Soluciones Ejercicios tema 5

Ejercicio 1

```
// Muestra un saludo por consola
public static void saludar(){
  System.out.println("¡Buenos días, mi gente!");
}
```

Ejercicio 2

```
// Muestra en consola un triángulo de altura 5
public static void triangle5(){
  String esp = " "; // espacio
  String ast = "*"; // asterisco
  String salida = ""; // línea de salida creada con espacios y asteriscos
  // cada línea del triángulo estará formada por espacios y asteríscos
  // los espacios irán disminuyendo de 1 en 1 y los asteriscos aumentando
  // de 2 en 2 para cada línea.
  for (int i = 1; i <= 5; i++){
    salida = esp.repeat(6 - i) + ast.repeat(i * 2 - 1);
    System.out.println(salida);
  }
}
```

Ejercicio 3

```
// muestra por consola un cuadrado de asteriscos de lado 5.
public static void quadrat(){
  String ast = "*";          // asterisco
  String salida = "";        // salida del cuadrado en cada línea
  // itera 5 veces, una para cada línea del cuadrado.
  for (int i = 0; i < 5; i++){
    salida = ast.repeat(5);    // crea la línea de salida
    System.out.println(salida);
  }
}
```

Ejercicio 4

```
// Dibuja un triángulo de tamaño entre 1 y 15.
public static void dibuixaTriangle(){
  Scanner scn = new Scanner(System.in);
  String ast = "*";                    // asterisco
  String esp = " ";                    // espacio
```

```
  String salida = "";
  int altura = 0;                       // altura del triángulo
  boolean alturaIncorrecta = false;     // true si altura tiene un valor inválido
  // solicita al usuario la introducción de la altura entre 1 y 15
  System.out.println("Introduce un valor entre 1 y 15");
  do{ // repite mientras la altura no sea correcta
    alturaIncorrecta = false;
    System.out.print("Altura del triángulo: ");
    while (!scn.hasNextInt()){
      System.out.print("\nValor no válido. Vuelva a introducirla: ");
      scn.nextLine();
    }
    altura = scn.nextInt();
    scn.nextLine();                     // elimina el retorno de carro
    if (altura < 1 || altura > 15){
      System.out.println("El valor debe estar entre 1 y 15.");
      alturaIncorrecta = true;
    }
  }while (alturaIncorrecta);
  // la altura ya es correcta. Dibuja el triángulo
  for (int i = 1; i <= altura; i++){
    salida = esp.repeat(altura + 2 - i);   // 2 espacios de más para evitar problemas
    salida += ast.repeat(i * 2 - 1);      // añade los asteriscos
    System.out.println(salida);
  }
}
```

Ejercicio 5

```
import java.util.Scanner;
public class App {
  public static void main(String[] args) throws Exception {
    double velocidad;
    System.out.println("Cálculo de la velocidad de un objeto");
    System.out.println("Introduce primero el espacio recorrido y luego el tiempo empleado");
    velocidad = (double)llegirEnter() / llegirEnter();
    System.out.println("La velocidad es: " + velocidad);
  }
  // lee un valor entero asegurandose que es correcto y lo devuelve
  public static int llegirEnter () {
    Scanner scn = new Scanner(System.in);
    // solicita el valor
    System.out.print("Introduce un valor entero: ");
    // itera mientras no sea válido
    while(!scn.hasNextInt()){
      System.out.print("\nNo es válido. Introduzca otro: ");
      scn.nextLine();
```

```
    }
    return scn.nextInt();       // devuelve el valor introducido válido
  }
}
```

Ejercicio 6

```
import java.util.Scanner;
public class App {
  public static void main(String[] args) throws Exception {
    double velocidad, espacio, tiempo;
    System.out.println("Cálculo de la velocidad de un objeto");
    // lee el espacio recorrido y el tiempo empleado
    espacio = llegirEnter("Introduce el espacio recorrido:");
    tiempo = llegirEnter("Introduce el tiempo empleado:");
    // calcula y muestra la velocidad
    velocidad = (double)espacio / tiempo;
    System.out.println("La velocidad es: " + velocidad);
  }

  // lee un valor entero válido y lo devuelve. Para leer el valor muestra el mensaje
  // pasado como parámetro
  public static int llegirEnter (String mensaje) {
    Scanner scn = new Scanner(System.in);
    // solicita el valor
    System.out.print(mensaje + " ");
    // itera mientras no sea válido
    while(!scn.hasNextInt()){
      System.out.print("\nNo es válido. " + mensaje + " ");
      scn.nextLine();
    }
    return scn.nextInt(); // devuelve el valor introducido válido
  }
}
```

Ejercicio 7

```
// devuelve el resultado de elevar base a exponente. Si exponente es
// negativo devuelve -1 y muestra un mensaje de error.
public static long potencia (int base, int exponente){
  long potencia = base; // cálculo de la potencia
  if (exponente < 0){
    potencia = -1;            // exponente negativo. Muestra un error y torna -1
    System.out.println("Exponente no válido");
  }else if (exponente == 0){     // exponente igual a cero
    potencia = 1;
  }else if (base == 0){         // la base es 0
```

```
      potencia = 0;
    }else{            // multiplica la base tantas veces como indica el exponente
      for (int i = 1; i < exponente; i++){
        potencia *= base;
      }
    }
    return potencia;    // devuelve el exponente
  }
```

