
              cont++;
            }
       }
     System.out.println(cont + " valores superiores a la media");
```

```
  }
}
```

Rotar los elementos de un ArrayList en Java

Método para desplazar todos los componentes de un Array un lugar a la derecha. El último elemento pasará a la primera posición.
Ejemplo, si el Array original contiene los siguientes valores:

15
22
4
56
71
10
2

Se realizarán los siguientes desplazamientos:

De forma que el contenido final de Array sea este:

2
15
22
4
56
71
10

En este caso el contenedor será un ArrayList. El método recibirá un ArrayList de tipo Integer con una serie de valores numéricos y devolverá el mismo ArrayList con sus elementos desplazados una posición a la derecha. Los desplazamientos se realizarán sobre el mismo array. No se puede utilizar un array auxiliar para reailzar el proceso.
Solución:
Vamos a resolverlo de dos maneras. La primera será el método que podríamos llamar clásico

para rotar las componentes de un array que consiste en guardar el último elemento del array en una variable auxiliar, a continuación desplazar todos los elementos una posición a la derecha desde el final hasta el principio del array y por último asignar al primer elemento el valor guardado en el auxiliar.
Este es el código Java de este primer método de desplazamiento:

```
public static void desplazarDerecha(ArrayList<Integer> a) {
      int i;
      int aux = a.get(a.size() - 1); //guardar el último elemento en una variable
      for (i = a.size() - 1; i > 0; i--) { //desplazar los elementos
         a.set(i, a.get(i - 1)); //a cada elemento se le asigna el anterior
      }
      a.set(0, aux); //asignar al primero el último que se guardó al principio
}
```

La segunda forma de resolverlo es utilizando dos métodos de la clase ArrayList:
add(posición, valor);
remove(posición);
Con el método add insertamos al principio del ArrayList el último valor. Insertar en una posición significa crear un elemento nuevo en esa posición y desplazar el resto de los elementos hacia la derecha.
Por ejemplo, si el array es este:

15
22
4
56
71
10
2

Al insertar el último elemento al principio, el contenido del array ahora es este:

2
15
22
4
56
71

10
2

Solo queda eliminar el último elemento y tendremos el resultado:

2
15
22
4
56
71
10

Este es el código Java de este segundo método de desplazamiento:

```
public static void desplazarDerecha2(ArrayList<Integer> a) {
        int aux = a.get(a.size()-1);  //guardar el último elemento en una variable
        a.add(0,aux);                 //insertar al principio el último valor
        a.remove(a.size()-1);         //eliminar el último elemento
}
```

Programa completo para utilizar estos métodos:

```
import java.util.ArrayList;
import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        ArrayList<Integer> A = leerArray();
        mostrar(A);
        //utilizando el método clásico
        desplazarDerecha(A);
        mostrar(A);
        //utilizando solo los métodos de ArrayList
        desplazarDerecha2(A);
        mostrar(A);
    }

    //pedir por teclado los valores del ArrayList
    public static ArrayList<Integer> leerArray() {
        Scanner sc = new Scanner(System.in);
        ArrayList<Integer> numeros = new ArrayList();
        int N;
```

```
        do {
            System.out.print("Número de elementos del array (>0): ");
            N = sc.nextInt();
        } while (N <= 0);
        for (int i = 0; i < N; i++) {
            System.out.print("elemento[" + i + "]= ");
            numeros.add(sc.nextInt());
        }
        return numeros;
    }

    //método clásico: guardamos el último elemento en un auxiliar,
    //desplazamos todos los elementos una posición a la derecha
    //finalmente guardamos en el primer elemento el valor guardado en el auxiliar
    public static void desplazarDerecha(ArrayList<Integer> a) {
        int i;
        int aux = a.get(a.size() - 1); //guardar el último elemento en una variable
        for (i = a.size() - 1; i > 0; i--) { //desplazar los elementos
            a.set(i, a.get(i - 1)); //a cada elemento se le asigna el anterior
        }
        a.set(0, aux); //asignar al primero el último que se guardó al principio
    }

    //Utilizando métodos de ArrayList:
    //guardamos el último elemento en una variable
    //insertamos este elemento al principio
    //eliminamos el último elemento
    public static void desplazarDerecha2(ArrayList<Integer> a) {
        int aux = a.get(a.size()-1);
        a.add(0,aux);
        a.remove(a.size()-1);
    }

    //Método para mostrar todo el ArrayList
    public static void mostrar(ArrayList<Integer> a) {
        System.out.println(a);
    }
}
```

Calcular la cadena más larga en un ArrayList de Strings

Programa Java para calcular el String de mayor longitud de todos los contenidos en un ArrayList de String. El programa utilizará los siguientes métodos además del método main:
Método leerArray(): este método recibe como parámetro el arrayList de Strings vacío. El método pide por teclado cadenas de caracteres y las añade al ArrayList. La lectura de cadenas termina cuando se introduce la palabra “FIN”.
Método cadenaMasLarga():este método recibe como parámetro el ArrayList de Strings con todas las cadenas leídas anteriormente y devuelve el String de mayor longitud.

Solución:
Programa que calcula el String de mayor longitud en un ArrayList de Strings.

```
import java.util.ArrayList;
import java.util.Scanner;
public class Main {

   public static void main(String[] args) {
      ArrayList<String> cadenas = new ArrayList();
      leerArray(cadenas);
      System.out.println("Cadena de mayor longitud : " + cadenaMasLarga(cadenas));
   }

   //llenar el ArrayLIst con Strings introducidos por teclado
   public static void leerArray(ArrayList<String> cadenas) {
      Scanner sc = new Scanner(System.in);
      String s;
      boolean masCadenas;
      do {
         masCadenas = true;
         System.out.print("Introduce una cadena de caracteres (Fin para acabar): ");
         s = sc.nextLine();
         if (s.equalsIgnoreCase("FIN")) {
            masCadenas = false;
         } else {
```

```
            cadenas.add(s);
         }
      } while (masCadenas);
   }

   //Calcular y devolver la cadena de mayor longitud
   public static String cadenaMasLarga(ArrayList<String> cadenas) {
      String mayor = cadenas.get(0);
      for (int i = 1; i < cadenas.size(); i++) {
         if (cadenas.get(i).length() > mayor.length()) {
            mayor = cadenas.get(i);
         }
      }
      return mayor;
   }
} //Fin Clase Principal
```

Contar el número de palabras de una frase en Java

Programa Java para contar el número de palabras que contiene una frase.
El programa lee un texto por teclado y lo guarda en un String. A continuación mostrará el número de palabras que contiene.
La forma más sencilla de resolverlo es utilizando la clase StringTokenizer. Esta clase sirve para dividir un String en partes, según unos delimitadores. Uno de estos delimitadores es el espacio en blanco, por lo tanto podemos aplicar StringTokenizer al texto ya que las palabras en un texto están separadas por espacios en blanco. El método countTokens() nos dirá cuantos elementos se han obtenido o lo que es lo mismo, cuantas palabras contiene el texto.

```
//Programa que cuenta el número de palabras de un texto.
import java.util.Scanner;
import java.util.StringTokenizer;
public class ContarPalabras {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      String frase;
      System.out.print("Introduce una frase: ");
```

```
        frase = sc.nextLine();
        StringTokenizer st = new StringTokenizer(frase);
        System.out.println("Número de palabras: " + st.countTokens());
    }
}
```

Como alternativa a utilizar la clase StringTokenizer, podemos resolverlo utilizando solo los métodos de la clase String. En este caso se tratará de buscar los espacios en blanco dentro de la cadena y contarlos.
Supondremos que las palabras están separadas entre ellas por un solo espacio en blanco. Para encontrar los espacios en blanco podemos usar el método indexOf(caracter) que devuelve la posición donde se ha encontrado el carácter dentro de la cadena. Si el carácter no se encuentra devuelve -1. También se puede usar indexOf (carácter, p) en este caso busca el carácter a partir de una posición p.

```
//Método que recibe un String y devuelve el número de palabras que contiene
public static int contarPalabras(String s) {
    int contador = 1, pos;
    s = s.trim(); //eliminar los posibles espacios en blanco al principio y al final
    if (s.isEmpty()) { //si la cadena está vacía
        contador = 0;
    } else {
            pos = s.indexOf(" "); //se busca el primer espacio en blanco
            while (pos != -1) { //mientras que se encuentre un espacio en blanco
                    contador++; //se cuenta una palabra
                    pos = s.indexOf(" ", pos + 1); //se busca el siguiente espacio en blanco
            }                                       //a continuación del actual
    }
    return contador;
}
```

Eliminar la última palabra de un String en Java

MÉTODO JAVA PARA QUITAR LA ÚLTIMA PALABRA DE UN STRING.
Vamos a escribir un método que reciba un String que contiene una frase. El método eliminará

la última palabra y devuelve un String con la frase modificada.
La forma de resolverlo es la siguiente: se busca el último espacio en blanco del texto mediante el método lastIndexOf. Este método devuelve la posición del último espacio en blanco. Si lastIndexOf devuelve -1 significa que no lo ha encontrado lo que quiere decir que la frase solo tiene una palabra o está vacía. En ese caso se devuelve una cadena vacía.
Si se encuentra el último espacio en blanco, se obtiene un nuevo String mediante el método substring desde la primera posición del texto hasta donde se encuentra el último espacio.
Para que esto funcione correctamente debemos asegurarnos de que el texto no contiene espacios en blanco al final ya que si fuese así no encontraríamos la última palabra con este método. Para eso se utiliza el método trim() que elimina posibles espacios en blanco al principio y al final del String.

