Capítulo 4

Abstracción Procedimental y de Datos

Este capítulo se centra en los mecanismos básicos proporcionados por los lenguajes imperativos para la consecución de la abstracción tanto de datos como de procesos. Así, se consideran las matrices (*arrays*) como mecanismo básico de abstracción de datos y se presentan ejemplos de uso de clases predefinidas en Java, como la clase String. Desde el punto de vista procedimental se introduce la modularización por subprogramas, en Java ejemplificada en la definición y escritura de métodos. Como parte del proceso se considera aquí el manejo simple de referencias a matrices y cadenas en conjunción con el paso y devolución de datos en métodos.

Por ello, todos los ejercicios que se detallan a continuación han sido clasificados en los dos siguientes apartados dependiendo de los contenidos que abordan:

- Cadenas (Strings)
- Matrices (*arrays*)
- Métodos
- Métodos y matrices (con especial hincapié en el uso de referencias en métodos)

SECCIÓN 4.1

Cuestiones

4.1.1. Cadenas

1. ¿Cuál es la salida del siguiente programa?

```
class Ejercicio {
  public static void main (String [] args){
    String c1, c2, c3;
    c1 = null;
    c2 = "suma";
    c3 = "Suma";
    c2 = c3;
    c3 = null;
    if ( c3 == c2 ){
      System.out.println("Dentro If");
    } else {
      if (c2.equals("suma")||c1 == null||c3.equals("Suma")){
        System.out.println("Dentro Else-If");
      } else {
        System.out.println("Dentro Else");
      }
    }
  }
}
```

   a) Dentro Else-If

   b) Dentro Else

   c) Dentro If

   d) Error: c3 no se puede asignar a null

   Solución: *a*

2. Suponiendo que se tienen dos cadenas (String) denominadas cadena1 y cadena2 inicializadas de forma totalmente independiente (no refieren al mismo objeto) a cadena1= “Hola” y cadena2=“Hola”. ¿Cuál es el resultado de las siguientes expresiones?

   a) cadena1==cadena2 ()

   b) cadena1.equals(cadena2)

Solución:

a) es false pues el == compara las referencias no los contenidos

b) Es true pues se comparan los contenidos

3. ¿Cuál es la salida del siguiente programa?

```
class Ejercicio {
  static double global = 5;
  public static void main(String[]a){
    metodo(3,"global");
    System.out.println((int)global);
  }
  public static void metodo(int num, String cadena){
    switch(cadena.charAt(cadena.length()-1)){
      case 'f':
        global += num;
        break;
      case 'r':
        global -= num;
        break;
      case 'l':
        global *= num;
        break;
      default:
        global /= num;
    }
  }
}
```

a) 8

b) 15

c) 1

d) 2

Solución: *b*

4. ¿Cuál es la salida del siguiente programa?

```
class Ejercicio {
  public static String duplicate(String s) {
    String t = s + s;
    return t;
```

```
  }
  public static void main (String [] args){
    String s = "H";
    s = duplicate(s);
    String t = "B";
    t = duplicate(duplicate(duplicate(t)));
    System.out.println(s +" "+ t);
  }
}
```

a) Error: Un método no se puede usar como argumento de otro método

b) HH BBBBBBBB

c) HH

d) BBBBBBBB

Solución: *b*

4.1.2. Métodos

5. ¿Cuál es la salida del siguiente programa?

```
class Ejercicio {
  public static void main(String [] args){
    int valor;
    int x,y;
    x=metodo1();
    y=metodo2(x);
    if (x==7 && y==2 || x==6 && y==7)
      valor=metodo3(x,y);
    else
      valor=metodo4(x,y);

    System.out.println (valor);
  }
  public static int metodo1(){
    int a=5;
    int b=++a+1;
    return  b ;
  }
  public static int metodo2(int x){
    int y=(int)17/x;
    return y ;
  }
```

```
  public static int metodo3(int x,int y){
    return y%x ;
  }
  public static int metodo4(int x,int y){
        return (x<y?x+1:x-1);
  }
}
```

Solución: 2

6. Describir y corregir los 4 errores del siguiente código Java:

```
public void metodoA(int[], int pos) {
  int sum=0;
  while( i < pos ) {
    sum = sum + vec[i];
  }
  return sum;
}
```

Solución:

a) El método devuelve una suma, pero está declarado void. Como la suma es de tipo int, el método debe devolver int.

b) El parámetro de entrada tipo array no tiene nombre. Ya que en el método se usa un array denominado vec, parece razonable que ese sea el nombre del parámetro.

c) La variable i no se declara ni se inicializa.

d) Dentro del bucle, la variable i no se modifica, con lo que se produce un bucle infinito.

Método corregido:

```
public int metodoA(int[] vec, int pos) {
  int sum=0;
  int i=0;
  while( i < pos ) {
    sum = sum + vec[i];
    i = i + 1;
  }
  return sum;
}
```

7. ¿Cuál es la salida del siguiente programa? ¿Por qué?

```
class Ejercicio {
  public static void main (String [] args){
    System.out.println (logico());
  }
  public static void logico() {
    boolean log = true;
    return log;
  }
}
```

Solución:

Error, el método no se puede declarar como void, devuelve un valor de tipo boolean.

8. ¿Cuál es la salida del siguiente programa?

```
class Ejercicio {
  public static void main (String[] args) {
    int numero1=4;
    int numero2=3;
    Integer numero3=new Integer (8);
    metodo1 (numero1, numero2, numero3);
    System.out.print(numero1+" ");
    System.out.print(numero2+" ");
    System.out.print(numero3);
  }
  public static void metodo1 (int numero1,
                int numero2, Integer numero3){
    numero1=numero2;
    Integer numero4 = new Integer (9);
    numero2= numero3.intValue();
    numero3= numero4;
    System.out.print(numero1 + " ");
    System.out.print(numero2 + " ");
    System.out.print(numero3 + " ");
  }
}
```