Ejercicio 8

```
  // Devuelve el mcm Mínimo Común Múltiplo de dos números. Si el mcm está fuera del
  // rango de los enteros, devuelve 0 y muestra un mensaje en consola.
  public static int mcm (int v1, int v2){
    int mcm = 0; // mínimo común múltiplo
    int maximo = 0, minimo = 0;
    boolean existe = false;
    maximo = (v1 > v2) ? v1 : v2;        // obtiene el máximo
    // a partir del máximo va aumentando de valor y dividiendo entre v1 y v2
    // si es divisible por los dos lo ha encontrado, sinó, aumenta i sigue buscando
    for (int i = maximo; i < Integer.MAX_VALUE; i++){
      if (i % v1 == 0 && i % v2 == 0){
        existe = true;              // existe el mcm
        mcm = i;
        break;
      }
    }
    // Sale del bucle. Puede haberlo encontrado no.
    if (!existe){                // si no existe muestra un mensaje
      System.out.println("El mcm de " + v1 + " y "+ v2 + " es demasiado grande.");
    }
    return mcm;                  // devuelve el mcm
  }
```

Ejercicio 9

```
  // recibe un entero y devuelve su faltorial
  public static int factorial (int n){
    int factorial = 1;          // cálculo del factorial
    if (n == 0 || n == 1){      // 0! = 1! = 1
      factorial = 1;
    }else if (n == -1){         // factorial de números negativos
      factorial = -1;
    }else if (n < -1){          // factorial de números positivos
      // multiplica todos los números entre n y -1
      for (int i = n; i <= -1; i++){
        factorial *= i;
```

```
    }
  } else {
    // multiplica todos los números entre n y 2
    for (int i = n; i > 1; i--){
      factorial *= i;
    }
  }
  return factorial;
}
```

Ejercicio 10

```
// método que recibe tres parámetros de tipo entero y positivos
// y devuelve el mcd de los tres
public static int mcd(int a, int b, int c){
  int min, resultado;
  // si hay números negativos no hace nada. Devuelve 0
  if (a <= 0 || b <= 0 || c <= 0)
    return 0;

  // obtiene el mínimo de los tres enteros
  if (a <= b && a <= c){
    min = a;
  } else if (b <= a && b <= c){
    min = b;
  }else{
    min = c;
  }
  resultado = min;
  // comienza por el mas pequeño y va bajando hasta llegar a 1 como límite.
  // Si encuentra un número que es divisor exacto de los tres, ese es el mcd
  // 1 es divisor de todos los números.
  for (int i= min; i >= 1; i--){
    if (a % i == 0 && b % i == 0 && c % i == 0){
      // encontrado.
      resultado = i;
      break;
    }
  }
  // Devuelve resultado.
  return resultado;
}
```

Ejercicio 11

```
// devuelve un valor aleatorio entre los dos valores recibidos como paràmetros.
```

```
public static int aleatorioEntre(int min, int max){
  int resultado;
  if (max < min){            // han pasado los parámetros en orden incorrecto
    int aux = max;
    max = min;
    min = aux;
  }
  // Obtiene el valor aleatorio entre min y máx.
  resultado = (int)(Math.random() * (max - min + 1) + min);
  return resultado;
}
```

Ejercicio 12

```
// Muestra en pantalla la tabla del número pasado como parámetro
public static void tablaDel(int valor){
  for (int i = 0; i<= 10; i++){
    System.out.println(valor + " x " + i + " = "+ (i * valor));
  }
}
```

Ejercicio 13

```
// Simula un ascensor. Indicando los pisos por los que pasa
// y los momentos en los que se abren y cierran las puertas.
public static void ascensor(int actual, int destino){
  // comprueba si sube o baja
  if(actual < destino){              // sube
    for (int i = actual; i <= destino; i++){
      if (i == actual){              // comienza el movimiento
        System.out.println("Cerrando Puertas.\nSubiendo");
      } else if (i == destino){      // piso final. LLega a destino
        System.out.println("Destino: piso " + i);
      }else{                         // piso intermedio
        System.out.println("Subiendo: piso " + i);
      }
    }
    // al salir del bucle ya está en el piso de destino.
    System.out.println("Abriendo puertas");
  }else {                            // baja
    for (int i = actual; i >= destino; i--){
      if (i == actual){              // comienza el movimiento
        System.out.println("Cerrando Puertas.\nBajando");
      } else if (i == destino){      // piso de destino
        System.out.println("Destino: piso " + i);
      } else {                       // piso intermedio
        System.out.println("Bajando: piso " + i);
```

```
        }
      }
      // al salir del bucle ya está en el piso de destino.
      System.out.println("Abriendo puertas");
    }
  }
```

Ejercicio 14

```
  // Recibe un array de String y lo muestra por consola, de forma que
  // cada elemento del array aparezca en una línea diferente.
  public static void muestraArray(String[] array){
    // recorre el array mostrando su contenido
    for (String frase : array){
      System.out.println(frase);
    }
  }
```

Ejercicio 15

```
  public static void main(String[] args) throws Exception {
    int vector[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
    System.out.println("Suma: " + sumaArray(vector));
  }

  // recibe como entrada un array de enteros y devuelve la suma de los
  // elementos del array
  public static void sumaArray(int[] array){
    int suma = 0;                    // variable local para calcular la suma
    for (int i = 0; i < array.length; i++){ // recorre el array
      suma += array[i];
    }
    return suma;                // devuelve la suma de los elementos del array
  }
```

