PROGRAMACIÓN ORIENTADA A OBJETOS
(Laboratorio de Prácticas)

Titulaciones de Grado en Ingeniería de Computadores, Ingeniería Informática y Sistemas de Información

Sesión 3
MODULARIDAD Y UTILIZACIÓN DE CLASES BÁSICAS EN JAVA

1. Modularidad

En este apartado realizaremos una primera aproximación a la modularidad dentro de una clase en Java realizando descomposición funcional. La siguiente aproximación la realizaremos en la siguiente sesión descomponiendo la aplicación en clases.

1.1 Ejemplo 1

El siguiente ejemplo muestra una clase con un método auxiliar que permite escribir el mensaje "Eco…" tantas veces como le indiquemos.

```
import java.util.Scanner;

public class Eco {

    public static void main(String[] args) {
        Scanner entrada = new Scanner(System.in);
        System.out.println("Introduce un número:");
        int n = entrada.nextInt();
        eco(n); //llamada a la función
    }

    public static void eco(int veces) {
        for (int i = 0; i < veces; i++) {
            System.out.println("Eco...");
        }
    }
}
```

1.2 Ejemplo 2

El siguiente ejemplo muestra el paso de parámetros por valor y por referencia.

```
//Paso de parámetros por valor y por referencia
public class Parametros {
      static String global= "global";     //variable global a la clase

      public static void main(String args[]) {
            int num = 1;                  //por valor
            int numeros[] = {1,2,3,4,5};  //los arrays se pasan por referencia
            String cadena = "abc";        //por valor

            System.out.println("\nNúmero antes de la llamada al método:"+num);
            System.out.println("\nArray antes de la llamada al método:");
            for (int i=0; i<5; i++) {
                  System.out.print(" "+numeros[i]);
```

```
            }
            System.out.println("\nCadena antes de la llamada al método:"+cadena);
            System.out.println("\nGlobal antes de la llamada al método:"+global);

            //llamamos al método
            convierte(num, numeros, cadena);

            System.out.println("\nNúmero después de la llamada al método:"+num);
            System.out.println("\nArray después de la llamada al método:");
            for (int i=0; i<5; i++) {
                  System.out.print(" "+numeros[i]);
            }
            System.out.println("\nCadena después de la llamada al método:"+cadena);

            System.out.println("\nGlobal después de la llamada al método:"+global);
      }

      public static void convierte(int num, int numeros[], String cadena) {
            num += 1;
            //sumamos 1 al array de enteros
            for (int i=0; i<5; i++) {
                  numeros[i] += 1;
            }
            cadena = "def";
            global = "nueva";
      }
}
```

1.3 Ejemplo 3

En el siguiente ejemplo calculamos la potencia de un número. Para realizarlo nos basamos en la clase Math que define múltiples métodos matemáticos.

```
import java.io.*;

public class Potencia {
      public static void main(String args[]) throws IOException  {
            BufferedReader entrada=
                  new BufferedReader(new InputStreamReader(System.in));

            double base = 0, exponente = 0, resultado;

            System.out.println("\nIntroduce la base:");
            base = Double.parseDouble(entrada.readLine());

            System.out.println("\nIntroduce el exponente:");
            exponente = Double.parseDouble(entrada.readLine());

            //llamamos al método
            resultado = calcula(base, exponente);
            System.out.println("\n\nEl resultado es: "+resultado);
      }

      public static double calcula(double num1, double num2) {
            //Calculamos con el método pow de la clase Math
            return Math.pow(num1, num2);
      }
}
```