```
public static String eliminarUltimaPalabra(String s) {
   int pos;
   s = s.trim();
   pos = s.lastIndexOf(" ");
   if (pos != -1) {
       s = s.substring(0, pos);
   } else {
       s = "";
   }
   return s;
}
```

Un método main para probarlo podría ser este:

```
public static void main(String[] args) {
   Scanner sc = new Scanner(System.in);
   String texto;
   do {
        System.out.println("Introduce texto: ");
        texto = sc.nextLine();
    } while (texto.isEmpty());
    texto = eliminarUltimaPalabra(texto);
    System.out.println(texto);
}
```

Se pide por teclado un texto, mediante el método isEmpty() se obliga a que el texto no esté vacío aunque esto no es necesario. Después se envía el String que contiene el texto al método para que elimine la última palabra y finalmete se muestra el String modificado.

Contar el número de veces que aparece un carácter en un texto en Java

PROGRAMA JAVA QUE CALCULE EL NÚMERO DE VECES QUE APARECE UN DETERMINADO CARÁCTER DENTRO DE UN TEXTO.

Este es un ejemplo típico que se plantea para trabajar con cadenas de caracteres y hacer uso de los métodos de la clase String de Java. En concreto en este caso para buscar el carácter dentro del texto utilizaremos el método indexOf de String. El método indexOf(carácter) devuelve la primera posición dentro del String donde se encuentra el carácter o -1 si no lo ha encontrado. También se puede invocar al método con dos parámetros indexOf(carácter, posición) en este caso la búsqueda del carácter dentro del String comienza a partir de la posición que se le envía como parámetro. Devuelve la primera posición donde se encuentra o -1 si no lo ha encontrado.

Para resolverlo escribiremos un método llamado contarCaracteres que reciba el texto (String) y el carácter (char) a buscar y devuelva el número de veces que se repite el carácter dentro del texto.

El proceso utilizado para buscar el carácter en el texto es el siguiente:

Se busca la primera vez que aparece el carácter con el método indexOf:

posicion = cadena.indexOf(caracter);

Si el carácter se ha encontrado se suma 1 a un contador y se sigue buscando a partir de la posición siguiente a la hallada:

posicion = cadena.indexOf(caracter, posicion + 1);

Este proceso se repite mientras el carácter se encuentre en la cadena. La búsqueda termina cuando indexOf devuelva -1 como posición.

```
//método para calcular el número de veces que se repite un carácter en un String
public static int contarCaracteres(String cadena, char caracter) {
    int posicion, contador = 0;
    //se busca la primera vez que aparece
    posicion = cadena.indexOf(caracter);
    while (posicion != -1) { //mientras se encuentre el caracter
       contador++;          //se cuenta
       //se sigue buscando a partir de la posición siguiente a la encontrada
       posicion = cadena.indexOf(caracter, posicion + 1);
    }
    return contador;
```

```
}
```

Un programa completo para utilizar este método puede ser el siguiente:

```
import java.io.IOException;
import java.util.Scanner;

public class Main {

    public static void main(String[] args) throws IOException {
        Scanner sc = new Scanner(System.in);
        String texto;
        char caracter;
        int numeroDeVeces = 0;
        do {
            System.out.println("Introduce texto: ");
            texto = sc.nextLine();
        } while (texto.isEmpty());
        System.out.print("Introduce un carácter: ");
        caracter = (char) System.in.read();
        numeroDeVeces = contarCaracteres(texto, caracter);
        System.out.println("El caracter " + caracter + " aparece " + numeroDeVeces + " veces");
    }

    //calcular el número de veces que se repite un carácter en un String
    public static int contarCaracteres(String cadena, char caracter) {
        int posicion, contador = 0;
        //se busca la primera vez que aparece
        posicion = cadena.indexOf(caracter);
        while (posicion != -1) { //mientras se encuentre el caracter
            contador++;          //se cuenta
            //se sigue buscando a partir de la posición siguiente a la encontrada
            posicion = cadena.indexOf(caracter, posicion + 1);
        }
        return contador;
    }
}
```

Ejemplos de recursividad. Calcular el cociente de dos números enteros.

Programa Java que calcule el cociente de dos números enteros de forma recursiva.

La solución recursiva del problema se plantea de la siguiente forma:

Caso Base: Si el dividendo es menor que el divisor el cociente es cero.

Si no, se restan el dividendo y el divisor. Al resultado se le vuelve a restar el divisor. Esta resta se repite mientras se pueda realizar, o sea, mientras el resultado sea mayor o igual que el divisor.
El número de veces que se ha hecho la resta es el cociente.

Con un ejemplo lo veremos más claro. Por ejemplo, para dividir 10 entre 3 haríamos:

10 - 3 = 7
7 - 3 = 4
4 - 3 = 1

No se puede seguir restando ya que el último valor obtenido (1) es menor que el divisor.
Como se han realizado 3 restas el cociente es 3.

```
import java.util.*;
public class Recursividad {

   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      int n1, n2;
      System.out.print("Introduzca dividendo: ");
      n1 = sc.nextInt();
      do {
         System.out.print("Introduzca divisor (>0): ");
         n2 = sc.nextInt();
      } while (n1 <= 0);
      System.out.printf("%d/%d = %d %n", n1, n2, cociente(n1, n2));
   }
```

```
    public static int cociente(int a, int b) {
        if (a < b)
            return 0;
        else
            return 1 + cociente(a - b, b);
    }
}
```

Ejemplos de recursividad. Pasar de Decimal a Binario de forma recursiva

Programa que lea un número entero entero mayor o igual que cero en base decimal y muestre su equivalente en binario de forma recursiva

El caso base se obtiene cuando el número es 0 ó 1. En ese caso el número binario equivalente es el mismo.

Si no, se hace una llamada recursiva al método, enviándole n/2.

Cuando en esas llamadas recursivas se envíe un 0 o un 1 se mostrará ese valor y a continuación se ejecutará la instrucción System.out.print(n % 2); que imprimirá el resto de la división en cada momento de la ejecucicón.

Para entender mejor como se producen la secuencia de llamadas recursivas puedes ver de forma gráfica el cálculo del factorial en la imagen de esta página.

```
import java.util.*;

/**
  * Método recursivo que dado un número entero en base decimal
 * muestre su equivalente en binario
 */
public class Recursividad {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n;
        do {
```

```
            System.out.print("Introduzca numero >0: ");
            n = sc.nextInt();
        } while (n < 0);
        System.out.println();
        System.out.print("Binario: ");
        decBin(n);
        System.out.println();
    }

    public static void decBin(int n) {
        if (n < 2) {
            System.out.print(n);
            return;
        } else {
            decBin(n / 2);
            System.out.print(n % 2);
            return;
        }
    }
}
```

Ejemplos de Recursividad. Calcular 2 elevado a n

Programa Java recursivo para calcular 2 elevado a n siendo n un número entero mayor o igual que 0.

La solución recursiva se basa en lo siguiente:

Caso Base:

si n = 0 entonces 2^0 = 1

Si n > 0 Podemos calcular 2^n como 2 * 2^n-1

Por ejemplo: 2^5 = 2 * 2^4

El programa que calcula 2 elevado a n de forma recursiva puede quedar así:

```
//Calcular 2 elevado a n de forma recursiva

import java.util.*;
public class Elevar2aN {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      int num;
      do{
        System.out.print("Introduce un numero entero >=0 ");
        num = sc.nextInt();
      }while(num<0);
      System.out.println("2 ^ " + num + " = " + potencia(num));
   }
   public static double potencia(int n){
        if(n==0)   //caso base
          return 1;
        else
          return 2 * potencia(n-1);
   }
}
```

La solución iterativa a este problema sería esta:

```
//Calcular 2 elevado a n de forma iterativa
import java.util.*;
public class Potencia2 {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      int num;
      do{
        System.out.print("Introduce un numero entero >=0 ");
        num = sc.nextInt();
      }while(num<0);
      System.out.println("2 ^ " + num + " = " + potencia(num));
   }
   public static double potencia(int n){
```

```
        double resultado=1;
        int i;
        for(i=1;i<=n;i++)
           resultado = resultado * 2;
        return resultado;
   } }
```

Ejemplos de Recursividad. Contar las cifras de un numero entero de forma recursiva

Programa java que calcula el número de cifras de un número entero de forma recursiva.

La solución recursiva se basa en lo siguiente:

Caso Base:

Si n < 10 el número tiene 1 cifra

Si n >=10 tiene las cifras de un número con una cifra menos, más 1.

Es decir, si por ejemplo el número es 1234, el número de cifras es las suma de cifras del número 123 más una.

El programa que calcula el número de cifras de un entero de forma recursiva es el siguiente:

```
import java.util.*;
public class CuentaCifras {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      int num;
      do{
        System.out.print("Introduce un numero entero >0 ");
        num = sc.nextInt();
      }while(num<=0);
      System.out.println("Número de cifras: " + numeroCifras(num));
   }
   public static int numeroCifras(int n){
        if(n < 10)  //caso base
```

```
            return 1;
          else
            return 1 + numeroCifras(n/10);
    }
}
```

Ejemplos de Recursividad. Suma Recursiva de números desde 1 hasta N en Java

Programa recursivo que calcula la suma desde 1 hasta un número entero N leído por teclado.

La solución recursiva se basa en lo siguiente:

Caso Base: Si n == 1 la suma es 1

Si n > 1
la suma es n + (la suma de los anteriores números hasta 1)
Por ejemplo, si n = 5, la suma es 5 más la suma desde 1 hasta 4. A su vez la suma si n = 4 es 4 + la suma desde 1 hasta 3 y así sucesivamente hasta llegar al caso base como se muestra en la imagen.

El programa que calcula la suma desde 1 hasta N de forma recursiva es el siguiente:

```
import java.util.*;
public class Suma1N {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int num;
        do{
          System.out.print("Introduce un numero entero >0 ");
          num = sc.nextInt();
        }while(num<=0);
        System.out.println("Suma desde 1 hasta " + num + " = " + suma1N(num));

    }
```

```
   //método recursivo para calcular la suma desde 1 hasta N
   public static double suma1N(int n){
         if(n == 1)  //caso base
           return 1;
         else
           return n + suma1N(n-1);
   }
}
```

Ejemplos de recursividad. Sumar dos números enteros

Método java que calcula la suma de dos números enteros de forma recursiva.

La solución recursiva para sumar dos números enteros se basa en lo siguiente:

Caso Base:

Si uno de los números es igual a cero, la suma es el otro número.

Por ejemplo, si queremos sumar a=3 y b=0 la suma es igual a a: suma=3

```
si a == 0
  la suma es b
sino si b == 0
        la suma es a
```

Procedimiento recursivo:

Si ninguno de los dos números a sumar es igual a cero, la suma de ambos la podemos expresar como:

suma = 1 + suma(a, (b-1))

Por ejemplo:
Dados los números a=3 y b =4
la suma de 3 + 4 es igual que sumar 1 + (3 + 3)
A su vez, sumar 3 + 3 es igual que 1 + (3 + 2)

Si repetimos el proceso hasta que b sea 0 obtendremos la suma de forma recursiva:

El código del método es el siguiente:

```
/*
 *   Método recursivo que calcula la suma de dos números enteros
 */
public static int suma(int a, int b) {
   if (b == 0) {
      return a;
   } else if (a == 0) {
            return b;
   } else {
      return 1 + suma(a, b - 1);
   }
}
```

Como ejemplo de uso vamos a escribir un programa que lee dos números enteros y muestra la suma utilizando el método recursivo:

```
import java.util.*;
public class Recursividad1 {

   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      int n1, n2;
      System.out.print("Introduzca primer numero: ");
      n1 = sc.nextInt();
      System.out.print("Introduzca segundo numero: ");
      n2 = sc.nextInt();
      System.out.println("suma: " + suma(n1, n2));

   }
   public static int suma(int a, int b) {
      if (b == 0) {
```

```
        return a;
    } else if (a == 0) {
        return b;
    } else {
        return 1 + suma(a, b - 1);
    }
}
```

Calcular el resto de la división de forma recursiva

Programa que calcula el resto de la división de dos números enteros de forma recursiva.
El planteamiento para resolverlo es el siguiente:
Caso Base: Si el dividendo es menor que el divisor, el resto es el dividendo.
Si el caso base no se cumple, entonces se resta el dividendo y el divisor. A este resultado se le vuelve a restar el divisor. Este proceso se repite hasta que se llega al caso base, es decir, cuando el resultado obtenido al restar sea menor que el divisor.
Con un ejemplo quedará más claro:
Supongamos que queremos calcular el resto de dividir 10 entre 3:
Restamos 10 – 3 Resultado 7
Restamos 7 – 3 Resultado: 4
Restamos 4 -3 Resultado: 1
Como 1 es menor que 3 no se realiza la operación. El resto de la división es 1.