Ejercicio 16

```
  public static void main(String[] args) throws Exception {
    int vector[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 14, 21, 18, 32}; // array con enteros
    int datos[]; // array para almacenar los resultados
    // llamo al método pasando el vector de enteros y guardando el vector con la salida
    datos = estadisticasArray(vector);
    System.out.println("Suma: "+ datos[0]);
    System.out.println("Máximo: "+ datos[1]);
    System.out.println("Mínimo: "+ datos[2]);
    System.out.println("Impares: "+ datos[3]);
```

```
    System.out.println("Múltiplos de 7: "+ datos[4]);
  }

  // recibe como entrada un array de enteros y devuelve otro array de enteros de
  // 5 elementos, donde cada elemento representa un cálculo:
  // primero --> Suma de los valores del array
  // segundo --> Máximo de los valores del array
  // tercero --> Mínimo de los valores del array
  // cuarto --> Número de elementos impares
  // quinto --> Número de elementos múltiplos de 7
  public static int[] estadisticasArray(int array[]){
    // declaro las variables locales necesarias para realizar los cálculos
    int[] salida = new int[5];          //array que devolverá con los resultados
    salida[1] = Integer.MIN_VALUE;  // el máximo lo inicia al valor más pequeño para un int
    salida[2] = Integer.MAX_VALUE;  // el mínimo lo inicia al valor más grande para un int
    // el resto de valores se inician a 0 con la declaración
    // recorre el array sumando, buscando el máximo y mínimo, contando impares y múltiplos de 7
    for (int i = 0; i < array.length; i++){       // recorrido del array
        salida[0] += array[i];                // suma el elemento
      if (salida[1] < array[i])               // comprueba máximo y mínimo
        salida[1] = array[i];
      if (salida[2] > array[i])
        salida[2] = array[i];
      if (array[i] % 2 != 0)
        salida[3]++;          // aumenta el número de los impares
      if (array[i] % 7 == 0)
        salida[4]++;          // aumenta el número de los múltiplos de 7
    }
    return salida;
  }
```

Ejercicio 17

```
// Clase que comprueba si el parámetro pasado al llamar a la clase es una
// IP válida o no. La forma de llamar será mediante elnombre de la aplicación
// seguido de una IP válida. Si no es así muestra un mensaje de error.
// Si la llamada es correcta, informa del tipo de IP y de si es pública o
// privada
// Métodos creados:
// main()   método de entrada a la aplicación
// tipoIP   devuelve una cadena con el tipo de IP y si és pública o privada
// esIP     devuelve true si la IP indicada al llamar al programa es válida
//          y false en el caso de que no se ponga IP, se ponga más de una
//          o no sea válida
public class compruebaIP{
  public static void main(String[] args) throws Exception {
    String resultado;
```

```
 String valores[];
 if (esIP(args)){
   // la IP es válida. LLamamos a tipoIP.
   resultado = tipoIP(args[0]);       // llama a tipoIP pasándole la IP
   valores = resultado.split(" ");
   System.out.println("La IP es de tipo " + valores[0]); // tipo de IP A-B-C-D-E
   System.out.println("Su ámbito es " + valores[1]);    // pública o privada
 } else {
   System.out.println("La forma de utilizar el programa es:");
   System.out.println("\n\t\t compruebaIP ip");
   System.out.println("\nDonde ip es 4 números entre 0 y 255 separados por un punto");
 }
}

// Recibe una cadena de entrada con una IP y devuelve una cadena formada por
// una letra seguida de un espacio y una palabra
// La letra indica el tipo de IP (A, B, C, D, E) y la palabra si es PÚBLICA o   PRIVADA
// En el caso de direcciones IP D y E, todas serán PÚBLICAS.
public static String tipoIP(String ip){
 String grupos[] = ip.split("\\.");   // divide la ip por los puntos
 int valor[] = new int[4];          // array con los valores enteros
 String salida = "";               // cadena a devolver
 for (int i = 0; i < grupos.length; i++){
   valor[i] = Integer.parseInt(grupos[i]); // convierte a int
 }
 // una vez tenemos los valores en formato numérico otenemos el tipo de dirección IP
 if (valor[0] <= 127){ // clase A
   salida = "A";
 }else if(valor[0] <= 191){        // clase B
   salida = "B";
 }else if (valor[0] <= 223){       // clase C
   salida = "C";
 }else if (valor[0] < 239){        // clase D
   salida = "D PÚBLICA";
 }else {
   salida = "E PÚBLICA";
 }
 // A continuación se mira si es pública
 if (valor[0] == 10){            // pública tipo A
   salida += " PÚBLICA";
 }else if (valor[0] == 172){
   if (valor[1] >= 16 && valor[1] <= 31){
     salida += " PÚBLICA";       // pública tipo B
   }else
     salida += " PRIVADA";
 }else if (valor[0] == 192 && valor[1] == 168){
   salida += " PÚBLICA";         // pública tipo C
```

```
  }else{                        // resto de casos es privada.
    salida += " PRIVADA";
  }
  return salida;
}

// recibe una cadena de texto y devuelve true si es una IP válida, en caso contrario false.
public static boolean esIP(String[] parametros){
  String ip;
  String grupos[];
  boolean bandera = true;
  int valor;
  if(parametros.length == 0){ // comprueba que haya una cadena
    return false; // no es IP válida
  }
  // comprueba que solo haya un parámetro
  if (parametros.length > 1) return false; // hay más de un parámetro

  ip = parametros[0]; // solo hay un parámetro. Candidato a IP.

  // comprueba si tiene 4 grupos separados por puntos
  grupos = ip.split("\\."); // divide la cadena por el caracter .
  if (grupos.length != 4) return false; // no es ip válida
  // comprueba que los 4 grupos sean numéricos
  bandera = true;
  for (int i = 0; i < grupos.length; i++){
    // cada uno de los dígitos de un grupo deben ser numéricos
    for (int j = 0; j < grupos[i].length(); j++){
      if (!Character.isDigit(grupos[i].charAt(j))){
        bandera = false; // no es un dígito numérico
        break; // sale del bucle
      }
    }
    if (!bandera) break;       // sale del bucle, deja de comprobar
  }
  if (!bandera) return false;   // no es ip válida. Tiene caracteres no numéricos
  // comprueba que los valores estén entre 0 y 255
  bandera = true;
  for (int i = 0; i < grupos.length; i++){
    valor = Integer.parseInt(grupos[i]);
    if (valor < 0 || valor > 255){
      bandera = false;
      break;                // no es un valor válido. Rompe el bucle
    }
  }
  if (!bandera)
    return false;             // no es una ip válida
```

```
    return true;              // ha superado todas las pruebas. IP válida
  }
 }
```