Solución: 3 8 9 4 3 8

9. ¿Cuál es el error del siguiente programa?

```
class Ejercicio {
  public static void main(String [] args) {
    int resultado;
    resultado=Valorb(3);
    System.out.println(resultado);
  }
  public static int Valorb (int a){
    boolean seguir;
    int b=3;
    if (a==0){
      seguir=true;
    }
    else {
      seguir=false;
    }
    do {
      System.out.print("El valor de b es "+b);
      b--;
      if ( b==0) {
        seguir=false;
      }
    }while (seguir);
  }
}
```

   a) Sobra el ; del while(seguir)

   b) if (b==0) debe ser if(b=0)

   c) Falta una sentencia return

   d) La variable a debe ser de tipo double

   Solución: *c*

10. ¿Cuál es la salida del siguiente programa? ¿Y si p=3?

```
class Ejercicio {
  public static void main (String [] args) {
    double nota=0;
    int p;
    int parcial1=5,parcial2=7;
```

```
    //Lectura del valor de p.
    //Por simplicidad lo inicializamos al valor deseado
    p=2;
    if (p==2) {
      nota=calcular_nota(parcial1,parcial2);
    }
    else if (p==3) {
      int parcial3=6;
      nota=calcular_nota(parcial1,parcial2,parcial3);
    }
    System.out.println(nota);
  }

  public static int calcular_nota(int p1,int p2){
    return (p1+p2)/2;
  }

  public static int calcular_nota(int p1,int p2, int p3){
    return (p1+p2+p3)/3;
  }
}
```

a) Daría error: no se pueden usar dos métodos con el mismo nombre

b) Para p=2 daría error y para p=3 imprime 6.0

c) Para p=2 imprime 6.0 y para p=3 imprime 6.0

d) Para p=2 imprime 6.0 y para p=3 daría error

Solución: *c*

11. ¿Cuál es la salida del siguiente programa para p=c? ¿Y para p=d?

```
class Ejercicio {
  public static void main (String [] args) {
    double radio=1;
    int lado=4, perimetro=0;
    char p;
    p=args[0].charAt(0);
    if (p=='c')
      perimetro= perimetro(radio);
    else if (p=='d')
      perimetro=perimetro(lado);
    System.out.println(perimetro);
  }
```

```
  public static int perimetro(double r){
    return (int)(2*Math.PI*r);
  }

  public static int perimetro(int l) {
    return 4*l;
  }
}
```

a) Para p=c imprime 6 y para p=d imprime 16

b) Daría error: no se pueden usar dos métodos con el mismo nombre

c) Para p=c imprime 6 y para p=d imprime 6

d) Para p=c imprime 16 y para p=d imprime 16

Solución: *a*

12. ¿Cuál es la salida del siguiente programa para op=c? ¿Y para op=t?

```
class Ejercicio {

  public static void main (String [] args) {
    int l=3, h=5, sup=0;
    char op;
    op=args[0].charAt(0);
    if (op=='c') {
      sup=calcular_superficie(l);
    }
    else if (op=='t') {
      sup=calcular_superficie(l,h);
    }
    System.out.println(sup);
  }

  public static int calcular_superficie(int l){
    return l*l;
  }

  public static int calcular_superficie(int l, int h) {
    return (int)((l*h)/2);
  }
}
```

a) Para op=c imprime 7 y para op=t imprime 9

b) Para op=c imprime 6 y para op=t imprime 6

c) Daría error: no se pueden usar dos métodos con el mismo nombre

d) Para op=c imprime 9 y para op=t imprime 7

Solución: *d*

13. ¿Cuál es la salida del siguiente programa para p=c? ¿Y para p=r?

```
class Ejercicio{
  public static void main (String [] args) {
    int radio=1, base=2, altura=4, area=0;
    char p;
    p=args[0].charAt(0);

    if (p=='c') {
      area=calcular_area(radio);
          }
    else if (p=='r') {
      area=calcular_area(base,altura);
    }

    System.out.println(area);
  }

  public static int calcular_area(int r){
    return (int)(Math.PI*r*r);
  }

  public static int calcular_area(int b, int a) {
    return b*a;
  }

}
```

a) Daría error: no se pueden usar dos métodos con el mismo nombre

b) Para p=c imprime 3 y para p=r imprime 8

c) Para p=c imprime 3 y para p=r imprime 3

d) Para p=c imprime 8 y para p=r imprime 8

Solución: *b*

14. ¿Cuál es la salida de este programa si el usuario elige la opción 5?

```
import java.io.*;
class Ejercicio {
  static BufferedReader teclado =
       new BufferedReader(new InputStreamReader(System.in));
  static void mostrarmenu(){
    System.out.println("1. Sumar dos n\'umeros");
    System.out.println("2. Restar dos n\'umeros");
    System.out.println("3. Multiplicar dos n\'umeros");
    System.out.println("4. Dividir dos n\'umeros");
    System.out.println("5. Calcular potencia");
    System.out.println("6. Acabar calculadora");
  }
  static int leeropcion() throws IOException{
    System.out.print("\nIntroduzca la opcion deseada...\n");
    int entrada=Integer.parseInt(teclado.readLine());
    while (entrada<1 || entrada>6){
      System.out.print("Opcion incorrecta. Vuelva a intentarlo...");
      entrada=Integer.parseInt(teclado.readLine());
    }
    return entrada;
  }
  public static void main(String args[]) throws IOException{
    int opcion=0;
    System.out.println("Programa que simula el funcionamiento"+
                                          " de una calculadora\n");
    mostrarmenu();
    leeropcion();
    System.out.println (opcion);
  }
}
```

a) 0

b) 5

c) Opción incorrecta. Vuelva a intentarlo...

d) Error: el método mostrarmenu no devuelve ningún valor

Solución: *a*

15. ¿Cuál es el error del siguiente programa?

```
import java.io.*;
class Ejercicio {
  static BufferedReader leer=new BufferedReader(new
                                  InputStreamReader(System.in));
  public static void main (String [] args) throws IOException {
    int resultado = lectura();
    System.out.println(resultado);
  }

  public static int lectura() throws IOException{
    int valor1, valor2, resultado;
    System.out.println("Introduzca un numero");
    valor1 = Integer.parseInt(leer.readLine());
    valor2 = 3;
    if (valor1 == 5){
      resultado = valor1;
      return resultado;
    }
    else {
      resultado = valor1+valor2;
    }
  }
}
```

a) Sobran las llaves del bloque else

b) Sobra throws IOException

c) El método está mal invocado

d) La sentencia return resultado debe estar detrás del bloque if-else

Solución: *d*

16. ¿Cuál es la salida del siguiente programa?

```
class Ejercicio {

  public static void main (String args[]){
    int b= 5;
    System.out.print(metodoA(b));
    System.out.println(metodoA(b));
  }
```

```
  static int metodoA (int y) {
    return y+8;
  }
  static double metodoA (int y) {
    return y+6;
  }
}
```

a) 13 11

b) 13 13

c) Error, se usan dos métodos con la misma firma

d) 11 11

Solución: *c*

17. ¿Cuál es la salida del siguiente programa?

```
class Ejercicio {

  public static void main (String [] args) {
    int n=3;
    long x=metodo(n);
    System.out.println(x);
  }

  static long metodo (int n){
    long suma;
    if  (n>1) {
      long ultimo,penultimo;
      suma=0;
      ultimo=1;
      penultimo=0;
      for (int i=2;i<=n;i++) {
        suma=ultimo+penultimo;
        penultimo=ultimo;
        ultimo=suma;
      }
    } else {
      suma=n;
    }
    return suma;
  }
}
```

Solución: 2

18. ¿Cuál es la salida del siguiente programa?

```
class Ejercicio {
  public static void main (String [] args) {
    System.out.println (metodoA(18,5));
    System.out.println (metodoA(18.0,5.0 ));

  }
  static double metodoA (int a, int b) {
    return(a+2)/(b +3);
  }
  static double metodoA (double a, double b) {
    return (a+2)/(b +3);
  }
}
```