```
public class EjemploRecursividad {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n1, n2;
        System.out.print("Introduzca dividendo: ");
        n1 = sc.nextInt();
        do {
                System.out.print("Introduzca divisor (>0): ");
                n2 = sc.nextInt();
        } while (n1 <= 0);
        System.out.printf("%d%%%d = %d %n", n1, n2, restoRecursivo(n1, n2));
    }

    //método que calcula el resto de la división de forma recursiva
    public static int restoRecursivo(int a, int b) {
```

```
        if (a < b) {
            return a;
        } else {
             return resto(a - b, b);
        }
    }
}
```

Mostrar las unidades de disco del sistema

Programa que muestre todas las unidades de disco del ordenador indicando para cada una de ellas el tamaño disponible y el tamaño total.
Solución: Utilizando la clase File de Java podemos obtener las unidades del sistema y además podemos obtener información sobre esas unidades como por ejemplo el espacio total de la unidad o el espacio disponible.
Para obtener las unidades de disco del sistema, la clase File proporciona el método estático listRoots()
public static File[] listRoots()
El método listRoots() devuelve un array de objetos de tipo File que representan el directorio raíz de cada una de las unidades de disco del sistema.
Además la clase File proporciona los métodos getTotalSpace y getFreeSpace que devuelven el espacio total de la unidad y el espacio libre respectivamente.
Utilizando estos tres métodos de File podemos resolver el ejercicio:

```
//Ejemplo de utilización de la clase File para obtener información sobre las unidades del sistema
import java.io.File;
public class EjercicioFileUnidades {
    public static void main(String[] args) {
        File [] unidades = File.listRoots();
        System.out.printf("   %20s %20s %n" , "Tamaño Total", "Tamaño disponible");
        for(File f : unidades){
            System.out.print(f);
            System.out.printf("%20s %20s %n", f.getTotalSpace(), f.getFreeSpace());
        }
    }
}
```

La instrucción File [] unidades = File.listRoots(); obtiene el array de unidades
A continuación mediante una instrucción for se recorre el array y para cada elemento se obtiene su espacio total y su espacio disponible.
Si se ejecuta este programa en un sistema Windows obtendremos una salida similar a esta:

	Tamaño Total	Tamaño disponible
C:\	488261021696	303251963904
D:\	10256117760	1201942528
E:\	0	0
G:\	31994183680	3702784000

Mostrar el contenido de un directorio

Programa que muestre el contenido de un directorio o carpeta. Se deben mostar los nombres de los archivos y los directorios que contiene en orden alfababético.
Solución:
Primero creamos un objeto File que va a representar el directorio o carpeta a mostrar. Por ejemplo, en un sistema Windows si queremos mostrar el contenido de la unidad C: escribimos la siguiente instrucción:
File directorio = new File("c:/");
Una vez creado el objeto, podemos utilizar el método list() de la clase File. El método list() devuelve un array de String con el nombre de todos los archivos y directorios que contiene el objeto que lo invoca.
String[] lista = directorio.list();
Ordenamos alfabéticamente el array lista mediante el método Arrays.sort:
Arrays.sort(lista);
Por último mostramos el contenido del array:

```
for (int i = 0; i < lista.length; i++) {
        System.out.println(lista[i]);
}
```

El programa completo es este:

```
//Mostrar el contenido de un directorio en Java
import java.io.File;
import java.util.Arrays;

public class MostrarDirectorio {
```

```
    public static void main(String[] args) {
        File directorio = new File("c:/"); //directorio a listar
        String[] lista = directorio.list();
        Arrays.sort(lista);
        for (int i = 0; i < lista.length; i++) {
            System.out.println(lista[i]);
        }
    }
}
```

Ficheros de texto en Java

Un fichero de texto está formado por secuencias de caracteres, organizados en líneas de igual o distinta longitud.
Todos los datos que aparecen en estos ficheros están formados por caracteres.
CREAR Y ESCRIBIR EN FICHEROS DE TEXTO EN JAVA
Para escribir en un fichero de texto utilizaremos dos clases:
FileWriter y PrintWriter.
La clase FileWriter permite tener acceso al fichero en modo escritura.
Para crear objetos FileWriter podemos utilizar los constructores:

FileWriter(String path)
FileWriter(File objetoFile);

El fichero se crea y si ya existe su contenido se pierde.
Si lo que necesitamos es abrir un fichero de texto existente sin perder su contenido y añadir más contenido al final utilizaremos los constructores:

FileWriter(String path, boolean append)
FileWriter(File objetoFile, boolean append)

Si el parámetro append es true significa que los datos se van a añadir a los existentes. Si es false los datos existentes se pierden.
La clase FileWriter proporciona el método write() para escribir cadenas de caracteres aunque lo normal es utilizar esta clase junto con la clase PrintWriter para facilitar la escritura.
La clase PrintWriter permite escribir caracteres en el fichero de la misma forma que en la

pantalla.
Un objeto PrintWriter se crea a partir de un objeto FileWriter.
Ejemplo:
FileWriter fw = new FileWriter("c:/ficheros/datos.txt");
PrintWriter salida = new PrintWriter(fw);
En este ejemplo se ha creado un fichero de texto llamado datos.txt. El fichero se encuentra dentro de la carpeta ficheros en la unidad C:

A partir de Java 5 se puede crear un objeto PrintWriter directamente a partir de un objeto File o de la ruta:
PrintWriter salida = new PrintWriter("c:/ficheros/datos.txt");
En este caso, si el fichero no existe se crea. Si no se puede crear un archivo con ese nombre o si ocurre algún error se lanza una excepción FileNotFoundException.
Una vez creado el objeto podemos utilizar print(), println() y printf() para escribir en el fichero como si fuese en pantalla.

Ejemplo de escritura de un fichero de texto:
Programa Java que lee texto por teclado y lo escribe en un fichero de texto llamado datos.txt. El proceso consiste en leer una línea de texto por teclado y escribirla en el fichero. Este proceso se repite hasta que se introduce por teclado la cadena FIN. La cadena FIN que indica el final de lectura no se debe escribir en el fichero.

```
import java.io.FileNotFoundException;
import java.io.PrintWriter;
import java.util.Scanner;
public class File11 {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      PrintWriter salida = null;
      try {
         salida = new PrintWriter("c:/ficheros/datos.txt"); //se crea el fichero
         String cadena;
         System.out.println("Introduce texto. Para acabar introduce la cadena FIN:");
         cadena = sc.nextLine(); //se introduce por teclado una cadena de texto
         while (!cadena.equalsIgnoreCase("FIN")) {
            salida.println(cadena);  //se escribe la cadena en el fichero
            cadena = sc.nextLine(); //se introduce por teclado una cadena de texto
         }
         salida.flush();
      } catch (FileNotFoundException e) {
         System.out.println(e.getMessage());
      } finally {
```

```
            salida.close();
        }
    }
}
```

El método flush() provoca que se escriban en el fichero los datos que puedan haber en el buffer de salida.
El método close() cierra la conexión con el fichero y libera los recursos que está usando la conexión.

A partir de Java 7 se puede usar la instrucción try-with-resources. Un resource (recurso) es un objeto que necesita ser cerrado después de usarlo. Una instrucción try-with-resources asegura que estos objetos serán cerrados al final de la instrucción try. Cualquier objeto que implemente la interface java.lang.AutoCloseable, entre ellos los que implementan la interface java.io.Closeable pueden ser usados como resources.
El ejemplo anterior se escribiría así:

```
import java.io.FileNotFoundException;
import java.io.PrintWriter;
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String cadena;
        try (PrintWriter salida = new PrintWriter("c:/ficheros/datos.txt")) {
            System.out.println("Introduce texto. Para acabar introduce la cadena FIN:");
            cadena = sc.nextLine();
            while (!cadena.equalsIgnoreCase("FIN")) {
                salida.println(cadena);
                cadena = sc.nextLine();
            }
        } catch (FileNotFoundException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

A continuación del try se escribe el recurso entre paréntesis. Ahora no es necesario escribir el bloque finally para cerrar el fichero.
La instrucción try puede contener varios recursos, en este caso irán separados por punto y coma.

Ejemplo: Programa que leer por teclado líneas de texto y las añade al final del ficheros

datos.txt. Para resolverlo vamos a modificar el programa anterior para que añada texto al fichero datos.txt, es decir, al volver a ejecutar el programa el contenido anterior del fichero no se pierde y el contenido nuevo se añade al final.

```
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Scanner;
public class Main {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      String cadena;
      try (FileWriter fw = new FileWriter("c:/ficheros/datos.txt", true);
            PrintWriter salida = new PrintWriter(fw)) {
         System.out.println("Introduce texto. Para acabar introduce la cadena FIN:");
         cadena = sc.nextLine();
         while (!cadena.equalsIgnoreCase("FIN")) {
            salida.println(cadena);
            cadena = sc.nextLine();
         }
      } catch (IOException ex) {
         System.out.println(ex.getMessage());
      }
   }
}
```

LECTURA DE FICHEROS DE TEXTO EN JAVA

Para leer en un fichero de texto utilizaremos dos clases:
FileReader y BufferedReader.
La clase FileReader permite tener acceso al fichero en modo lectura.
Para crear objetos FileReader podemos utilizar los constructores:

```
FileReader(String ruta)
FileReader(File objetoFile);
```

Ambos lanzan una excepción FileNotFoundException si el fichero no existe.
La clase FileReader proporciona el método read() para leer caracteres del fichero aunque lo normal es realizar la lectura mediante la clase BufferedReader.