Ejercicio 18

```
// Recibe un parámetro entero, cantidad y devuelve un array con "cantidad"
// valores aleatorios entre 0 y 1.
public static double[] aleatoriosN(int cantidad){
  double salida[] = new double[cantidad];
  for (int i = 0; i < salida.length; i++){
    salida[i] = Math.random();
  }
  return salida;
}
```

Ejercicio 19

```
public static void main(String[] args) throws Exception {
  int array[] = aleatoriosNMinMax(0, 5, 15);
  if (array != null){
    for (int i = 0; i < array.length; i++){
      System.out.println(array[i]);
    }
  }
}

// recibe tres parámetros:
// num: indica el número de enteros aleatorios de debe generar.
// min - max: valores mínimo y máximo entre los que se deben de encontrar
// los valores aleatorios generados
// devuelve un array de enteros con los valores aleatorios generados
public static int[] aleatoriosNMinMax(int num, int min, int max){
  int salida[];
  // si num es negativo o 0 no genera ningún array, devuelve null
  if (num <= 0) return null;
  if (min > max){           // valores min y max invertidos
    int aux = min;
    min = max;
    max = aux;
  }
  // crea el array con el tamaño indicado por num
  salida = new int[num];
  // recorre el array almacenando valores aleatorios enteros entre min y max.
  for (int i = 0; i < num; i++){
    salida[i] = (int)(Math.random() * (max - min + 1)) + min;
```

```
  }
  return salida;        // devuelve el array generado
}
```

Ejercicio 20

```
// Muestra los valores de un array de enteros. Muestra un máximo de 15 valores por línea
public static void muestraArray(int[] array){
  for (int i = 1; i <= array.length; i++){
    if (i % 15 == 0){
      System.out.println(array[i-1]);       // cambia de línea
    }else {
      System.out.print(array[i-1] + ", ");   // muestra el valor del array
    }
  }
  System.out.println("");
}
```

Ejercicio 21

```
public static void main(String[] args) throws Exception {
  int vector[] = {1, 4, 2, 6, 2, 3, 9, 8, 5, 4, 2, 3, 4, 2, 3, 1, 3, 1, 6, 2,
                  8, 1, 9, 1, 1, 2, 4, 6, 8, 9, 1, 2, 9};
  System.out.println(convierteIntString(vector));
}

// Recibe un array de enteros y devuelve un string con los
// valores enteros separados por un espacio
public static String convierteIntString(int[] array){
  String salida = "";
  for (int i = 0; i < array.length; i++){
    salida += String.valueOf(array[i] + " ");   // añade el entero del array
  }
  salida = salida.trim();     // elimina el espacio del final
  return salida;              // devuelve la cadena con los elementos del array
}
```

Ejercicio 22

```
// recibe como entrada una cadena que contiene números enteros separados por un espacio
// devuelve un array de enteros con los valores de la cadena.
public static int[] convierteStringInt(String entrada){
  String vector[];
  int salida[];
  vector = entrada.split(" ");        // divide por espacios
  salida = new int[vector.length];    // reserva espacio para el vector de enteros
```

```
  // recorre vector convirtiendo String's en números enteros.
  for (int i = 0; i < salida.length; i++){
    salida[i] = Integer.parseInt(vector[i]); // convierte String a int
  }
  return salida;                  // devuelve el array de int.
}
```

Ejercicio 23

```
// Recibe dos cadenas de entrada. La primera con una frase formada por palabras y espacios.
// la segunda, una palabra, sin espacios.
// Devuelve true si la palabra se encuentra en la frase. En caso contrario false
public static boolean paraulaTrobada(String frase, String palabra){
  boolean encontrada = false;
  String array[];
  frase = frase.toUpperCase();       // pasa a mayúsculas
  palabra = palabra.toUpperCase();
  array = frase.split(" ");          // divide la frase por espacios
  for (int i = 0; i < array.length; i++){
    if (palabra.equals(array[i])){
      encontrada = true;       //Ha encontrado la palabra buscada
      break;
    }
  }
  return encontrada;             // devuelve si la ha encontrado
}
```

Ejercicio 24

```
public static void main(String[] args) throws Exception {
  String frases[] = {"Los 7 magníficos", "Desayuno con diamantes", "La habitación roja",
                     "Amanecer en el caribe", "La casa de la pradera",
                     "Solo ante el peligro", "Un día de fiesta", "La casa de papel"};
  String palabra = "de";
  int lista[] = paraulaEnArray(frases, palabra);
  for (int i = 0; i < lista.length; i++){
    System.out.println(lista[i]);
  }
}

// Recibe un array de String's que contiene frases y un String que contiene
// una palabra.
// Devuelve un array con los índices de las frases que contienen la palabra
public static int[] paraulaEnArray(String [] frases, String palabra){
  int salida[] = null;         // salida
  int aux[] = null;            // auxiliar. Se utiliza para ir aumentando el array
  // cada vez que se encuentra un elemento
```

```
  // recorre el array buscando la palabra en cada frase. Si la encuentra crea un array
  // añadiendo un elemento nuevo con el índice de la frase donde se encuentra la palabra
  for(int i = 0; i < frases.length; i++){
    if (paraulaTrobada(frases[i], palabra)){
      if (salida != null){
        // aumenta el array añadiendo un nuevo elemento
        salida = Arrays.copyOf(salida, salida.length + 1);
        // en el nuevo elemento pone el índice de la frase que contiene la palabra
        salida[salida.length - 1] = i;
      }else{              // es el primer elemento del array
        salida = new int[1];
        salida[0] = i;
      }
    }
  }
  return salida;
}

////////////////////////////////////////////////////////////////
public static boolean paraulaTrobada(String frase, String palabra){
...   // ejercicio anterior
}
```