Solución:

```
2.0
2.5
```

19. ¿Cuál es la salida del siguiente programa?

```
class Ejercicio {
  public static void main (String [] args) {
    int a=1, b=1;
    System.out.println(metodo (a,b));
  }
  public static double metodo(int m, int n){
    return (n>=m)?(n+2*m/2*n-m):(n+m/n-2*n);
  }
}
```

Solución: 1.0

20. ¿Qué forma debe tener el segundo bucle for del método main para que la salida del programa fuera la que se encuentra debajo del código de la clase?

```
class Ejercicio {
  public static void main(String [] args ) {
    int filas=5, numero;
    for (int n=0; n<filas; n++){
      for (       ????       ) {
        System.out.print(" ");
      }
```

```
      for (int k=0; k<=n;k++){
        numero=c(n,k);
        System.out.print(numero+" ");
      }
      System.out.println();
    }
  }
  public static int c(int n, int k) {
    return fact(n)/(fact(k)*fact(n-k));
  }
  public static int fact(int n) {
    int producto=1;
    for (int i=1; i<=n;i++){
      producto=producto*i;
    }
    return producto;
  }
}
```

```
    1
   1 1
  1 2 1
 1 3 3 1
1 4 6 4 1
```

a) int i=0; i<filas-n; i++

b) int i=n; i<(2*n-filas)-n; i++

c) int i=0; i<filas+n; i++

d) int i=n; i<filas-n; i++

Solución: *a*

21. Indicar cuál de las siguientes afirmaciones no es válida

a) En una clase es posible definir los métodos *int imprimir (char c)* e *int imprimir (int e)*

b) En una clase es posible definir los métodos *int imprimir (char c)* y *char imprimir (int e)*

c) En una clase es posible definir los métodos *int imprimir (char c)* e *int imprimir (double d)*

d) En una clase es posible definir los métodos *int imprimir (char c)* e *int imprimir (int e, double d)*

Solución: *b*

22. ¿Cuál es la salida del siguiente programa?

```
class Ejercicio {
  public static void main(String[] args) {
    int x = 3;
    int y = 5;
    System.out.println("x=" + x + " y=" + y);
    y = metodo(y,x);
    System.out.println("x=" + x + " y=" + y);
    x = metodo(x,y);
    System.out.println("x=" + x + " y=" + y);
  }
  static int metodo(int x,int y) {
    if (x > y)
    x = x+1;
    else
    x = y-1;
    return x;
  }
}
```

Solución:

```
x=3 y=5
x=3 y=6
x=5 y=6
```

23. ¿Cuál es el error del siguiente programa?

```
import java.util.Scanner;
class Ejercicio {
  static Scanner leer=new Scanner(System.in);
  public static void main (String [] args) {
    int valor = lectura();
    System.out.println(valor);
  }
  public static int lectura(){
    int valor;
    System.out.println("Introduzca un numero");
    valor=leer.nextInt();
    if (valor < 0){
      valor=-valor;
       return valor;
    }
  }
}
```

a) Sobran las llaves del bloque if

b) Falta una cadena entre comillas entre los paréntesis del método System.out.println

c) El método lectura está mal invocado

d) La sentencia return valor debe estar a continuación del bloque if

Solución: *d*

24. ¿Cuál es la salida de este programa?

```
class Ejercicio {
  static int global=1;  // Variable global y static porque no se
                        // crea ningun objeto

  public static void main(String[] args) {
    int local=1;
    System.out.println("local_antes="+local
                                + " global_antes="+global);
    metodo_1(local);
    System.out.println("local_despues="+local
                                + " global_despues="+global);
  }

  public static void metodo_1(int var_local) {
    global=2;
    var_local=2;
    System.out.println("local_metodo="+var_local
                               + " global_metodo="+global);
  }
}
```

Solución:

local_antes=1 global_antes=1
local_metodo=2 global_metodo=2
local_despues=1 global_despues=2

25. ¿Cuál es la salida del siguiente programa si $n \geq 1$?

```
class Ejercicio {
  public static void main (String [] args){
    int n=Integer.parseInt(args[0]);
    int x=Integer.parseInt(args[1]);
```

```
    if (n<1){
      System.out.println("Error : n es menor que 1");
    } else {
      System.out.printf("%d", metodo(n,x));
    }
  }
  public static long metodo(int n, int x){
    long resultado=1;
    while(n>0){
      if(x %2 == 0){
        resultado *= x;
        n--;
      }
      x++;
    }
    return resultado;
  }
}
```

a) El producto de los n primeros números enteros pares a partir de un número x

b) El cociente de los n primeros números enteros pares a partir de un número x

c) El producto de los n primeros números enteros a partir de un número x

d) Error: n es menor que 1

Solución: *a*

26. ¿Cuál es la salida del siguiente programa?

```
import java.util.Locale;
class Ejercicio {
  public static void main (String [] args){
    double suma=metodo(3);
    System.out.printf(Locale.US,"%.1f",suma);
  }
  public static void metodo(int pos) {
    double sum=3.4, i=1;
    do {
      sum = sum + i;
      i++;
    }while( i < pos );
    return sum;
  }
}
```

a) 6.4

b) Error: bucle infinito

c) Error: El tipo de retorno es incorrecto

d) 5.4

Solución: *c*

27. De los cuatro métodos sobrecargados siguientes, ¿qué método llamará f(1,2)?

a)

```
public static void f(int x, double y) {
  System.out.println("f(int, double)");
}
```

b)

```
public static void f(double x, int y) {
  System.out.println("f(double, int)");
}
```

c)

```
public static void f(int x, int y) {
  System.out.println("f(int, int)");
}
```

d)

```
public static void f(double x, double y) {
  System.out.println("f(int, int)");
}
```

Solución: *c*

28. ¿Cuál es la salida del siguiente programa si escogemos la opción *Butaca*?

```
import java.util.Scanner;
class Ejercicio {
  static Scanner teclado=new Scanner(System.in);
  static int opcion;
  static void mostrarmenu(){
    System.out.println("1. Butaca");
    System.out.println("2. Anfiteatro");
    System.out.println("3. Palco");
    System.out.println("4. Entresuelo");
    System.out.println("5. Salir");
  }
```

```
  static void leeropcion() {
    System.out.print("\nIntroduzca la opcion deseada...\n");
    do {
      opcion=teclado.nextInt();
    }while (opcion<1 || opcion>5);
  }
  public static void main(String args[]) {
    mostrarmenu();
    leeropcion();
    System.out.println (opcion);
  }
}
```

a) 1

b) Error: La variable opción no está declarada en el método main

c) No imprime nada porque al invocar al método leeropcion no se recoge el valor que devuelve en ninguna variable

d) Error: Bucle infinito en el método leeropcion

Solución: *a*

4.1.3. Matrices

29. Indicar cuáles de las siguientes declaraciones de matrices no son válidas y por qué:

a) int primos = {2,3,5,7,11};

b) int [] resultados = int [30];

c) int [] primos = new {2,3,5,7,11};

d) char [] notas= new char [];

e) float tiempos[]= {11.3, 21.4, 32.8};