Para leer utilizando la clase BufferedReader se debe crear un objeto BufferedReader a partir de un objeto FileReader:
Ejemplo:

```
FileReader fr = new FileReader("c:/ficheros/datos.txt");
BufferedReader entrada = new BufferedReader (fr);
```

Una vez creado el objeto BufferedReader podemos utilizar:

- El método readLine() para leer líneas de texto del fichero (String). Este método devuelve null cuando no hay más líneas para leer.
- El método read() para leer carácter a carácter. Devuelve un entero que representa el código Unicode del carácter leído. Devuelve -1 si no hay más caracteres.

Ambos métodos lanzan una excepción IOException si ocurre un error de lectura.
El fichero se debe cerrar cuando ya no se use, mediante el método close(). Este método lanza una excepción IOException.
Ejemplo de lectura de un fichero de texto:
Programa Java que lee el contenido del fichero datos.txt creado en el ejemplo anterior y lo muestra por pantalla. El proceso consiste en leer una línea del fichero y mostrarla por pantalla. El proceso se repite hasta que se llegue al final del fichero y no hayan más líneas que leer. Cuando esto ocurre el método readLine() devuelve null.

```
import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;

public class File13 {
   public static void main(String[] args) {
      FileReader fr = null;
      try {
         fr = new FileReader("c:/ficheros/datos.txt");
         BufferedReader entrada = new BufferedReader(fr);
         String cadena = entrada.readLine(); //se lee la primera línea del fichero
         while (cadena != null) {  //mientras no se llegue al final del fichero
            System.out.println(cadena); //se nuestra por pantalla
            cadena = entrada.readLine(); //se lee la siguiente línea del fichero
         }
      } catch (FileNotFoundException e) {
         System.out.println(e.getMessage());
      } catch (IOException e) {
         System.out.println(e.getMessage());
      } finally {
```

```
            try {
               if (fr != null) {
                  fr.close();
               }
            } catch (IOException e) {
               System.out.println(e.getMessage());
            }
         }
      }
   }
```

Ejemplo:

Mostrar por pantalla el contenido del fichero de texto datos.txt pero en este caso lo vamos a leer carácter a carcater. El proceso consiste en leer un carácter del fichero y mostrarlo por pantalla. Este proceso se repite hasta que no queden más caracteres que leer en el fichero, es decir, hasta que se alcance el finall del fichero. En este caso el método read() devuelve -1.

```
import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;

public class File14 {
   public static void main(String[] args) {
      FileReader fr = null;
      try {
         fr = new FileReader("c:/ficheros/datos.txt");
         BufferedReader entrada = new BufferedReader(fr);
         int car = entrada.read(); //se lee el primer carácter del fichero
         while (car != -1) { //mientras no se llegue al final del fichero
            System.out.print((char) car); //se nuestra por pantalla
            car = entrada.read(); //se lee el siguiente carácter del fichero
         }
      } catch (FileNotFoundException e) {
         System.out.println(e.getMessage());
      } catch (IOException e) {
         System.out.println(e.getMessage());
      } finally {
         try {
            if (fr != null) {
               fr.close();
            }
         } catch (IOException e) {
```

```
            System.out.println(e.getMessage());
        }
    }
  }
}
```

LECTURA DE FICHEROS DE TEXTO CON SCANNER

A partir de Java 5 se puede leer un fichero de texto utilizando la clase Scanner igual que si leyéramos por teclado.

Para ello se le pasa al constructor de Scanner el objeto File asociado al fichero.

Esta operación lanza una excepción FileNotFoundException.

Ejemplo de lectura de un fichero de texto con Scanner:

Programa que lee línea a línea el contenido del fichero datos.txt utilizando la clase Scanner. Se utiliza el método hasNext() de Scanner para saber si quedan más datos que leer en el fichero. Este método devuelve false si se ha llegado al final del fichero y true en caso contrario.

```
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;
public class File12 {
   public static void main(String[] args) {
      File f = new File("c:/ficheros/datos.txt");
      String cadena;
      Scanner entrada = null;
      try {
         entrada = new Scanner(f); //se crea un Scanner asociado al fichero
         while (entrada.hasNext()) { //mientras no se alcance el final del fichero
            cadena = entrada.nextLine(); //se lee una línea del fichero
            System.out.println(cadena); //se muestra por pantalla
         }
      } catch (FileNotFoundException e) {
         System.out.println(e.getMessage());
      } finally{
         entrada.close();
      }
   }
}
```

Ejemplo de lectura de ficheros de texto con Scanner:

Disponemos de un fichero de texto llamado enteros.txt que contiene números enteros. El siguiente programa lee los números y los muestra. Muestra también la cantidad de números leídos y su suma.

Por ejemplo, si el fichero enteros.txt contiene los siguientes números:
323 34 234 990 22 3 1
5463 28 34 0 7

El programa mostrará por pantalla:

323
34
234
990
22
3
1
5463
28
34
0
7
Número leídos: 12
Suma 7139
Se utilizará el método hasNextInt() de Scanner para saber si quedan más enteros que leer en el fichero. El método hasNextInt() devuelve true cuando lo siguiente que se va a extraer del fichero es un entero y devuelve false en caso contrario.
La lectura acaba cuando no quedan más enteros (se ha llegado al final del fichero) o cuando encuentra un carácter no válido como entero.

Por ejemplo, si el contenido del fichero enteros.txt es el siguiente:

323 34 KKK 234 990 22 3 1
5463 28 34 0 7

El programa mostrará por pantalla:
323
34
Número leídos: 2
Suma 357

```
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;
public class File15 {
   public static void main(String[] args) {
```

```
        File f = new File("c:/ficheros/enteros.txt");
        int numero, suma = 0, cont = 0;
        Scanner entrada = null;
        try {
           entrada = new Scanner(f);
           while (entrada.hasNextInt()) {
                numero = entrada.nextInt();
                System.out.println(numero);
                suma = suma + numero;
                cont++;
           }
           System.out.println("Número leídos: " + cont);
           System.out.println("Suma " + suma);
        } catch (FileNotFoundException e) {
           System.out.println(e.getMessage());
        } finally{
           entrada.close();
        }
     }
}
```

Ejemplo de lectura de un fichero de texto con Scanner:

Disponemos de un fichero de texto llamado enteros.txt que contiene los siguientes números enteros separados por espacios en blanco o comas:

```
34,45,23 8, 9
12 23
```

El siguiente programa Java lee el contenido del fichero y muestra los números. Muestra también la cantidad de números leídos y su suma.

El programa lee líneas completas del fichero y las pasa a un StringTokenizer del que se extraen los números.

```
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;
import java.util.StringTokenizer;

public class File16 {
   public static void main(String[] args) {
        File f = new File("c:/ficheros/enteros.txt");
        int numero, suma = 0, cont = 0;
        StringTokenizer st;
        Scanner entrada = null;
        String cadena;
```

```
        try {
            entrada = new Scanner(f);
            while (entrada.hasNext()) {
                cadena = entrada.nextLine();
                st = new StringTokenizer(cadena, " ,");
                while (st.hasMoreTokens()) {
                    numero = Integer.parseInt(st.nextToken());
                    System.out.println(numero);
                    suma = suma + numero;
                    cont++;
                }
            }
            System.out.println("Número leídos: " + cont);
            System.out.println("Suma " + suma);
        } catch (FileNotFoundException e) {
            System.out.println(e.getMessage());
        } finally {
            entrada.close();
        }
    }
}
```

LECTURA DE FICHEROS DE TEXTO EN JAVA

Para leer en un fichero de texto utilizaremos dos clases:
FileReader y BufferedReader.
La clase FileReader permite tener acceso al fichero en modo lectura.
Para crear objetos FileReader podemos utilizar los constructores:

```
FileReader(String ruta)
FileReader(File objetoFile);
```

Ambos lanzan una excepción FileNotFoundException si el fichero no existe.
La clase FileReader proporciona el método read() para leer caracteres del fichero aunque lo normal es realizar la lectura mediante la clase BufferedReader.
Para leer utilizando la clase BufferedReader se debe crear un objeto BufferedReader a partir de un objeto FileReader:
Ejemplo:

```
FileReader fr = new FileReader("c:/ficheros/datos.txt");
```

BufferedReader entrada = new BufferedReader (fr);

Una vez creado el objeto BufferedReader podemos utilizar:

- El método readLine() para leer líneas de texto del fichero (String). Este método devuelve null cuando no hay más líneas para leer.
- El método read() para leer carácter a carácter. Devuelve un entero que representa el código Unicode del carácter leído. Devuelve -1 si no hay más caracteres.