Ejercicio 25

```
public static void main(String[] args) throws Exception {
  int v[] = {25, 12, 1, 9, 18, 45, 22, 33, 47, 65, 87, 45, 65, 98, 78, 65,
       45,32, 25, 21, 12 ,13 ,46, 79, 64, 97, 85, 86, 64, 42, 87, 93, 32,
       21, 23, 87, 89, 56, 54, 52, 54, 57, 59, 52, 51, 53, 51, 45, 65, 52,
       25, 75, 85, 95, 65, 15, 65, 32, 65, 85, 75, 75, 45, 85, 65, 95, 95,
       45, 65, 32, 65, 45, 65, 45, 85, 25, 56, 85, 73, 64, 92, 13, 52, 44};
  for (int i = 1; i < 99; i++){
    System.out.println(i + " aparece " + cuentaRepeticiones(array, i) + " veces");
  }
}

// Recibe un array de enteros y un número entero y devuelve la cantidad
// de veces que aparece el entero en el array.
public static int cuentaRepeticiones(int[] lista, int valor){
  int repet = 0;              // contador de repeticines
  for (int i = 0; i < lista.length; i++){
    if (lista[i] == valor){ // encontrado
      repet++;              // aumenta el número de repeticiones
    }
  }
  return repet;               // devuelve el número de repeticiones
}
```

Ejercicio 26

```
public static void main(String[] args) throws Exception {
  String array[] = {"Cada día", "que amanece observo la salida",
       "del Sol por las verdes colinas,", "mientras", "los pájaros cantan hermosas melodías.",
       "Recuerdo, como si fuera", "ayer, como el canto", "de los mirlos",
       "enloqueció al divisar un alcón", "sobrevolando por encima", "de los prados."};
  System.out.println(uneCadenas(array));
}

// Recibe un array de String y devuelve un String con la unión de todas
// las cadenas insertando un espacio entre ellas.
public static String uneCadenas(String[] frases){
  String salida = "";
  for (int i = 0; i < frases.length; i++){ // recorre el array
    if (i != 0) {              // no es la primera cadena a unir
      salida += (" " + frases[i]);
    }else{                     // es la primera cadena a unir
      salida = frases[i];
    }
  }
  return salida;
}
```

Ejercicio 27

```
 // lee un valor entero válido y lo devuelve. Para leer el valor muestra el mensaje
 // pasado como parámetro
 public static int llegirEnter (String mensaje) {
   Scanner scn = new Scanner(System.in);
   // solicita el valor
   System.out.print(mensaje + " ");
   // itera mientras no sea válido
   while(!scn.hasNextInt()){
     System.out.print("\nNo es válido. " + mensaje + " ");
     scn.nextLine();
   }
   return scn.nextInt(); // devuelve el valor introducido válido
 }

// lee un valor entero asegurandose que es correcto y lo devuelve
public static int llegirEnter () {
  Scanner scn = new Scanner(System.in);
  // solicita el valor
  System.out.print("Introduce un valor entero: ");
  // itera mientras no sea válido
```

```
        while(!scn.hasNextInt()){
          System.out.print("\nNo es válido. Introduzca otro: ");
          scn.nextLine();
        }
        return scn.nextInt();        // devuelve el valor introducido válido
      }
```

Ejercicio 28

```
    // método que recibe tres parámetros de tipo entero y positivos
    // y devuelve el mcd de los tres
    public static int mcd(int a, int b, int c){
      int min, resultado;
      // si hay números negativos no hace nada. Devuelve 0
      if (a <= 0 || b <= 0 || c <= 0)
        return 0;

      // obtiene el mínimo de los tres enteros
      if (a <= b && a <= c){
        min = a;
      } else if (b <= a && b <= c){
        min = b;
      }else{
        min = c;
      }
      resultado = min;
      // comienza por el mas pequeño y va bajando hasta llegar a 1 como límite.
      // Si encuentra un número que es divisor exacto de los tres, ese es el mcd
      // 1 es divisor de todos los números.
      for (int i= min; i >= 1; i--){
        if (a % i == 0 && b % i == 0 && c % i == 0){
          // encontrado.
          resultado = i;
          break;
        }
      }
      // Devuelve resultado.
      return resultado;
    }
      // método que recibe dos parámetros de tipo entero y positivos
      // y devuelve el mcd de los tres
      public static int mcd(int a, int b){
       int min, resultado;
       // si hay números negativos no hace nada. Devuelve 0
       if (a < 0 || b == 0)
         return 0;
       // obtiene el mínimo de los dos enteros
```

```
 if (a <= b ){
  min = a;
 } else {
  min = b;
 }
 resultado = min;
 // comienza por el mas pequeño y va bajando hasta llegar a 1 como límite.
 // Si encuentra un número que es divisor exacto de los dos, ese es el mcd
 // 1 es divisor de todos los números.
 for (int i = min; i >= 1; i--){
  if (a % i == 0 && b % i == 0 ){
   // encontrado.
   resultado = i;
   break;
  }
 }
 // Devuelve resultado.
 return resultado;
}

// método que recibe un array de enteros y positivos
// y devuelve el mcd de todos sus valores
public static int mcd(int array[]){
 int min, resultado;
 boolean esDivisor = false;
 // si hay números negativos no hace nada. Devuelve 0
 min = Integer.MAX_VALUE; // mínimo inicial
 // recorre el array buscando el valor mínimo
 for (int i : array){
  if(i <= 0) // si hay valores negativos no hace nada
   return 0;
  if (min > i)
   min = i; // actualiza el mínimo
 }
 resultado = min;
 // comienza por el mas pequeño y va bajando hasta llegar a 1 como límite.
 // Si encuentra un número que es divisor exacto de todos, ese es el mcd
 // 1 es divisor de todos los números.
 for (int i = min; i >= 1; i--){
  // divide todos los elementos del array entre i. Para si encuentra uno que no sea múltiplo
  esDivisor = true; // partimos de considerar que i es dividor de todos
  for (int j = 0; j < array.length; j++){
   if (array[j] % i != 0 ){
    esDivisor = false; // encontrado un elemento que no es múltiplo de i. i no es el mcd
    break;
   }
  }
  if (esDivisor ){ // encontrado.
```

```
      resultado = i;
      break;
    }
   }
   // Devuelve resultado.
   return resultado;
}
```