Solución:

a) No es válida. Debe escribirse: int [] primos = {2,3,5,7,11};

b) No es válida. Debe escribirse: int [] resultados= new int [30]

c) No es válida. Debe escribirse: int [] primos= {2,3,5,7,11};

d) No es válida. Debe escribirse: char [] notas= new char [dimension];

e) Válida

30. Indicar y corregir el error del código siguiente:

a)

```
int b[]= new int [10];
for (i=0;i<=b.length;i++)
  b[i]=1;
```

b)

```
int a[][]={{1,2},{3,4}};
a[1,1]=5;
```

Solución:

a) Se hace referencia a la matriz fuera de sus límites, b[10]. Debe cambiarse el operador <= por <.

b) Especificación incorrecta de los índices de la matriz. Cambiar a a[1][1]=5;

31. Suponga que n es de tipo int y que a es una matriz de tipo int[]. Suponga también que la longitud (length) de a es $\geq n$. ¿Para qué valores de n los siguientes fragmentos de programa no tendrán el mismo efecto?

a)

```
int r=0;
int i=0;
while (i<n) {
  r=r+a[i];
  i++;
}
```

b)

```
int r=0;
int i=0;
do {
  r=r+a[i];
  i++;
} while (i<n);
```

Solución: n=0

32. ¿Cuál es la salida del siguiente programa?

```
class Ejercicio {
  public static void main (String[] args) {
    int [] lista = {1,2,1,3,1,4,1,5};
    int i=0; int j=0;
    for (i=0; i<lista.length; i++) {
      if (lista [i] !=1 )
        lista [j++]=lista[i];
      lista[j]=0;
    }
    System.out.print (lista[0]);
    for (i=1; lista[i]!=0;i++)
      System.out.print (" "+lista[i]);
  }
}
```

Solución: 2 3 4 5

33. ¿Cuál es la salida del siguiente programa?

```
class Ejercicio {
  public static void main (String [] args) {
    int i=0;
    {
      int[] a ={0,1,2};
    }
    for (i=0; i<a.length; i++)
      System.out.println (a[i]);
    }
}
```

Solución:

Da error de compilación porque *a* sólo está definida dentro del bloque que la engloba.

34. Indicar cuál de las siguientes declaraciones de matrices es la única válida

    a) int [] resultado= int [20];

    b) int [] numeros= new {1,2,3,4,5};

    c) char [] caracteres= new char [];

    d) float [] temperaturas= {20,25,30,35,40,50};

Solución: *d*

35. Completar el siguiente método que imprime los elementos de una matriz bidimensional (la matriz puede tener distinto número de elementos en cada fila).

```
public static void imprime (int [][] a) {
  for (int i=0; i<a.length; i++) {
    for (int j=0; -------- ; j++){
      System.out.print(a[i][j]+" ");
    }
    System.out.println();
  }
}
```

   a) $j < a[i][j]$

   b) $j < a.length$

   c) $j < a[i].length$

   d) $j < a[j]$

   Solución: *c*

36. ¿Cuál es la salida del siguiente programa?

```
class Ejercicio {
  public static void main (String [ ] args){
    int [][] matriz = {{2,1},{3,4}};
    for (int i=0; i<matriz.length;i++)
      System.out.println(matriz [i][2]);
  }
}
```

   Solución: Matriz fuera de los límites

37. ¿Cuál es la salida del siguiente programa?

```
class Ejercicio {
  public static void main (String [] args) {
    int i=0;
    {
      int[] a = {0,1,2};
      if (Integer.parseInt(args[0]) == 1)  a[2]=3;
    }
    for (i=0; i<a.length; i++)
      System.out.println (a[i]);
  }
}
```

Solución:

Da error de compilación porque *a* sólo está definida dentro del bloque que la engloba.

38. Indicar cuál es el error del siguiente programa:

```
class Ejercicio {
  public static void main (String [] a){
    int b[]= new int [10];
    for (int i=0;i<=b.length;i++){
      b[i]=i;
    }
    System.out.println(b[9]);
  }
}
```

a) Se hace referencia a la matriz fuera de sus límites

b) La declaración de la matriz es incorrecta

c) La matriz no está inicializada

d) El argumento que se pasa al método main debe llamarse args

Solución: *a*

39. Indicar cuál es el error del siguiente fragmento de código:

```
int[] a;
for (int i = 0; i < 10; i++)
  a[i] = i * i;
```

a) La matriz no se puede recorrer hasta 10 porque nos saldríamos fuera de los límites de la matriz

b) El fragmento de código es correcto. No hay error

c) No se ha reservado memoria para a[] con new

d) Falta una llave en el bucle for

Solución: *c*

40. ¿Cuál es la salida del siguiente fragmento de código?

```
int[] a = { 1, 2, 3 };
int[] b = { 1, 2, 3 };
System.out.println(a == b);
```

a) true

b) false

c) Error: *a* y *b* no se puede comparar porque son matrices

d) 1 2 3

Solución: *b*

41. ¿Cuál es la salida del siguiente programa?

```
class Ejercicio {
  public static void main (String [] args){
    int [] a = new int[5];
      for (int i = 0; i < 5; i++)
        a[i] = 4 - i;
      for (int i = 0; i < 5; i++)
        a[i] = a[a[i]];
      for (int i = 0; i < 5; i++)
        System.out.print(a[i]);
  }
}
```

a) 00000

b) 12321

c) 01210

d) Error: la sentencia a[i] = a[a[i]]; es incorrecta

Solución: *c*

4.1.4. Métodos y matrices

42. ¿Cuál es la salida del siguiente programa?

```
class Ejercicio {
  public static void main (String[] args) {
    int numeros[] = {7, 7, 7, 7};
    System.out.println ("El indice buscado es: " +
                    Busqueda.buscar (numeros, numeros[3]));
  }
}

class Ejercicio {
  public static int buscar (int[] numeros, int buscado) {
    int indice = 0;
    while (indice < numeros.length) {
      if (buscado == numeros[indice])
        return indice;  //encontrado
      indice++;
    }
    return -1; //no encontrado
  }
}
```

Solución: El indice buscado es 0

43. ¿Cuál es la salida del siguiente programa?

```
class Ejercicio {

  public static void main (String[] args){
    int [][] matriz={{1,2},{3,4}};
    imprime(matriz);
    intercambia(matriz);
    imprime(matriz);
  }

  public static void imprime (int [][] m){
    for (int i=0; i<m.length;i++){
      for (int j=m.length-1; j>=0;j--){
        System.out.print(m[i][j]);
      }
      System.out.println();
    }
    System.out.println();
  }
```

```
  public static void intercambia (int [][] m){
    int aux;
    aux=m[0][0];
    m[0][0]=m[1][1];
    m[1][1]=aux;
  }
}
```