Ambos métodos lanzan una excepción IOException si ocurre un error de lectura.
El fichero se debe cerrar cuando ya no se use, mediante el método close(). Este método lanza una excepción IOException.
Ejemplo de lectura de un fichero de texto:
Programa Java que lee el contenido del fichero datos.txt creado en el ejemplo anterior y lo muestra por pantalla. El proceso consiste en leer una línea del fichero y mostrarla por pantalla. El proceso se repite hasta que se llegue al final del fichero y no hayan más líneas que leer. Cuando esto ocurre el método readLine() devuelve null.

```
import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;

public class File13 {
   public static void main(String[] args) {
      FileReader fr = null;
      try {
         fr = new FileReader("c:/ficheros/datos.txt");
         BufferedReader entrada = new BufferedReader(fr);
         String cadena = entrada.readLine(); //se lee la primera línea del fichero
         while (cadena != null) {  //mientras no se llegue al final del fichero
            System.out.println(cadena); //se nuestra por pantalla
            cadena = entrada.readLine(); //se lee la siguiente línea del fichero
         }
      } catch (FileNotFoundException e) {
         System.out.println(e.getMessage());
      } catch (IOException e) {
         System.out.println(e.getMessage());
      } finally {
         try {
            if (fr != null) {
               fr.close();
            }
         } catch (IOException e) {
```

```
            System.out.println(e.getMessage());
        }
    }
  }
}
```

Ejemplo:

Mostrar por pantalla el contenido del fichero de texto datos.txt pero en este caso lo vamos a leer carácter a carcater. El proceso consiste en leer un carácter del fichero y mostrarlo por pantalla. Este proceso se repite hasta que no queden más caracteres que leer en el fichero, es decir, hasta que se alcance el finall del fichero. En este caso el método read() devuelve -1.

```
import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;

public class File14 {
   public static void main(String[] args) {
      FileReader fr = null;
      try {
         fr = new FileReader("c:/ficheros/datos.txt");
         BufferedReader entrada = new BufferedReader(fr);
         int car = entrada.read(); //se lee el primer carácter del fichero
         while (car != -1) { //mientras no se llegue al final del fichero
            System.out.print((char) car); //se nuestra por pantalla
            car = entrada.read(); //se lee el siguiente carácter del fichero
         }
      } catch (FileNotFoundException e) {
         System.out.println(e.getMessage());
      } catch (IOException e) {
         System.out.println(e.getMessage());
      } finally {
         try {
            if (fr != null) {
               fr.close();
            }
         } catch (IOException e) {
            System.out.println(e.getMessage());
         }
      }
   }
}
```

LECTURA DE FICHEROS DE TEXTO CON SCANNER

A partir de Java 5 se puede leer un fichero de texto utilizando la clase Scanner igual que si leyéramos por teclado.

Para ello se le pasa al constructor de Scanner el objeto File asociado al fichero.

Esta operación lanza una excepción FileNotFoundException.

Ejemplo de lectura de un fichero de texto con Scanner:

Programa que lee línea a línea el contenido del fichero datos.txt utilizando la clase Scanner. Se utiliza el método hasNext() de Scanner para saber si quedan más datos que leer en el fichero. Este método devuelve false si se ha llegado al final del fichero y true en caso contrario.

```
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;
public class File12 {
   public static void main(String[] args) {
      File f = new File("c:/ficheros/datos.txt");
      String cadena;
      Scanner entrada = null;
      try {
         entrada = new Scanner(f); //se crea un Scanner asociado al fichero
         while (entrada.hasNext()) { //mientras no se alcance el final del fichero
            cadena = entrada.nextLine(); //se lee una línea del fichero
            System.out.println(cadena); //se muestra por pantalla
         }
      } catch (FileNotFoundException e) {
         System.out.println(e.getMessage());
      } finally{
         entrada.close();
      }
   }
}
```

Ejemplo de lectura de ficheros de texto con Scanner:

Disponemos de un fichero de texto llamado enteros.txt que contiene números enteros. El siguiente programa lee los números y los muestra. Muestra también la cantidad de números leídos y su suma.

Por ejemplo, si el fichero enteros.txt contiene los siguientes números:

323 34 234 990 22 3 1

5463 28 34 0 7

El programa mostrará por pantalla:

323
34
234
990
22
3
1
5463
28
34
0
7
Número leídos: 12
Suma 7139
Se utilizará el método hasNextInt() de Scanner para saber si quedan más enteros que leer en el fichero. El método hasNextInt() devuelve true cuando lo siguiente que se va a extraer del fichero es un entero y devuelve false en caso contrario.
La lectura acaba cuando no quedan más enteros (se ha llegado al final del fichero) o cuando encuentra un carácter no válido como entero.

Por ejemplo, si el contenido del fichero enteros.txt es el siguiente:

323 34 KKK 234 990 22 3 1
5463 28 34 0 7

El programa mostrará por pantalla:
323
34
Número leídos: 2
Suma 357

```
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;
public class File15 {
   public static void main(String[] args) {
      File f = new File("c:/ficheros/enteros.txt");
      int numero, suma = 0, cont = 0;
      Scanner entrada = null;
      try {
         entrada = new Scanner(f);
```

```
            while (entrada.hasNextInt()) {
                numero = entrada.nextInt();
                System.out.println(numero);
                suma = suma + numero;
                cont++;
            }
            System.out.println("Número leídos: " + cont);
            System.out.println("Suma " + suma);
        } catch (FileNotFoundException e) {
            System.out.println(e.getMessage());
        } finally{
            entrada.close();
        }
    }
}
```

Ejemplo de lectura de un fichero de texto con Scanner:

Disponemos de un fichero de texto llamado enteros.txt que contiene los siguientes números enteros separados por espacios en blanco o comas:

34,45,23 8, 9
12 23

El siguiente programa Java lee el contenido del fichero y muestra los números. Muestra también la cantidad de números leídos y su suma.

El programa lee líneas completas del fichero y las pasa a un StringTokenizer del que se extraen los números.

```
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;
import java.util.StringTokenizer;

public class File16 {
    public static void main(String[] args) {
        File f = new File("c:/ficheros/enteros.txt");
        int numero, suma = 0, cont = 0;
        StringTokenizer st;
        Scanner entrada = null;
        String cadena;
        try {
            entrada = new Scanner(f);
            while (entrada.hasNext()) {
                cadena = entrada.nextLine();
                st = new StringTokenizer(cadena, " ,");
```

```
            while (st.hasMoreTokens()) {
                numero = Integer.parseInt(st.nextToken());
                System.out.println(numero);
                suma = suma + numero;
                cont++;
            }
        }
        System.out.println("Número leídos: " + cont);
        System.out.println("Suma " + suma);
    } catch (FileNotFoundException e) {
        System.out.println(e.getMessage());
    } finally {
        entrada.close();
    }
  }
}
```

Ficheros de texto en Java

Un fichero de texto está formado por secuencias de caracteres, organizados en líneas de igual o distinta longitud.
Todos los datos que aparecen en estos ficheros están formados por caracteres.
CREAR Y ESCRIBIR EN FICHEROS DE TEXTO EN JAVA
Para escribir en un fichero de texto utilizaremos dos clases:
FileWriter y PrintWriter.
La clase FileWriter permite tener acceso al fichero en modo escritura.
Para crear objetos FileWriter podemos utilizar los constructores:

FileWriter(String path)
FileWriter(File objetoFile);

El fichero se crea y si ya existe su contenido se pierde.
Si lo que necesitamos es abrir un fichero de texto existente sin perder su contenido y añadir más contenido al final utilizaremos los constructores:

FileWriter(String path, boolean append)
FileWriter(File objetoFile, boolean append)

Si el parámetro append es true significa que los datos se van a añadir a los existentes. Si es false los datos existentes se pierden.
La clase FileWriter proporciona el método write() para escribir cadenas de caracteres aunque lo normal es utilizar esta clase junto con la clase PrintWriter para facilitar la escritura.
La clase PrintWriter permite escribir caracteres en el fichero de la misma forma que en la pantalla.
Un objeto PrintWriter se crea a partir de un objeto FileWriter.
Ejemplo:

```
FileWriter fw = new FileWriter("c:/ficheros/datos.txt");
PrintWriter salida = new PrintWriter(fw);
```

En este ejemplo se ha creado un fichero de texto llamado datos.txt. El fichero se encuentra dentro de la carpeta ficheros en la unidad C:

A partir de Java 5 se puede crear un objeto PrintWriter directamente a partir de un objeto File o de la ruta:

```
PrintWriter salida = new PrintWriter("c:/ficheros/datos.txt");
```

En este caso, si el fichero no existe se crea. Si no se puede crear un archivo con ese nombre o si ocurre algún error se lanza una excepción FileNotFoundException.
Una vez creado el objeto podemos utilizar print(), println() y printf() para escribir en el fichero como si fuese en pantalla.

Ejemplo de escritura de un fichero de texto:
Programa Java que lee texto por teclado y lo escribe en un fichero de texto llamado datos.txt. El proceso consiste en leer una línea de texto por teclado y escribirla en el fichero. Este proceso se repite hasta que se introduce por teclado la cadena FIN. La cadena FIN que indica el final de lectura no se debe escribir en el fichero.

```
import java.io.FileNotFoundException;
import java.io.PrintWriter;
import java.util.Scanner;
public class File11 {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      PrintWriter salida = null;
      try {
         salida = new PrintWriter("c:/ficheros/datos.txt"); //se crea el fichero
         String cadena;
         System.out.println("Introduce texto. Para acabar introduce la cadena FIN:");
         cadena = sc.nextLine(); //se introduce por teclado una cadena de texto
         while (!cadena.equalsIgnoreCase("FIN")) {
            salida.println(cadena);  //se escribe la cadena en el fichero
```

```
            cadena = sc.nextLine(); //se introduce por teclado una cadena de texto
        }
        salida.flush();
      } catch (FileNotFoundException e) {
        System.out.println(e.getMessage());
      } finally {
        salida.close();
      }
   }
}
```

El método flush() provoca que se escriban en el fichero los datos que puedan haber en el buffer de salida.
El método close() cierra la conexión con el fichero y libera los recursos que está usando la conexión.

A partir de Java 7 se puede usar la instrucción try-with-resources. Un resource (recurso) es un objeto que necesita ser cerrado después de usarlo. Una instrucción try-with-resources asegura que estos objetos serán cerrados al final de la instrucción try. Cualquier objeto que implemente la interface java.lang.AutoCloseable, entre ellos los que implementan la interface java.io.Closeable pueden ser usados como resources.
El ejemplo anterior se escribiría así:

```
import java.io.FileNotFoundException;
import java.io.PrintWriter;
import java.util.Scanner;
public class Main {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      String cadena;
      try (PrintWriter salida = new PrintWriter("c:/ficheros/datos.txt")) {
        System.out.println("Introduce texto. Para acabar introduce la cadena FIN:");
        cadena = sc.nextLine();
        while (!cadena.equalsIgnoreCase("FIN")) {
          salida.println(cadena);
          cadena = sc.nextLine();
        }
      } catch (FileNotFoundException e) {
        System.out.println(e.getMessage());
      }
   }
}
```

A continuación del try se escribe el recurso entre paréntesis. Ahora no es necesario escribir el bloque finally para cerrar el fichero.
La instrucción try puede contener varios recursos, en este caso irán separados por punto y coma.