Para sobrecargar cambiando el tipo a long es todo igual, solo que donde pone int hay que poner long.

Ejercicio 29

```
// Recibe un array de String y lo muestra por consola, de forma que
// cada elemento del array aparezca en una línea diferente.
public static void muestraArray(String[] array){
  // recorre el array mostrando su contenido
  for (String frase : array){
    System.out.println(frase);
  }
}

// Muestra los valores de un array de enteros. Muestra un máximo de 15 valores por línea.
public static void muestraArray(int[] array){
  for (int i = 1; i <= array.length; i++){
    if (i % 15 == 0){
      System.out.println(array[i-1]); // cambia de línea
    }else {
      System.out.print(array[i-1] + ", "); // muestra el valor del array
    }
  }
  System.out.println("");
}

// Recibe un array de double y lo muestra por consola, de forma que
// cada 10 elementos cambia de línea.
public static void muestraArray(double[] array){
  for (int i = 1; i <= array.length; i++){
    if (i % 10 == 0){
      System.out.println(array[i-1]); // cambia de línea
    }else {
      System.out.print(array[i-1] + ", "); // muestra el valor del array
    }
  }
  System.out.println("");
}

// Recibe un array de char y lo muestra por consola, de forma que
// cada 30 chars cambia de línea.
```

```
public static void muestraArray(char[] array){
 for (int i = 1; i <= array.length; i++){
  if (i % 30 == 0){
   System.out.println(array[i-1]); // cambia de línea
  }else {
   System.out.print(array[i-1] + ", "); // muestra el valor del array
  }
 }
 System.out.println("");
}

// Recibe un array de boolean y lo muestra por consola, de forma que
// cada 15 boolens cambia de línea.
public static void muestraArray(boolean [] array){
 for (int i = 1; i <= array.length; i++){
  if (i % 15 == 0){
   System.out.println(array[i-1]); // cambia de línea
  }else {
   System.out.print(array[i-1] + ", "); // muestra el valor del array
  }
 }
 System.out.println("");
}
```

Ejercicio 30

```
// recibe como entrada un array de enteros y devuelve la suma de los
// elementos del array
public static int sumaArray(int[] array){
 int suma = 0; // variable local para calcular la suma
 for (int i = 0; i < array.length; i++){ // recorre el array
  suma += array[i];
 }
 return suma; // devuelve la suma de los elementos del array
}

// recibe como entrada un array de double y devuelve la suma de los
// elementos del array
public static double sumaArray(double[] array){
 double suma = 0.0; // variable local para calcular la suma
 for (int i = 0; i < array.length; i++){ // recorre el array
  suma += array[i];
 }
 return suma; // devuelve la suma de los elementos del array
}
```

Ejercicio 31

```
// Clase Punt, utilizada para almacenar la información de un punto en un plano
public class Punt {
 int x;
 int y;
 // constructor sin parámetros. Inicia x e y a 0
 public Punt(){
  this.x = 0;
  this.y = 0;
 }
 // constructor con parámetros de tipo entero
 public Punt(int x, int y){
  this.x = x;
  this.y = y;
 }
}
```

Ejercicio 32 (*los métodos siguientes irán dentro de la clase Punt*)

```
// devuelve el valor del atributo X
public int getX(){
 return x;
}
// devuelve el valor del atributo Y
public int getY(){
 return y;
}
```

Ejercicio 33 (el siguiente método irá dentro de Punt)

```
// devuelve la distancia entre el objeto actual y el objeto Punt pasado
// como parámetro
public double distancia(Punt p){
  return Math.sqrt(Math.pow((this.x - p.x), 2) + Math.pow((this.y - p.y), 2));
}
```

Ejercicio 34-A

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

public class App {
 public static void main(String[] args) throws Exception {
  Punt p1 = new Punt(5,1);
  Punt p2 = new Punt(9,4);
```

```
  Punt ambu[]; // array de objetos de tipo Punt

  System.out.println("Distancia = " + p1.distancia(p2));
  ambu = llegirAmbulancies("src/ambulancies.txt");
 }

 // lee el fichero pasado como parámetro y almacena el resultado en un array
 // de objetos Punt
 public static Punt[] llegirAmbulancies(String nombre) throws Exception {
  Scanner scn = new Scanner(new File(nombre));
  Punt salida[] = new Punt[199];
  String linea[] = null;
  int i = 0;
  while(scn.hasNext()){
   linea = scn.nextLine().split(",");
   salida[i] = new Punt(Integer.parseInt(linea[0]),Integer.parseInt(linea[1]));
   i++;
  }
  scn.close();
  return salida;
 }
}
```

Ejercicio 34-B

```
 public static void main(String[] args) throws Exception {
  Punt ambu[]; // array de objetos de tipo Punt
  double distMin = Double.MAX_VALUE; // mayor valor para un double
  double distancia = 0;
  Punt dest = new Punt(-4584, 520); // crea el punto de destino.
  int cercano = -1; // índice de la ambulacia maś cercana

  ambu = llegirAmbulancies("src/ambulancies.txt");