Solución:

```
21
43

24
13
```

44. ¿Cuál es la salida del siguiente programa?

```
class Ejercicio {
  public static void main (String[] args) {
    int [] alfa={2 ,3 ,4};
    int [] beta=alfa;
    beta[1]=3;
    alfa= modificar (alfa ,beta);
    for (int i=0; i<alfa.length ;i++)
      System . out. print (alfa[i]+" "+beta[i]+" ");
  }
  public static int [] modificar (int [] a, int [] b) {
    int [] c={5 ,6 ,7};
    for (int i=0; i< a.length ; i++)
      c[i]=a[i]+c[i];
    return c;
  }
}
```

Solución: 7 2 9 3 11 4

45. ¿Cuál es la salida del siguiente programa?

```
class Ejercicio {
  public static void main(String [] args) {
    int [] a={0,1,2};
    int [] b=a;
    b[1]=3;
```

```
    a=metodo1(a ,b);
    for (int i=0; i<a.length;i++)
      System.out.print(a[i]+" "+b[i]+" ");
  }
  public static int [] metodo1(int [] a, int [] b) {
    int [] c={3,4,5};
    for (int i=0; i< a.length; i++)
      c[i]=a[i]+b[i];
    return c;
  }
}
```

a) 0 0 6

b) 3 4 2

c) 0 3 1 4 2 5

d) 0 0 6 3 4 2

Solución: *d*

46. ¿Cuál es la salida del siguiente programa?

```
class Ejercicio {
  public static void main(String [] args) {
    int [] b={0,1,2};
    int [] a=b;
    b[1]=3;
    b=metodo1(a);
    for (int i=0; i<a.length;i++)
      System.out.print(a[i]+" "+b[i]+" ");
  }

  public static int [] metodo1(int [] a) {
    int [] c={3,4,5};
    for (int i=0; i< a.length; i++)
      a[i]=a[i]+c[i];
    return c;
  }
}
```

Solución: 3 3 7 4 7 5

47. ¿Cuál es la salida del siguiente programa?

```
class Ejercicio {
   public static void main(String [] args) {
     int [] a={3,4,5};
     int [] b=a;
     b[1]=3;
     a=metodo1(a ,b);
     for (int i=0; i<a.length;i++)
       System.out.print(a[i]+" "+b[i]+" ");
   }

   public static int [] metodo1(int [] a, int [] b) {
     int[] c={0,1,2};
     for (int i=0; i< a.length; i++)
       c[i]=a[i]+b[i];
     return c;
   }
}
```

a) 6 3 6 3 10 5

b) 6 6 10

c) 3 3 5

d) 3 6 3 6 5 10

Solución: *a*

48. ¿Cuál es la salida del siguiente programa?

```
class Ejercicio {
  public static void main (String [] args) {
    int [] a= {3,4,5};
    metodo1(a, a[1]);
    for (int i=0; i<a.length; i++)
      System.out.print (a[i]+ " ");
  }
  static void metodo1(int [] b, int c) {
    c++;
    for (int i=0; i<b.length; i++) {
      b[i] = b[i] + c;
    }
  }
}
```

a) 3 4 5

b) 8 10 10

c) Error

d) 8 9 10

Solución: *d*

49. ¿Cuál es la salida del siguiente programa?

```
class Ejercicio {
  public static void main (String[] args) {
    int[] numeros = {3,7,5,1,2,24,13,6};
    metodo(numeros);
    for (int i=0; i < numeros.length; i++)
      System.out.print (numeros[i++]+" ");
  }

  public static void metodo (int[] numeros) {
    int m, aux;
    for (int j=0; j<numeros.length-1; j++) {
      m=j;
      for (int k=j+1; k<numeros.length; k++)
        if (numeros[k] >numeros[m])
          m=k;
      aux=numeros[m];
      numeros[m]=numeros[j];
      numeros[j]=aux;
    }
    numeros[0]=10;
  }
}
```

Solución: 10 7 5 2

50. ¿Cuál es la salida del siguiente programa?

```
class Ejercicio {
  public static void main ( String [] args ) {
    int [] v1 ={5 , 72, 9, 12, 44, 69};
    int [] v2 ={29 , 0, 85};
    cambia1 ( v1 [4]);
    cambia1 ( v2 [2]);
    cambia2 ( v2 );
    v1 = cambia3 ( v2 );
```

```
    System .out . print ( v1[0]);
    for (int i=1; i< v2.length ;i++)
      System.out.print (" "+ v1[i]);
  }

  public static void cambia1 (int valor ) {
    valor = 0;
    valor = valor +10;
  }

  public static void cambia2 (int [] l ) {
    l[1]= l[2];
  }

  public static int [] cambia3 (int [] l ) {
    for (int i=0; i< l.length ;i++)
      l[i]= l[i]+i;
    return l ;
  }
}
```

Solución: 29 86 87

51. ¿Cuál es la salida del siguiente programa?

```
class Ejercicio {

  public static void main (String [] args) {
    int [] matriz= {1,2,3};
    metodoA(matriz);
    for (int i=0; i<matriz.length; i++)
      System.out.print (matriz[i]++ + " ");
    for (int i=0; i<matriz.length; i++)
      System.out.print (matriz[i] + " ");
  }

  static void metodoA(int [] b) {
    for (int i=0; i<b.length; i++) {
      b[i] = b[i] + 3;
    }
  }
}
```

Solución: 4 5 6 5 6 7

52. ¿Cuál es la salida del siguiente programa?

```
class Ejercicio {
  public static void main (String [] args) {
    int [] a = {0,1,2};
    int [] b = {3,4,5};
    metodo1 (a ,b);
    for (int i=0; i<a.length; i++)
      System.out.print (a[i]+" "+b[i]+" ");
  }
  public static void metodo1 (int[] a, int[] b) {
    a=b;
    a[0]=6;
  }
}
```

Solución: 0 6 1 4 2 5

53. ¿Cuál es la salida del siguiente programa?

```
class Ejercicio {
  public static void main (String [] args) {
    int [] matriz= {1,2,3};
    metodoA(matriz);
    for (int i=0; i<matriz.length; i++)
      System.out.print (++matriz[i]+ " ");
  }
  static void metodoA(int [] b) {
    for (int i=0; i<b.length; i++) {
      b[i] = b[i] + 3;
    }
  }
}
```

Solución: 5 6 7

54. ¿Cuál es la salida del siguiente fragmento de código?

```
class Ejercicio {
  public static void main (String args[]){
    int [] a ={4,3,5};
    int [] b=a;
    b [1]=metodoA(a, b[0]);
    System.out.println (a[2]+" "+b[1]);
  }
```

```
  static int metodoA (int []a, int n) {
    for (int i=0; i<a.length; i++) {
      a[i]=a[i]+n;
    }
    return a[0];
  }
}
```

a) 5 3

b) 5 8

c) 9 8

d) 5 9

Solución: *c*

55. ¿Cuál es la salida del siguiente programa?

```
class Ejercicio {

  public static void main(String [] args) {
    char[] matriz1={'c', 'a', 's', 'a'}, matriz2=null;
    String cadena = "oeoe ";
    matriz1[3]=cadena.charAt(1);

    metodo1(matriz2, matriz1);
    System.out.println(matriz1);

  }

  public static void metodo1(char [] m1, char[] m2) {
    String cadena = "ptpt" ;
    m1=m2;
    m1[0] = cadena.charAt(1) ;
  }
}
```

a) Error porque no se le reservó memoria a *matriz1*.

b) Error porque no se le reservó memoria a *m2*.

c) Imprime por pantalla: paso.

d) Imprime por pantalla: tase.