Ejemplo: Programa que leer por teclado líneas de texto y las añade al final del ficheros datos.txt. Para resolverlo vamos a modificar el programa anterior para que añada texto al fichero datos.txt, es decir, al volver a ejecutar el programa el contenido anterior del fichero no se pierde y el contenido nuevo se añade al final.

```
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Scanner;
public class Main {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      String cadena;
      try (FileWriter fw = new FileWriter("c:/ficheros/datos.txt", true);
            PrintWriter salida = new PrintWriter(fw)) {
         System.out.println("Introduce texto. Para acabar introduce la cadena FIN:");
         cadena = sc.nextLine();
         while (!cadena.equalsIgnoreCase("FIN")) {
            salida.println(cadena);
            cadena = sc.nextLine();
         }
      } catch (IOException ex) {
         System.out.println(ex.getMessage());
      }
   }
}
```

LECTURA DE FICHEROS DE TEXTO EN JAVA
Para leer en un fichero de texto utilizaremos dos clases:
FileReader y BufferedReader.
La clase FileReader permite tener acceso al fichero en modo lectura.
Para crear objetos FileReader podemos utilizar los constructores:

FileReader(String ruta)
FileReader(File objetoFile);

Ambos lanzan una excepción FileNotFoundException si el fichero no existe.
La clase FileReader proporciona el método read() para leer caracteres del fichero aunque lo normal es realizar la lectura mediante la clase BufferedReader.
Para leer utilizando la clase BufferedReader se debe crear un objeto BufferedReader a partir de un objeto FileReader:
Ejemplo:

```
FileReader fr = new FileReader("c:/ficheros/datos.txt");
BufferedReader entrada = new BufferedReader (fr);
```

Una vez creado el objeto BufferedReader podemos utilizar:
- El método readLine() para leer líneas de texto del fichero (String). Este método devuelve null cuando no hay más líneas para leer.
- El método read() para leer carácter a carácter. Devuelve un entero que representa el código Unicode del carácter leído. Devuelve -1 si no hay más caracteres.

Ambos métodos lanzan una excepción IOException si ocurre un error de lectura.
El fichero se debe cerrar cuando ya no se use, mediante el método close(). Este método lanza una excepción IOException.
Ejemplo de lectura de un fichero de texto:
Programa Java que lee el contenido del fichero datos.txt creado en el ejemplo anterior y lo muestra por pantalla. El proceso consiste en leer una línea del fichero y mostrarla por pantalla. El proceso se repite hasta que se llegue al final del fichero y no hayan más líneas que leer. Cuando esto ocurre el método readLine() devuelve null.

```
import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;

public class File13 {
   public static void main(String[] args) {
      FileReader fr = null;
      try {
         fr = new FileReader("c:/ficheros/datos.txt");
         BufferedReader entrada = new BufferedReader(fr);
         String cadena = entrada.readLine(); //se lee la primera línea del fichero
         while (cadena != null) {  //mientras no se llegue al final del fichero
            System.out.println(cadena); //se nuestra por pantalla
            cadena = entrada.readLine(); //se lee la siguiente línea del fichero
```

```
            }
        } catch (FileNotFoundException e) {
            System.out.println(e.getMessage());
        } catch (IOException e) {
            System.out.println(e.getMessage());
        } finally {
            try {
                if (fr != null) {
                    fr.close();
                }
            } catch (IOException e) {
                System.out.println(e.getMessage());
            }
        }
    }
}
```

Ejemplo de lectura de un fichero de texto con Scanner:

Disponemos de un fichero de texto llamado enteros.txt que contiene los siguientes números enteros separados por espacios en blanco o comas:

34,45,23 8, 9
12 23

El siguiente programa Java lee el contenido del fichero y muestra los números. Muestra también la cantidad de números leídos y su suma.

El programa lee líneas completas del fichero y las pasa a un StringTokenizer del que se extraen los números.

```
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;
import java.util.StringTokenizer;

public class File16 {
    public static void main(String[] args) {
        File f = new File("c:/ficheros/enteros.txt");
        int numero, suma = 0, cont = 0;
        StringTokenizer st;
        Scanner entrada = null;
        String cadena;
        try {
            entrada = new Scanner(f);
            while (entrada.hasNext()) {
                cadena = entrada.nextLine();
```

```
            st = new StringTokenizer(cadena, " ,");
            while (st.hasMoreTokens()) {
               numero = Integer.parseInt(st.nextToken());
               System.out.println(numero);
               suma = suma + numero;
               cont++;
            }
         }
         System.out.println("Número leídos: " + cont);
         System.out.println("Suma " + suma);
      } catch (FileNotFoundException e) {
         System.out.println(e.getMessage());
      } finally {
         entrada.close();
      }
   }
}
```

Ficheros de texto en Java

Un fichero de texto está formado por secuencias de caracteres, organizados en líneas de igual o distinta longitud.
Todos los datos que aparecen en estos ficheros están formados por caracteres.
CREAR Y ESCRIBIR EN FICHEROS DE TEXTO EN JAVA
Para escribir en un fichero de texto utilizaremos dos clases:
FileWriter y PrintWriter.
La clase FileWriter permite tener acceso al fichero en modo escritura.
Para crear objetos FileWriter podemos utilizar los constructores:

FileWriter(String path)
FileWriter(File objetoFile);

El fichero se crea y si ya existe su contenido se pierde.
Si lo que necesitamos es abrir un fichero de texto existente sin perder su contenido y añadir

más contenido al final utilizaremos los constructores:

FileWriter(String path, boolean append)
FileWriter(File objetoFile, boolean append)

Si el parámetro append es true significa que los datos se van a añadir a los existentes. Si es false los datos existentes se pierden.
La clase FileWriter proporciona el método write() para escribir cadenas de caracteres aunque lo normal es utilizar esta clase junto con la clase PrintWriter para facilitar la escritura.
La clase PrintWriter permite escribir caracteres en el fichero de la misma forma que en la pantalla.
Un objeto PrintWriter se crea a partir de un objeto FileWriter.
Ejemplo:
FileWriter fw = new FileWriter("c:/ficheros/datos.txt");
PrintWriter salida = new PrintWriter(fw);
En este ejemplo se ha creado un fichero de texto llamado datos.txt. El fichero se encuentra dentro de la carpeta ficheros en la unidad C:

A partir de Java 5 se puede crear un objeto PrintWriter directamente a partir de un objeto File o de la ruta:
PrintWriter salida = new PrintWriter("c:/ficheros/datos.txt");
En este caso, si el fichero no existe se crea. Si no se puede crear un archivo con ese nombre o si ocurre algún error se lanza una excepción FileNotFoundException.
Una vez creado el objeto podemos utilizar print(), println() y printf() para escribir en el fichero como si fuese en pantalla.

Ejemplo de escritura de un fichero de texto:
Programa Java que lee texto por teclado y lo escribe en un fichero de texto llamado datos.txt. El proceso consiste en leer una línea de texto por teclado y escribirla en el fichero. Este proceso se repite hasta que se introduce por teclado la cadena FIN. La cadena FIN que indica el final de lectura no se debe escribir en el fichero.

```
import java.io.FileNotFoundException;
import java.io.PrintWriter;
import java.util.Scanner;
public class File11 {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      PrintWriter salida = null;
      try {
         salida = new PrintWriter("c:/ficheros/datos.txt"); //se crea el fichero
         String cadena;
```

```
            System.out.println("Introduce texto. Para acabar introduce la cadena FIN:");
            cadena = sc.nextLine(); //se introduce por teclado una cadena de texto
            while (!cadena.equalsIgnoreCase("FIN")) {
                salida.println(cadena);  //se escribe la cadena en el fichero
                cadena = sc.nextLine(); //se introduce por teclado una cadena de texto
            }
            salida.flush();
        } catch (FileNotFoundException e) {
            System.out.println(e.getMessage());
        } finally {
            salida.close();
        }
    }
}
```

El método flush() provoca que se escriban en el fichero los datos que puedan haber en el buffer de salida.
El método close() cierra la conexión con el fichero y libera los recursos que está usando la conexión.

A partir de Java 7 se puede usar la instrucción try-with-resources. Un resource (recurso) es un objeto que necesita ser cerrado después de usarlo. Una instrucción try-with-resources asegura que estos objetos serán cerrados al final de la instrucción try. Cualquier objeto que implemente la interface java.lang.AutoCloseable, entre ellos los que implementan la interface java.io.Closeable pueden ser usados como resources.
El ejemplo anterior se escribiría así:

```
import java.io.FileNotFoundException;
import java.io.PrintWriter;
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String cadena;
        try (PrintWriter salida = new PrintWriter("c:/ficheros/datos.txt")) {
            System.out.println("Introduce texto. Para acabar introduce la cadena FIN:");
            cadena = sc.nextLine();
            while (!cadena.equalsIgnoreCase("FIN")) {
                salida.println(cadena);
                cadena = sc.nextLine();
            }
        } catch (FileNotFoundException e) {
```

```
            System.out.println(e.getMessage());
        }
    }
}
```

A continuación del try se escribe el recurso entre paréntesis. Ahora no es necesario escribir el bloque finally para cerrar el fichero.
La instrucción try puede contener varios recursos, en este caso irán separados por punto y coma.

Ejemplo: Programa que leer por teclado líneas de texto y las añade al final del ficheros datos.txt. Para resolverlo vamos a modificar el programa anterior para que añada texto al fichero datos.txt, es decir, al volver a ejecutar el programa el contenido anterior del fichero no se pierde y el contenido nuevo se añade al final.

```
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String cadena;
        try (FileWriter fw = new FileWriter("c:/ficheros/datos.txt", true);
                PrintWriter salida = new PrintWriter(fw)) {
            System.out.println("Introduce texto. Para acabar introduce la cadena FIN:");
            cadena = sc.nextLine();
            while (!cadena.equalsIgnoreCase("FIN")) {
                salida.println(cadena);
                cadena = sc.nextLine();
            }
        } catch (IOException ex) {
            System.out.println(ex.getMessage());
        }
    }
}
```

LECTURA DE FICHEROS DE TEXTO EN JAVA

Para leer en un fichero de texto utilizaremos dos clases:

FileReader y BufferedReader.
La clase FileReader permite tener acceso al fichero en modo lectura.
Para crear objetos FileReader podemos utilizar los constructores:

FileReader(String ruta)
FileReader(File objetoFile);

Ambos lanzan una excepción FileNotFoundException si el fichero no existe.
La clase FileReader proporciona el método read() para leer caracteres del fichero aunque lo normal es realizar la lectura mediante la clase BufferedReader.
Para leer utilizando la clase BufferedReader se debe crear un objeto BufferedReader a partir de un objeto FileReader:
Ejemplo:

FileReader fr = new FileReader("c:/ficheros/datos.txt");
BufferedReader entrada = new BufferedReader (fr);

Una vez creado el objeto BufferedReader podemos utilizar:
- El método readLine() para leer líneas de texto del fichero (String). Este método devuelve null cuando no hay más líneas para leer.
- El método read() para leer carácter a carácter. Devuelve un entero que representa el código Unicode del carácter leído. Devuelve -1 si no hay más caracteres.