  // recorremos el array ambu buscando la ambulancia más próxima al punto
  // de destino: -4584, 520
  for (int i = 0; i < ambu.length; i++){
   distancia = ambu[i].distancia(dest);
   if (distancia < distMin){
    distMin = distancia;
    cercano = i;
   }
  }
  System.out.println("La ambulacia más cercana está en: " +
            ambu[cercano].getX() + ", " + ambu[cercano].getY());
 }
```

Ejercicio 35

```
public static void main(String[] args) throws Exception {
 Punt ambu[]; // array de objetos de tipo Punt
 double distMin = Double.MAX_VALUE; // mayor valor para un double
 double distancia = 0;
 Punt dest = new Punt(-4946, 680); // crea el punto de destino.
 int cercano = -1; // índice de la ambulacia maś cercana

 ambu = llegirAmbulancies("src/ambulancies.txt");

 // recorremos el array ambu buscando la ambulancia más próxima al
 // punto de destino: –4946, 680
 for (int i = 0; i < ambu.length; i++){
  distancia = ambu[i].distancia(dest);
  if (distancia < distMin){
   distMin = distancia;
   cercano = i;
  }
  // muestra si está a una distancia inferior a 1000
  if (distancia <= 1000.0){
   System.out.print("Ubicación (" + ambu[i].getX() + ", " +
                ambu[i].getY() + "). ");
   System.out.println("Distancia " + distancia);
  }
 }
 System.out.println("La ambulacia más cercana está en: " + ambu[cercano].getX() +
              ", " + ambu[cercano].getY());
}
```

Ejercicio 36

```
// Clase Punt, utilizada para almacenar la información de un punto en un plano
public class Punt {
 private int x;
 private int y;
 // constructores
 public Punt(){ ... }
 public Punt(int x, int y){ .... }

 // getters
 public int getX(){ ... }
 public int getY(){ ... }

 // devuelve la distancia entre el objeto actual y el objeto Punt
 //pasado como parámetro
 public double distancia(Punt p){ ... }
```

```
  // establece el valor del atributo X
  void setX(int x){
    this.x = x;
  }
  // establece el valor del atributo X
  void setY(int y){
    this.y = y;
  }
}
```

Ejercicio 37-A

```
// Asigna a X e Y valores aleatorios dentro de un rango
void setAleatori(int minX, int maxX, int minY, int maxY){
  int x = (int)(Math.random() * (maxX - minX + 1) + minX);
  int y = (int)(Math.random() * (maxY - minY + 1) + minY);
  setX(x);
  setY(y);
}
```

Ejercicio 37-B

```
public static void main(String[] args) throws Exception {
  Punt array[] = new Punt[500];      // array de objetos de tipo Punt
  for(int i = 0; i < array.length; i++){
    array[i] = new Punt();           // crea el objeto Punt
    array[i].setAleatori(1, 60, 1, 20); // asigna valores aleatorios al objeto punt
  }
}
```

Ejercicio 38

```
// devuelve true si la distancia entre el objeto actual y el objeto
// Punt pasado como parámetro es inferior o igual a la distancia
// pasada como parámetro
public boolean estaCerca(Punt punt, int distancia) {
  boolean resultado = true;
  if (distancia < this.distancia(punt)){
    resultado = false; // distancia mayor a la pasada
  } else {
    resultado = true; // distancia menor o igual a la pasada
  }
  return resultado;
}
```

Ejercicio 39

```
public static void main(String[] args) throws Exception {
 Punt array[] = new Punt[500]; // array de objetos de tipo Punt
 // inicia el array con puntos aleatorios
 for(int i = 0; i < array.length; i++){
  array[i] = new Punt(); // crea el objeto Punt
  array[i].setAleatori(1, 60, 1, 20); // asigna valores aleatorios al objeto punt
 }
 // muestra los 20 valores de la coordenada Y
 for(int i = 1; i <= 20; i++){
  System.out.println(dibuixaLinia(array, i, '-', 'X'));
 }
}

// lee el fichero pasado como parámetro y almacena el resultado en un array
// de objetos Punt
public static Punt[] llegirAmbulancies(String nombre) throws Exception {...}

// recibe un array de objetos Punt y debe dibujar una línea horizontal correspondiente
// al valor de y indicado en posY. Para representar la existencia de un punto en la línea
// se utiliza caracter hiha, y para representar  no existencia el caracter res
// devuelve una cadena de caracteres con la línea dibujada.
public static String dibuixaLinia (Punt array[], int posY, char res, char hiha){
 String salida = "";
 boolean encontrado = false;
 // Para cada posición de X mira si hay un punto en el array.
 for(int i = 1; i <= 60; i++){ // itera para crear los 60 caracteres de la línea
  // para cada posición busca si hay un punto en el array que tenga
  // el valor y igual al de posY, y el valor de x igual a i
  // si lo encuentra lo marca con hiha, si no lo encuentra lo marca
  // con res
  encontrado = false; // partimos de que no se ha encontrado un punto en la posición
  for (int j = 0; j < array.length; j++ ){
   if (array[j].getY() == posY){ // comprueba que el punto esté en la línea
    if (array[j].getX() == i){  // comprueba que esté en el valor de X actual de la línea de salida
     encontrado = true;
     break; // no hace falta seguir comprobando, ya hay un punto
    }
   }
  }
  if (encontrado)
   salida += String.valueOf(hiha);
  else
   salida += String.valueOf(res);
 }
 return salida;
}
```

Ejercicio 40