Solución: *d*

56. ¿Cuál es la salida del siguiente programa?

```
class Ejercicio {
  public static void main(String [] args) {
    int [] Valores1 ={1, 2, 3, 4};
    int [] Valores2 ={9, 8, 7};
    metodo1(Valores1[1]);
    metodo1(Valores2);
    Valores1=metodo2(Valores2);
    for (int i=0; i<Valores1.length; i++)
      System.out.print(" " + Valores1[i]);
  }

  public static void metodo1(int valores) {
    valores = 7;
  }

  public static void metodo1(int [] lista) {
    lista[1]=lista[2];
  }

  public static int [] metodo2(int [] lista) {
    for (int i=0; i<lista.length;i++)
      lista[i]=lista[i]+i;
    return lista;
  }
}
```

Solución: 9 8 9

57. ¿Cuál es la salida del siguiente programa?

```
class Ejercicio {
  public static void main(String [] args) {
    int [] Valores1 ={1, 2, 3, 4, 24, 13};
    int [] Valores2 ={1, 1, 1, 1};
    metodo1(Valores1[1]);
    metodo1(Valores2);
    Valores1=metodo2(Valores2);
    for (int i=0; i<Valores1.length;i++)
      System.out.print(" " + Valores1[i]);
  }

  public static void metodo1(int valores) {
    valores = 7;
  }
```

```
  public static void metodo1(int [] lista) {
    lista[1]=lista[2];
  }
  public static int [] metodo2(int [] lista) {
    for (int i=0; i<lista.length;i++)
      lista[i]=lista[i]+i;
    return lista;
  }
}
```

Solución: 1 2 3 4

58. ¿Cuál es la salida del siguiente programa?

```
class Ejercicio {
  public static void main(String [] args) {
    int[] a={3,4,5};
    int [] b=a;
    b[1]=3;
    a=metodo1(a ,b);
    for (int i=0; i<a.length;i++)
      System.out.print(a[i]+" "+b[i]+" ");
  }
  public static int [] metodo1(int [] d, int [] e) {
    e[1]=4;
    int [] c={0,0,0};
    for (int i=0; i< d.length; i++)
      c[i]=d[i]+e[i];
    return c;
  }
}
```

Solución: 6 3 8 4 10 5

59. ¿Cuál es la salida del siguiente programa?

```
class Ejercicio {

  private String [] cadena1;
  private String [] cadena2;

  public Cadenas (String [] cadena1, String [] cadena2){
    this.cadena1=cadena1;
    this.cadena2=cadena2;
  }
```

```
  public void imprimir_cadena(){
    cadena2=cadena1;
    cadena2[2]=" final ";
    cadena1=modificar_cadena();
    for (int i=0; i<cadena1.length;i++){
      if (cadena1[i].equals(cadena2[i])){
        System.out.print(cadena1[i]);
      }
    }
  }
  private String [] modificar_cadena() {
    cadena1[1]="abstract";
    String [] cad3={" for "," while ", " do-while "};
    for (int i=0; i<cad3.length; i++) {
      cad3 [i]=cadena2[i];
    }
    return cad3;
  }
}
class Ejercicio {
  public static void main(String [] args) {
    String [] cadena1 ={" interface ", " double ", " class "};
    String [] cadena2 ={" int ", " long ", " short "};
    Cadenas cadenas=new Cadenas (cadena1, cadena2);
    cadenas.imprimir_cadena();
  }
}
```

Solución: interface abstract final

60. ¿Cuál es la salida del siguiente programa?

```
class Ejercicio {
  public static void main(String [] args) {
    int[] matriz1={9,1,3,5};
    int [] matriz2=matriz1;
    matriz2[3]=4;
    matriz1[1]=3;
    metodo1(matriz1, matriz2);
    System.out.print(matriz1[1]+" "+matriz2[2]+" ");
  }
  public static void metodo1(int [] m1, int [] m2) {
    m1[1]=4;
    m2[1]=m1[3]+m2[2];
  }
}
```

a) 9 1

b) 1 3

c) 3 1

d) 7 3

Solución: *d*

61. ¿Cuál es la salida del siguiente programa?

```
class Ejercicio {
  public static void main (String [] args)  {
    int m=2,q=2;
    int[][] a= {{1,1},{2,2}};
    int[][] b= {{2,2},{1,1}};
    matrices(a,b);
    for(int i=0; i<m; i++) {
      for(int j=0; j<q; j++) {
        System.out.print(a[i][j]+" ");
      }
    }
  }
  static void matrices(int [][] a, int [][] b) {
    int m=a.length,q=b[0].length,n=a[0].length;
    for(int i=0; i<m; i++) {
      for(int j=0; j<q; j++) {
        a[i][j]=0;
        for(int k=0; k<n; k++) {
           a[i][j]=a [i][j]+a[i][k]*b[k][j];
        }
      }
    }
    a[0][0]=2;
    b[0][1]=4;
 }
}
```

a) 2 4 2 4

b) 2 2 1 1

c) 2 4 2 8

d) Daría error: el método matrices debe devolver la matriz a

Solución: *c*

62. ¿Cuál es la salida del siguiente programa?

```
class Ejercicio {
  public static void main (String [] args)  {
    int[][] m= {{1,1},{3,3}};
    int[][] n= {{1,1},{1,3}};
    metodo(m,n);
    for(int i=0; i<m.length; i++) {
      for(int j=0; j<m[0].length; j++) {
        System.out.print(m[i][j]+" ");
      }
    }
  }
  static void metodo(int [][] e, int [][] f) {
    for(int i=0; i< e.length; i++) {
      for(int j=0; j<f[0].length; j++) {
        e[i][j]=0;
        for(int k=0; k< e[0].length; k++) {
           e[i][j]=e[i][j]+e[i][k]*f[k][j];
        }
      }
    }
    e[1][0]=8;
    f[0][1]=4;
  }
}
```