1.4 Ejercicios modularidad

1. Realizar un programa que imprima todos los números de un rango de valores. El programa pedirá dos números (el menor y el mayor) y llamará a un método auxiliar encargado de imprimir todos los que se encuentren en su rango.
2. Reutilizando el programa anterior escribir solo los números pares en el rango de valores. El método auxiliar encargado de imprimir todos los valores que se encuentren en su rango tiene que hacer uso de otra función llamada esPar(int num) que devolverá un booleano indicando si el número es par o no.
3. Realizar un programa que determine cuál es el mayor de dos números. El programa tendrá un método mayor que recibirá dos valores de tipo entero y devolverá cual es el mayor. Desde el método main se pedirá al usuario los valores y se llamará al método mayor.
4. Repetir el ejercicio anterior con una versión que devuelva el mayor de tres números pero reutilizando el método mayor de dos números.
5. Realizar un programa que permita simular una calculadora. El programa tendrá un método calculadora que recibirá dos valores de tipo double y un carácter que indicará la operación a realizar (+,-,*,/). Desde el método main se pedirá al usuario los valores y la operación y se llamará al método calculadora.

2. Arrays

Para familiarizarse con el uso de los arrays y clases básicas de Java presentamos el siguiente ejemplo, que genera de forma aleatoria 10 números entre el 0 y el 15 (haciendo uso de la clase Random) y a continuación nos indica que introduzcamos un número y comprueba si ese número está en el array (utilizando la clase Arrays):

```
import java.io.*;
import java.util.*;

public class Adivina {

    public static void main(String args[]) throws IOException {
        BufferedReader entrada = new BufferedReader(new InputStreamReader(System.in));
        int num;
        int numeros[] = new int[10];
        boolean adivinado = false;
        Random rand = new Random();

        //generamos números aleatorios entre el 0 y el 15
        for (int i = 0; i < numeros.length; i++) {
            numeros[i] = rand.nextInt(16);
        }

        //Ordenamos el array
        Arrays.sort(numeros);

        //Pedimos el número
        System.out.println("Introduzca el número (0-15):");
        num = Integer.parseInt(entrada.readLine());

        //llamamos al método
        adivinado = adivinado(numeros, num);

        if (adivinado) {
            System.out.println("\nNúmero adivinado");
        } else {
            System.out.println("\nNúmero NO adivinado");
        }

        //Imprimimos todos los números
```

```
        System.out.println("\nTodos los números:");
        for (int i = 0; i < numeros.length; i++) {
            System.out.print(" " + numeros[i]);
        }
    }

    public static boolean adivinado(int numeros[], int num) {
        return (Arrays.binarySearch(numeros, num) >= 0);
    }
}
```

2.1 Ejercicios sobre Arrays

A continuación se proponen una serie de ejercicios sobre arrays, en estos hay que tratar de modularizar los programas lo más posible creando la mayor cantidad de métodos.

1. Escribir un programa que genere 100 números enteros aleatorios entre el 1 y el 500, los almacene en un array, los ordene y a continuación genere un array de caracteres que contenga una 'p' en las posiciones donde hay números pares y una 'i' en los impares. Mostrar el contenido de ambos arrays en filas de 10 elementos formados por pares de número y letra. Ejemplo:

   5 i - 9 i - 13 i - 18 p - 19 i - 20 p - 23 i - 34 p - 42 p - 44 p -

   46 p - 63 i - 76 p - 77 i - 80 p - 81 i - 89 i - 94 p - 97 i - 100 p -

   101 i - 103 i - 105 i - 114 p - 117 i - 119 i - 126 p - 126 p - 142 p - 144 p -

   145 i - 148 p - 153 i - 169 i - 172 p - 180 p - 181 i - 200 p - 202 p - 204 p -

   207 i - 210 p - 211 i - 212 p - 213 i - 215 i - 218 p - 221 i - 225 i - 236 p -

   241 i - 249 i - 254 p - 258 p - 265 i - 270 p - 272 p - 287 i - 288 p - 292 p -

   298 p - 304 p - 305 i - 305 i - 315 i - 318 p - 326 p - 328 p - 329 i - 330 p -

   335 i - 337 i - 342 p - 346 p - 360 p - 361 i - 363 i - 364 p - 364 p - 370 p -

   371 i - 384 p - 386 p - 394 p - 395 i - 399 i - 402 p - 404 p - 427 i - 436 p -

   447 i - 454 p - 456 p - 466 p - 469 i - 481 i - 482 p - 488 p - 490 p - 492 p –

2. Escribir un programa que genere 50 números enteros aleatorios entre el 97 y el 123 que representarán las letras del alfabeto (exceptuando la ñ), los almacene en un array de caracteres, y cuente cuantas vocales se han generado. *Nota: para pasar del código entero a carácter se utiliza: char c = (char) numero;*