Ambos métodos lanzan una excepción IOException si ocurre un error de lectura.
El fichero se debe cerrar cuando ya no se use, mediante el método close(). Este método lanza una excepción IOException.
Ejemplo de lectura de un fichero de texto:
Programa Java que lee el contenido del fichero datos.txt creado en el ejemplo anterior y lo muestra por pantalla. El proceso consiste en leer una línea del fichero y mostrarla por pantalla. El proceso se repite hasta que se llegue al final del fichero y no hayan más líneas que leer. Cuando esto ocurre el método readLine() devuelve null.

```
import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;

public class File13 {
   public static void main(String[] args) {
      FileReader fr = null;
      try {
         fr = new FileReader("c:/ficheros/datos.txt");
         BufferedReader entrada = new BufferedReader(fr);
```

```
            String cadena = entrada.readLine(); //se lee la primera línea del fichero
            while (cadena != null) {  //mientras no se llegue al final del fichero
                System.out.println(cadena); //se nuestra por pantalla
                cadena = entrada.readLine(); //se lee la siguiente línea del fichero
            }
        } catch (FileNotFoundException e) {
            System.out.println(e.getMessage());
        } catch (IOException e) {
            System.out.println(e.getMessage());
        } finally {
            try {
                if (fr != null) {
                    fr.close();
                }
            } catch (IOException e) {
                System.out.println(e.getMessage());
            }
        }
    }
}
```

Ejemplo:

Mostrar por pantalla el contenido del fichero de texto datos.txt pero en este caso lo vamos a leer carácter a carcater. El proceso consiste en leer un carácter del fichero y mostrarlo por pantalla. Este proceso se repite hasta que no queden más caracteres que leer en el fichero, es decir, hasta que se alcance el finall del fichero. En este caso el método read() devuelve -1.

```
import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;

public class File14 {
    public static void main(String[] args) {
        FileReader fr = null;
        try {
            fr = new FileReader("c:/ficheros/datos.txt");
            BufferedReader entrada = new BufferedReader(fr);
            int car = entrada.read(); //se lee el primer carácter del fichero
            while (car != -1) { //mientras no se llegue al final del fichero
                System.out.print((char) car); //se nuestra por pantalla
                car = entrada.read(); //se lee el siguiente carácter del fichero
            }
```

```
        } catch (FileNotFoundException e) {
            System.out.println(e.getMessage());
        } catch (IOException e) {
            System.out.println(e.getMessage());
        } finally {
            try {
                if (fr != null) {
                    fr.close();
                }
            } catch (IOException e) {
                System.out.println(e.getMessage());
            }
        }
    }
}
```

LECTURA DE FICHEROS DE TEXTO CON SCANNER

A partir de Java 5 se puede leer un fichero de texto utilizando la clase Scanner igual que si leyéramos por teclado.

Para ello se le pasa al constructor de Scanner el objeto File asociado al fichero.

Esta operación lanza una excepción FileNotFoundException.

Ejemplo de lectura de un fichero de texto con Scanner:

Programa que lee línea a línea el contenido del fichero datos.txt utilizando la clase Scanner. Se utiliza el método hasNext() de Scanner para saber si quedan más datos que leer en el fichero. Este método devuelve false si se ha llegado al final del fichero y true en caso contrario.

```
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;
public class File12 {
    public static void main(String[] args) {
        File f = new File("c:/ficheros/datos.txt");
        String cadena;
        Scanner entrada = null;
        try {
            entrada = new Scanner(f); //se crea un Scanner asociado al fichero
            while (entrada.hasNext()) { //mientras no se alcance el final del fichero
                cadena = entrada.nextLine(); //se lee una línea del fichero
                System.out.println(cadena); //se muestra por pantalla
            }
        } catch (FileNotFoundException e) {
            System.out.println(e.getMessage());
```

```
        } finally{
            entrada.close();
        }
    }
}
```

Ejemplo de lectura de ficheros de texto con Scanner:
Disponemos de un fichero de texto llamado enteros.txt que contiene números enteros. El siguiente programa lee los números y los muestra. Muestra también la cantidad de números leídos y su suma.

Por ejemplo, si el fichero enteros.txt contiene los siguientes números:
323 34 234 990 22 3 1
5463 28 34 0 7

El programa mostrará por pantalla:

323
34
234
990
22
3
1
5463
28
34
0
7
Número leídos: 12
Suma 7139
Se utilizará el método hasNextInt() de Scanner para saber si quedan más enteros que leer en el fichero. El método hasNextInt() devuelve true cuando lo siguiente que se va a extraer del fichero es un entero y devuelve false en caso contrario.
La lectura acaba cuando no quedan más enteros (se ha llegado al final del fichero) o cuando encuentra un carácter no válido como entero.

Por ejemplo, si el contenido del fichero enteros.txt es el siguiente:

323 34 KKK 234 990 22 3 1
5463 28 34 0 7

El programa mostrará por pantalla:

```
323
34
Número leídos: 2
Suma 357
```

```
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;
public class File15 {
   public static void main(String[] args) {
      File f = new File("c:/ficheros/enteros.txt");
      int numero, suma = 0, cont = 0;
      Scanner entrada = null;
      try {
         entrada = new Scanner(f);
         while (entrada.hasNextInt()) {
               numero = entrada.nextInt();
               System.out.println(numero);
               suma = suma + numero;
               cont++;
         }
         System.out.println("Número leídos: " + cont);
         System.out.println("Suma " + suma);
      } catch (FileNotFoundException e) {
         System.out.println(e.getMessage());
      } finally{
         entrada.close();
      }
   }
}
```

Obtener la linea de mayor longitud y la de menor longitud dentro de un fichero de texto

Programa que obtiene la línea de mayor tamaño y la de menor tamaño dentro de un fichero de texto.
Para resolver este ejercicio se utiliza la clase Scanner para leer el fichero. La lectura se realiza línea a línea hasta que se alcance el final del fichero. Para determinar cuál es la de mayor longitud y cuál es la menor, se lee la primera línea del fichero y se toma como la línea mayor y la menor. A continuación se leen el resto de líneas y para cada una se compara su tamaño con la mayor y menor actuales.
El nombre del fichero se selecciona de forma gráfica utilizando la clase JFileChooser.

Solución:

```
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;
import javax.swing.JFileChooser;

public class FicheroDeTexto {

   public static void main(String[] args) {

      Scanner entrada = null;
      String cadena, mayor = null, menor = null;
      JFileChooser j = new JFileChooser();
      j.showOpenDialog(j); //muestra la ventana para seleccionar el archivo
      try {
         String ruta = j.getSelectedFile().getAbsolutePath(); //obtiene la ruta del archivo
seleccionado
         File f = new File(ruta);              //se crea un File con el archivo seleccionado
         entrada = new Scanner(f);         //se crea un Scanner para leer el archivo
         if (entrada.hasNext()) {             //si el archivo no está vacío
            cadena = entrada.nextLine();  //lectura de la primera línea
            mayor = menor = cadena;      //se toma como mayor y menor
         }

         while (entrada.hasNext()) {        //resto de líneas del fichero
            cadena = entrada.nextLine();  //se lee la siguiente línea
            if (cadena.length() > mayor.length()) { //si su longitud es mayor que la longitud de
la mayor actual
               mayor = cadena;  //se toma como mayor
```

```
            } else if (cadena.length() < menor.length()) { //sino si su longitud es menor que la
longitud de la menor actual
                menor = cadena;
            }
        }

        if (mayor == null || menor == null) {
            System.out.println("Fichero vacío");
        } else {
            System.out.println("Línea más larga:");
            System.out.println(mayor);
            System.out.println("Línea más corta:");
            System.out.println(menor);
        }
      } catch (FileNotFoundException e) {
        System.out.println(e.getMessage());
      } catch (NullPointerException e) { //si se cierra la ventana de selección del archivo sin
haberlo seleccionado.
        System.out.println("No se ha seleccionado ningún archivo");
      } catch (Exception e) {
        System.out.println(e.getMessage());
      } finally {
        if (entrada != null) {
          entrada.close();
        }
      }
   }
}
```

Buscar palabras o cadenas de texto en un archivo

Programa Java para buscar una palabra o una cadena en un fichero de texto.

El programa pedirá que se introduzca una palabra o un texto por teclado y realizará su búsqueda por todo el archivo. Se mostrará por pantalla el número de línea y el contenido de la

línea del fichero que contiene la cadena buscada. Si la cadena buscada aparece en varias líneas se mostrarán todas ellas. Si el fichero no contiene el texto buscado se mostrará un mensaje indicándolo.

Por ejemplo, tenemos un fichero de texto llamado TemaFicheros.txt con el siguiente contenido:

A partir de Java 7 se puede usar la instrucción try-with-resources.
Un resource (recurso) es un objeto que necesita ser cerrado.
Un try-with-resources asegura que estos objetos serán cerrados.
Un objeto AutoCloseable puede ser usado como recurso.

Si el texto a buscar es recurso el programa mostrará por pantalla:

Archivo: TemaFicheros.txt
Texto a buscar: recurso
Linea 2: Un resource (recurso) es un objeto que necesita ser cerrado.
Linea 4: Un objeto AutoCloseable puede ser usado como recurso.

Si el texto a buscar ahora es array el programa mostrará por pantalla:

Archivo: TemaFicheros.txt
Texto a buscar: array
array no se ha encontrado en el archivo

Solución:
Para resolver el ejercicio de búsqueda de texto en ficheros leeremos el fichero línea a línea utilizando la clase Scanner. Para cada línea se comprueba si contiene o no el texto buscado, si la línea contiene lo que buscamos se muestra por pantalla junto a su número de línea. Para obtener el número de línea utilizaremos un contador que se incrementa cada vez que se lea una nueva línea desde el archivo. Se utilizará una variable de tipo boolean para saber si se ha encontrado o no el texto buscado.