```
public static String dibuixaLinia (Punt array[], Punt punto, int posY,
                        int distancia, char res, char prop,
                        char llun, char focus, char sup){
 String salida = "";
 boolean encontrado = false;
 // Para cada posición de X mira si hay un punto en el array.
 for(int i = 1; i <= 60; i++){ // itera para crear los 60 caracteres de la línea
  // para cada posición busca si hay un punto en el array en dicha posición.
  // También si va el objeto punto. Además, si va un punto lo representa de una forma
  // u otra en función de la distancia a la que se encuentra. Si el objeto punto va en
  // la línea, puede coincidir con otro punto, lo representa con sup.
  encontrado = false; // partimos de que no se ha encontrado un punto en la  posición
  char elemento = res; // caracter a añadir
  if (punto.getY() == posY && punto.getX() == i)
   elemento = focus;
  for (int j = 0; j < array.length; j++ ){
   if (array[j].getY() == posY){ // comprueba que el punto esté en la línea
    if (array[j].getX() == i){ // comprueba que esté en el valor de X actual de la línea de salida
     if (elemento == focus)
      elemento = sup; // hay superposicion
     else if (punto.distancia(array[j]) <= distancia)
      elemento = prop;
     else
      elemento = llun;
     encontrado = true;
     break; // no hace falta seguir comprobando, ya hay un punto
    }
   }
  }
  if (encontrado)
   salida += String.valueOf(elemento);
  else
   salida += String.valueOf(elemento);
 }
 return salida;
}
```

Exercici 41

```
// suma los números desde n hasta 1 y devuelve la suma
public int sumaRecursiva(int n){
  if (n == 0)
```

```
    return 0;
  else
    return n + sumaRecursiva(n - 1);
}
```

Exercici 42

```
// imprime de forma recursiva los números desde 1 hasta n.
public void imprimeSerie(int n){
  if (n == 1){
    System.out.print(1 );
  } else {
    imprimeSerie(n - 1);
    System.out.print(" " + n );
  }
}
```

Exercici 43

```
// imprime de forma recursiva los números desde n hasta 1.
public void imprimeSerieIversa(int n){
  if (n == 1){
    System.out.println(1 );
  } else {
    System.out.print( n + " " );
    imprimeSerieIversa(n - 1);
  }
}
```

Exercici 44

```
// Calcula el número de dígitos de un número entero positivo.
public int numeroDigitos(int n){
  if (n < 10){
    return 1;
  } else {
   return 1 + numeroDigitos(n / 10);
  }
}
```

Exercici 45

```
// recibe como entrada un número entero y devuelve su factorial
// utilizando un algoritmo recursivo
```

```java
public int factorial(int n){
  if (n == 0){
    return 1; // 0! = 1
  } else if (n == 1){
    return 1; // 1! = 1
  } else {
    return n * factorial(n - 1); // recursividad
  }
}
```

Exercici 46

Fibonacci (1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, ...)

```java
// Recibe un entero como parámetro de entrada y calcula el valor de la serie
// de fibonacci correspondiente a dicha posición. Los dos primeros elementos
// de la serie de Fib. siempre son 1. El resto de valores es la suma de los
// dos anteriores.
public int fibonacci(int n){
  if (n == 2 || n == 1){
    return 1;
  } else {
    return fibonacci(n - 1) + fibonacci(n - 2);
  }
}
```

Exercici 47

```java
// obtiene el resultado de elevar base a exp de forma recursiva.
// Los parámetros de entrada son base y exp, de tipo entero.
// devuelve un entero.
public int potencia(int base, int exp){
  if (base == 0)
    return 0;
  if (exp == 0)
    return 1;
  if (exp == 1) {
    return base;
  } else {
    return base * potencia(base, exp - 1);
  }
}
```

Exercic 48

```java
// recibe un valor entero y lo muestra invertido
public void invierteNumero(int n){
  // si solo tiene un dígito lo imprime
  if (Math.abs(n) < 10){
    System.out.print(Math.abs(n));
  if (n < 0) // si es negativo imprime el signo
    System.out.println("-");
  else // si es + cambia de línea
    System.out.print("");
  } else {
    // n tiene más de 1 dígito. Imprime el residuo (dígito derecha)
    // y llama a invertir el cociente de la división entre 10 (dígitos
    // de la derecha)
    System.out.print(Math.abs(n % 10) );
    invierteNumero(n / 10);
  }
}
```

Exercici 49

```java
// recibe dos parámetros: base y altura y dibu un rectángulo
// iterando para cada fila
public void dibujaRect(int base, int altura){
  if (altura == 0){
    return; // no hace nada
  } else {
    System.out.println("*".repeat(base));
    dibujaRect(base, altura - 1);
  }
}
```

Exercici 50

```java
// recibe como entrada un String y devuelve true si es palíndromo
public boolean esPalindromo(String frase){
  if (frase.length() == 1)
    return true; // 1 caracter es palindromo
  if (frase.length() == 2){
    if (frase.charAt(0) == frase.charAt(1))
      return true; // 2 caracteres iguales
    else
      return false; // 2 caracteres distintos
  } else {
    // si el primer y último carácter son iguales, comprueba la subcadena
    // que va del segundo carácter al penúltimo.
    if (frase.charAt(0) == frase.charAt(frase.length() - 1)){
      return esPalindromo(frase.substring(1, frase.length()-1));
    } else {
```

```
        return false; // 1r char i último distintos
      }
    }
  }
```

Exercici 51

```
// combruea si el parámetro recibido, n, es un número binario correcto.
// esBinario(n) = esBinario(n % 10) && esBinario(n / 10)
public boolean esBinario(int n){
  int residuo;
  if (n < 10){
    if (n >= 0 && n < 2)
      return true;
    else
      return false;
  } else {
    residuo = n % 10;
    return esBinario(residuo) && esBinario(n / 10);
  }
}
```