a) 1 4 8 12

b) Daría error: metodo debe devolver la matriz e

c) 1 4 4 3

d) 1 4 3 3

Solución: *a*

63. ¿Cuál es la salida del siguiente programa?

```
class Ejercicio {

  public static void main(String [] args) {
    String [] cadena1 ={"byte ", "bit ", "palabra "};
    String [] cadena2 ={"protected ", "public ", "private "};
    imprimir_cadena(cadena1,cadena2);
  }
```

```
  static void imprimir_cadena(String []cad1,String []cad2){
    cad2=cad1;
    cad2[2]="binario ";
    cad1=modificar_cadena(cad1,cad2);
    for (int i=0; i<cad1.length;i++){
      if (cad1[i].equals(cad2[i])){
        System.out.print(cad1[i]);
      }
    }
  }
  static String [] modificar_cadena(String [] cade1,
    String [] cade2) {
    cade1[1]="Megabyte ";
    String [] cad3={"Gigabyte ","Megabyte ", "Terabyte "};
    for (int i=0; i<cad3.length; i++) {
      cad3 [i]=cade2[i];
    }
    return cad3;
  }
}
```

Solución: byte Megabyte binario

64. ¿Cuál es la salida del siguiente programa?

```
class Ejercicio {
  public static void main(String [] args) {
    int[] matriz1={1,132,100,99};
    int [] matriz2=matriz1;
    matriz1[0]=101;
    matriz2[2]=30;
    metodo1(matriz1, matriz2);
    System.out.print(matriz1[1]+" "+matriz2[3]+" ");
  }
  public static void metodo1(int [] m1, int [] m2) {
    m1[3]=98;
    m2[1]=m1[0]+m2[2];
  }
}
```

a) 132 99

b) 98 99

c) 131 98

d) 131 99

Solución: *c*

65. ¿Cuál es la salida del siguiente programa?

```
class Ejercicio {
  public static void main(String [] args) {
    int [] veca={3,2,5,10,33};
    int [] vecb=veca;
    veca[1]=54;
    vecb[3]=11;
    cambio (veca, vecb);
    System.out.print(veca[0]+" "+vecb[2]+" ");
  }
  public static void cambio(int [] aa, int [] bb) {
    aa[2]=2;
    bb[0]=aa[2]+bb[2];
  }
}
```

a) 3 5

b) 2 2

c) 4 2

d) 4 5

Solución: *c*

66. ¿Cuál es la salida del siguiente programa?

```
class Ejercicio {
  public static void main(String [] args){
    int [][]matriz={{1},{1,2},{1,2,3}};
    cambiar(matriz);
    System.out.println(
    matriz[matriz.length-1][matriz.length-1]);
  }
  public static void cambiar(int [][]matriz){
    for (int i=matriz.length-1;i>=0;i--)
      for (int j=matriz[i].length-1;j>=0;j--)
        matriz[i][j]=i+j;
  }
}
```

Solución: *4*

67. ¿Cuál es la salida del siguiente programa?

```
class Ejercicio{
  public static void main (String [] args)  {
    int[][] unamatriz= {{2,1},{2,2}};
    int[][] otramatriz= {{2,1},{2,2}};
    calculo(unamatriz,otramatriz);
    for(int i=0; i<unamatriz.length; i++) {
      for(int j=0; j<otramatriz.length; j++) {
        System.out.print(unamatriz[i][j]+" ");
      }
    }
  }
  static void calculo(int [][] um, int [][] om) {
    for(int i=0; i<om.length; i++) {
      for(int j=0; j<um[0].length; j++) {
        om[i][j]=0;
        for(int k=0; k<um[0].length; k++) {
          om[i][j]=om [i][j]+om[i][k]*um[k][j];
        }
      }
    }
    om[0][0]=2;
    um[1][1]=2;
  }
}
```

a) 2 1 2 2

b) 2 4 2 8

c) Daría error: el método calculo debe devolver la matriz om

d) 2 1 2 1

Solución: *a*

68. ¿Cuál es la salida del siguiente programa?

```
class Ejercicio {

  public static void main(String [] args){
    int resultado;
    int m,n;
    m=m1();
    n=m2(m);
```

```
    if (m==5 && n==6 || m==6 && n==3)
      resultado=m3(m,n);
     else
      resultado=m4(m,n);
    System.out.println (resultado);
  }
  public static int m1(){
    int p=5;
    int q=++p;
    return q ;
  }
  public static int m2(int m){
    int y=(int)19/m;
    return y ;
  }
  public static int m3(int m,int n){
    return n%m ;
  }
  public static int m4(int m, int n){
    return (n<m?+n-2:m+2);
  }
}
```

a) 8

b) 1

c) 3

d) 0

Solución: *c*

69. ¿Cuál es la salida del siguiente programa?

```
class Ejercicio {
  public static void main (String [] args){
    int dimension=2;
    int [][]matriz=metodo(dimension);
    for(int i =0; i< dimension; i++){
      for(int j= 0; j< dimension ;j++){
        System.out.print (matriz[i][j]);
      }
      System.out.print (" ");
    }
  }
```

```
  public static int[][] metodo(int dimension){
    int [][] matriz1 = new int[dimension][dimension];
    for(int i =0; i< dimension; i++){
      for(int j= 0; j< dimension ;j++){
        matriz1[i][j]= i+j;
      }
    }
    return matriz1;
  }
}
```

a) 12 10

b) 01 21

c) 12 01

d) 01 12

Solución: *d*

70. ¿Cuál es la salida del siguiente programa?

```
class Ejercicio {
  public static void main (String [] args){
    int [][] matriz={{1,2},{3,4}};
    imprimir_matriz(matriz);
    intercambiar_matriz(matriz);
    imprimir_matriz(matriz);
  }

  public static void imprimir_matriz (int [][] mat){
    for (int p=0; p<mat.length;p++){
      for (int q=mat.length-1; q>=0;q--){
       System.out.print(mat[p][q]);
      }
    }
    System.out.print("  ");
  }

  public static void intercambiar_matriz (int [][] matrix){
    int aux;
    aux=matrix[0][0];
    matrix[0][0]=matrix[1][1];
    matrix[1][1]=aux;
  }
}
```

a) 2143 2413

b) 1234 4231

c) 1234 1234

d) 1234 4321

Solución: *a*

71. ¿Cuál es la salida del siguiente programa?

```
class Ejercicio{
  public static void main (String [] args){
    char [][] matriz={{'a','b'},{'c'}};
    imprime(matriz);
    intercambia(matriz);
    imprime(matriz);
  }
  public static void imprime (char [][] m){
    for (int i=0; i<m.length;i++){
      for (int j=m[i].length-1; j>=0;j--){
        System.out.print(m[i][j]);
      }
    }
    System.out.print(" ");
  }
  public static void intercambia (char [][] m){
    char c;
    c = m[0][0];
    m[0][0] = m[1][0];
    m[1][0] = c;
  }
}
```