Código del programa Java de búsqueda de texto en archivos:

```
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;
import javax.swing.JFileChooser;

public class BurcarTextoEnArchivo {
```

```
    public static void main(String[] args) {

        Scanner entrada = null;
        String linea;
        int numeroDeLinea = 1;
        boolean contiene = false;
        Scanner sc = new Scanner(System.in);

        //Para seleccionar el archivo
        JFileChooser j = new JFileChooser();
        j.showOpenDialog(j);

        //Introducimos el texto a buscar
        System.out.print("Introduce texto a buscar: ");
        String texto = sc.nextLine();

        try {
            //guardamos el path del fichero en la variable ruta
            String ruta = j.getSelectedFile().getAbsolutePath();
            //creamos un objeto File asociado al fichero seleccionado
            File f = new File(ruta);
            //creamos un Scanner para leer el fichero
            entrada = new Scanner(f);
            //mostramos el nombre del fichero
            System.out.println("Archivo: " + f.getName());
            //mostramos el texto a buscar
            System.out.println("Texto a buscar: " + texto);
            while (entrada.hasNext()) { //mientras no se llegue al final del fichero
                linea = entrada.nextLine();  //se lee una línea
                if (linea.contains(texto)) {   //si la línea contiene el texto buscado se muestra por
pantalla
                    System.out.println("Linea " + numeroDeLinea + ": " + linea);
                    contiene = true;
                }
                numeroDeLinea++; //se incrementa el contador de líneas
            }
            if(!contiene){ //si el archivo no contienen el texto se muestra un mensaje indicándolo

                System.out.println(texto + " no se ha encontrado en el archivo");
            }
        } catch (FileNotFoundException e) {
```

```
            System.out.println(e.toString());
        } catch (NullPointerException e) {
            System.out.println(e.toString() + "No ha seleccionado ningún archivo");
        } catch (Exception e) {
            System.out.println(e.toString());
        } finally {
            if (entrada != null) {
                entrada.close();
            }
        }
    }
}//Fin programa de búsqueda de texto en un archivo
```

Crear un fichero de texto en Java

Crear un fichero de texto en Java con caracteres aleatorios.

Programa Java para crear un fichero de texto que contenga caracteres obtenidos de forma aleatoria. Los caracteres a incluir en el fichero serán las letras de la A a la Z (mayúsculas y minúsculas incluida la ñ) y espacios en blanco. El número total de caracteres a escribir se pide por teclado.

Por ejemplo, si el número de caracteres a escribir en el fichero es 30 el contenido del fichero creado podría ser este:

kUIdzzMVjnRÑhyUPDANgIXmnb Pkhl

El fichero se llamará "caracteres.txt". Se debe crear en la carpeta "ficheros" en la unidad C:

Solución:
Para crear el fichero de texto utilizaremos la clase PrintWriter:

PrintWriter salida = new PrintWriter("c:/ficheros/caracteres.txt")

Utilizaremos la clase Random para obtener números de forma aleatoria. Obtendremos números entre 0 y 255 que es el rango de caracteres de la tabla ASCII. Una vez obtenido el

número comprobaremos si corresponde a una letra mayúscula o minúscula o a un espacio en blanco o a la letra ñ. Si es así se escribe en el fichero de texto.

Código del programa Java para crear un fichero y escribir caracteres aleatorios:

```
import java.io.FileNotFoundException;
import java.io.PrintWriter;
import java.util.Random;
import java.util.Scanner;

public class CrearArchivoCaracteresAleatorios {

    public static void main(String[] args) {

        Random rnd = new Random();
        Scanner sc = new Scanner(System.in);
        int cont = 0, n, cantidad;

        do{
            System.out.println("Introduce número de caracteres: ");
            cantidad = sc.nextInt();
        }while(cantidad < 1);

        try (PrintWriter salida = new PrintWriter("c:/ficheros/caracteres.txt")) {

            while (cont < cantidad) {

                n = rnd.nextInt(255); //se obtiene un número

                //si el número corresponde a una letra o a un espacio se escribe en el fichero
                if ((Character.toUpperCase((char) n) >= 'A'
                        && Character.toUpperCase((char) n) <= 'Z')
                        || ((char) n == ' ')
                        || ((char) n == 'ñ')
                        || ((char) n == 'Ñ')) {
                    salida.print((char) n);
                    cont++;
                }
            }

        } catch (FileNotFoundException e) {
```

```
            System.out.println(e.getMessage());
        }
    }
}
```

Leer números de un archivo de texto

Programa Java que lee un archivo de texto que contiene números de tipo int y double. El archivo a leer está formado por dos líneas. La primera línea del fichero contiene números enteros separados por espacios en blanco. La segunda línea contiene números de tipo double separados también por espacios en blanco.
Por ejemplo:
2 6 -1 0 5 10 1 8 2 100
5,75 -8,25 4,25
No conocemos a priori la cantidad de números que hay en cada línea del archivo.
El programa debe leer el archivo de texto, mostrar por pantalla los números enteros y su suma y a continuación mostrar por pantalla los números double y su suma.
Por ejemplo, si el archivo contiene los valores anteriores, el programa mostrará por pantalla:
Números de tipo int:
2 6 -1 0 5 10 1 8 2 100
Suma de los int: 133
Números de tipo double:
5.75 -8.25 4.25
Suma de los doubles: 1.75
Solución:
Para leer el archivo de texto utilizaremos la clase Scanner.
Sabemos que en la primera línea del archivo se encuentran los números enteros. Para leer cada número entero del archivo utilizaremos el método nextInt() de Scanner. Como no sabemos cuántos números hay en la línea, para poder leerlos todos utilizaremos el método hasNextInt() de Scanner dentro de un bucle while. El método hasNextInt() devuelve true cuando lo siguiente que se va a leer del archivo es un entero, por lo tanto podemos usarlo para saber si quedan más enteros por leer.
Cuando hasNextInt() devuelva false se habrán acabado lo números enteros y debemos empezar a leer los de tipo double. Cada double del archivo se leerá con en método nextDouble(). Utilizaremos ahora el método hasNextDouble() dentro de un bucle while para poder leerlos todos.

Código del programa Java para leer números de un archivo de texto:

```
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;

public class LeerArchivoTextoConNumeros {

    public static void main(String[] args) {
        int numeroEntero, sumaInt = 0;
        double numeroDouble, sumaDouble = 0;

        File f = new File("c:/ficheros/numeros.txt");

        try  (Scanner entrada = new Scanner(f)) {

            //Primero están todos los int seguidos
            System.out.println("Números de tipo int: ");

            while (entrada.hasNextInt()) { //mientras queden enteros por leer
                numeroEntero = entrada.nextInt(); //se lee un entero del archivo
                System.out.print(numeroEntero + " "); //se muestra por pantalla
                sumaInt = sumaInt + numeroEntero;  //se suma
            }

            //cuando acaba la lectura de enteros se muestra su suma
            System.out.println("\nSuma de los int: " + sumaInt);

            //Cuando terminan los int empiezan los double
            System.out.println("Números de tipo double: ");

            while (entrada.hasNextDouble()) { //mientras queden double por leer
                numeroDouble = entrada.nextDouble(); //se lee un double del archivo
                System.out.print(numeroDouble + " ");  //se muestra por pantalla
                sumaDouble = sumaDouble + numeroDouble; //se suma

            }

            //cuando acaba la lectura de double se muestra su suma
            System.out.println("\nSuma de los doubles: " + sumaDouble);
```

```
        } catch (FileNotFoundException e) {
            System.out.println(e.toString());
        } catch (Exception e) {
            System.out.println(e.toString());
        }
    }
}
```

Java Ejercicios Básicos Resueltos 3

Ejercicios básicos Java con estructura secuencial

Relación Nº 3: Ejercicios 8, 9 y 10

Ejercicio 8:
Programa que tome como dato de entrada un número que corresponde a la longitud del radio una esfera y nos calcula y escribe el volumen de la esfera que se corresponden con dicho radio.

La fórmula para calcular el volumen de la esfera es
v = (4/3)*PI*r^3

```
/*
 * Programa que calcula el volumen de una esfera
 */
import java.util.*;
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        double radio, volumen;
        System.out.print("Introduzca radio de la esfera: ");
        radio = sc.nextDouble();
        System.out.println("Volumen de la esfera de radio " + radio + " = "
                    + (4.0/3)* Math.PI * Math.pow(radio, 3));
```

```
    }
}
```

La operación para calcular el volumen es: (4.0/3)* Math.PI * Math.pow(radio, 3))

Debemos tener cuidado con la división entre números enteros. Si hacemos 4/3 nos da como resultado 1, por eso se debe escribir al menos uno de los dos números como double. En este caso se ha puesto el numerador como double simplemente escribiendo 4.0 y de esta forma el resultado de la división 4.0/3 será de tipo double.

Ejercicio 9:
Programa Java que calcule el área de un triángulo en función de las longitudes de sus lados (a, b, c), según la siguiente fórmula:
Area = RaizCuadrada(p*(p-a)*(p-b)*(p-c))
donde p = (a+b+c)/2
Para calcular la raíz cuadrada se utiliza el método Math.sqrt()

```
/*
 * Programa que calcule el área de un triángulo en función de las longitudes de sus lados (a, b, c)
 * según la siguiente fórmula: area=raiz2(p(p-a)(p-b)(p-c)) donde p = (a+b+c)/2
 */
import java.util.*;
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        double a,b,c,p;
        System.out.print("Introduzca longitud del primer lado del triángulo: ");
        a = sc.nextDouble();
        System.out.print("Introduzca longitud del segundo lado del triángulo: ");
        b = sc.nextDouble();
        System.out.print("Introduzca longitud del tercer lado del triángulo: ");
        c = sc.nextDouble();
        p = (a+b+c)/2;
    System.out.println("Area -> " +  Math.sqrt(p*(p-a)*(p-b)*(p-c)));
    }
}
```

Ejercicio 10:
Programa Java que lea un número entero de 3 cifras y muestre por separado las cifras del número.

```
/*
 * Programa que lea un número de 3 cifras y muestre por pantalla las cifras del número
 */
import java.util.*;
public class Main{
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      int N;
      System.out.print("Introduzca valor de N: ");
      N = sc.nextInt(); //supondremos que el número introducido tiene 3 cifras
      System.out.println("Primera cifra de " + N + " -> " + (N/100));
      System.out.println("Cifra central de " + N + " -> " + (N/10)%10);
      System.out.println("Última cifra  de " + N + " -> " + (N%10));

   }
}
```

Recuerda que la división entre enteros da como resultado la parte entera de la división (sin decimales). Si por ejemplo N = 123 la operación N/10 da como resultado 12 y no 12.3
Recuerda que el operador % obtiene el resto de la división.