3. Escribir un programa que sea capaz de calcular la letra de un NIF a partir del número del DNI. El programa debe poseer al menos un método encargado de pedir al usuario el DNI de 8 dígitos y otro que calculará la letra del NIF (se pueden añadir más métodos auxiliares). Al finalizar el programa se debe presentar el NIF completo con el formato: ocho dígitos, un guion y la letra en mayúscula; por ejemplo: 00395469-F. La letra se calculará de la siguiente forma: Se obtiene el resto de la división entera del número del DNI entre 23 y se usa la siguiente tabla para obtener la letra que corresponde, esta tabla debe estar almacenada en un array para buscar la letra por su posición.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
T	R	W	A	G	M	Y	F	P	D	X	B	N	J	Z	S	Q	V	H	L	C	K	E

4. Realizar un programa que se encargue de crear una matriz de 3 filas por 20 columnas que representará los sueldos de los empleados de una empresa. En la primera fila se deben generar 20 números decimales aleatorios entre 0 y 300.000 € representando los salarios brutos de los empleados, la segunda fila representará la retención de estos salarios, calculada a partir de la siguiente tabla y por último la tercera fila representará el salario neto, es decir, el salario bruto menos la retención. Presentar por pantalla los datos de la matriz con la información de cada empleado. Ejemplo: *Sueldo del empleado 0: SB: 113763.05745992783 - R: 53468.63700616608 - SN: 60294.420453761755*

Salario Bruto	Retención
Desde 0 € a 17.707	24,00%
Desde 17.708 a 33.007	30,00%
Desde 33.008 a 53.407	40,00%
Desde 53.408 a 120.000	47,00%
Desde 120.001 a 175.000	49,00%
Desde 175.001 a 300.000	51,00%

Nota: De forma voluntaria intentar presentar los sueldos con separadores de miles y solo dos decimales. Ejemplo: Sueldo del empleado 0: SB: 40.185,83 - R: 16.074,33 - SN: 24.111,5

3. Ejercicios cadenas de caracteres

- Haciendo uso de la clase **String** hacer los siguientes ejercicios:
  1. Escribir un programa que pida una palabra y un entero n y vaya rotando el carácter inicial de la palabra, al final de la misma, tantas veces como indique n. (Por ejemplo, “monja”,3 debe devolver “jamon” y “monja”;5 “monja”).
  2. Escribir un programa que solicite una cadena y una letra y nos devuelva las posiciones que ocupa esa letra en la cadena.
  3. Escribir un programa que nos diga cuantas veces se repiten cada una de las vocales en una cadena que el usuario introduce por teclado.
  4. Escribir un programa que reciba un NIF con 9 caracteres (ej. 00395469F) y nos diga si la letra es correcta.

- Haciendo uso de la clase **StringTokenizer** hacer los siguientes ejercicios:
  1. Escribir un programa al que se le pasa un número de teléfono de la forma: 91-8885566. El programa deberá usar la clase StringTokenizer para extraer el código de la comunidad y el resto del número, convertir el código de la comunidad en int y el resto en long, y presentarlo por pantalla.
  2. Escribir un programa que lea una frase y nos diga cuántas palabras la componen.
  3. Escribir un programa que pida una cadena y la divida en palabras y presente las palabras en orden inverso. Consejo: Utilizar la clase **StringBuilder**.

- Haciendo uso del método **split** de la clase **String** repetir ejercicios de la clase StringTokenizer.