a) bac
bca

b) abc
abc

c) abc
cba

d) bac
abc

Solución: *a*

72. ¿Cuál es la salida del siguiente programa?

```
class Ejercicio {
  public static void main(String [] args) {
    int [] m1 ={100, 101, 102, 103, 104, 105, 106, 107, 108, 109};
    int [] m2 ={90, 91, 92, 93};
    metodo1(m1 [7]);
    metodo1(m2 [3]);
    metodo2(m2);
    m1 = metodo3(m2);
    System.out.print(m1[3]);
    for (int i=m2.length-2; i>=0 ;i--)
      System.out.print(" "+m1[i]);
  }
  public static void metodo1(int n) {
    n = 0;
    n=n+10;
  }
  public static void metodo2(int [] matriz1) {
    matriz1[2]=matriz1[0];
  }
  public static int [] metodo3(int [] matriz2) {
    int [] matriz3=new int[matriz2.length];
    for (int i=0; i<matriz3.length;i++)
      matriz3[i]=matriz2[i]+i;
    return matriz3;
  }
}
```

a) 96

b) 90 91 92 93

c) 96 90 92 92

d) 96 92 92 90

Solución: *d*

73. ¿Cuál es la salida del siguiente programa?

```
class Ejercicio {
  public static void main (String [] args) {
    String [] cadena= {"martes", "jueves", "viernes"};
    String [] c1=cadena;
    metodo1(cadena, cadena[2]);
```

```
    c1[0]="lunes";
    c1=null;
    for (int i=0; i<cadena.length; i++)
      System.out.print (cadena[i]);
  }

  static void metodo1(String [] c1, String c2) {
    c2="&";
    for (int i=0; i<c1.length; i++) {
      c1[i] = c2 +c1[i];
    }
  }
}
```

a) martes&jueves&viernes

b) Nada: la cadena apunta a null

c) lunesjuevesviernes

d) lunes&jueves&viernes

Solución: *d*

SECCIÓN 4.2

Problemas

4.2.1. Cadenas

1. Construir un programa que reciba una frase de texto por línea de órdenes y devuelva la frecuencia absoluta de aparición de las diferentes parejas de caracteres consecutivas. Visualizar por pantalla las frecuencias no nulas junto con su pareja de caracteres asociada, a razón de 6 por línea.

   Solución:

```
class Ejercicio {

  public static void main (String argv[]) {
    char c1,c2;
    int i=0,j=0, numparam,contador=0;
    StringBuilder cadena = new StringBuilder();
    int parejas[ ][ ] = new int[27][27];
    numparam = argv.length;

    for (i = 0; i < numparam; i++){
      cadena.append(argv[i].toLowerCase());
      if (i != numparam-1)
        cadena.append(' ');
    }
    System.out.println("Aviso: el espacio en blanco se " +

    for (int k = 0; k < cadena.length()-1; k++){
      c1 = cadena.charAt(k);
      c2= cadena.charAt(k+1);

      if (c1==' ')
        i=26;
      else
        i=c1-'a';

      if (c2==' ')
        j=26;
      else
        j=c2-'a';

      parejas[i][j]++;
    }
```

```
    for(i = 0; i < 27; i++){
      for(j = 0; j < 27; j++){
        if (parejas[i][j]!=0) {
          int char1 ='a'+i;
          int char2= 'a'+j;
          System.out.print((char)char1 +""+
            (char)char2 +":" +parejas[i][j] + " ");
          contador++;
          if (contador%6==0)
            System.out.println();

        }
      }
    }
  }
}
```

2. Construir un programa que extraiga todas las apariciones de un determinado carácter, en una palabra almacenada en un String y devuelva por pantalla el String resultante. Tanto la palabra como el carácter a eliminar serán introducidos en el programa por línea de comandos.

   Solución:

```
class Ejercicio {
  public static void main (String argv[]) {
    String palabra, car;
    if (argv.length == 2) {
      palabra = argv[0];
      car = argv[1];

      // El metodo toCharArray() transforma un string en un
      // array de caracteres que puede ser manipulado
      char pal[ ] = palabra.toCharArray();
      int lon = pal.length;

      for (int i = 0; i < lon; i++)
      if (pal[i] == car.charAt(0)) {
        for(int j = i; j < lon-1; j++)
          pal[j] = pal[j+1];
          --lon;
          --i;
        }
```

```
      // El metodo copyValueOf transforma parcial o
      //completamente un array de caracteres en un string
      palabra = String.copyValueOf(pal,0,lon);
      System.out.println(palabra);
    } else {
      System.out.println("Este programa solo admite"
                                        +" dos parametros");
    }
  }
}
```

3. Escribir un método que dada un cadena de caracteres, calcule la suma de todos los dígitos que hay en ella. Escribir un programa que dada una cadena de caracteres por línea de ordenes invoque al método anterior e imprima la suma de los dígitos de la cadena. Use el método printf para imprimir.

   Solución:

```
class Ejercicio {

  public static void main (String [] args) {
    String cadena=args [0];
    int suma=sumadedigitos (cadena);
    System.out.printf("La suma de los digitos de la"
          +" cadena %s es %d", cadena, suma);
  }

  static int sumadedigitos(String cad){
    int sum=0;
    for (int i=0; i<cad.length(); i++){
      if((cad.charAt(i) >= '0') && (cad.charAt(i) <= '9'))
        sum=sum+(int)cad.charAt(i)-(int)'0';
    }
    return sum;
  }
}
```

4. Escribir un método que reciba como parámetro una cadena y que devuelva una nueva cadena que corresponda a la cadena inicial invertida. Escribir un programa que imprima la cadena inicial y la nueva en el método main. La cadena inicial debe leerse por línea de órdenes. Como ejemplo para probar el programa puede usar la cadena inicial casa donde la salida debe ser *asac*.

Solución:

```
class Ejercicio {
  public static void main(String [] args){
    String cadena=intercambiar (args[0]);
    System.out.println(cadena);
  }

  public static String intercambiar(String cadena){
    String nuevo="";
    for(int i = cadena.length()-1; i>=0; i--){
      nuevo += cadena.charAt(i)+"";
    }
    return nuevo;
  }
}
```

5. Diseñar e implementar en Java un programa que indique si una palabra es un palíndromo. Para ello use un método llamado palíndromo. Escribir un programa principal que primero compruebe que la cadena que se le pase al método palíndromo tiene menos de 15 caracteres. En caso de que la cadena sea mayor, el programa principal indicará que no se va a comprobar si la cadena es palíndroma. Use la clase BufferedReader para la entrada de datos y el método println para imprimir por pantalla la información. Nota: Una palabra es un palíndromo cuando se lee igual de izquierda a derecha que de derecha a izquierda. Probar: casa, ana, dabalearrozalazorraelabad.

   Solución:

   a) Análisis: La especificación está clara en el enunciado, recordar que hay que comprobar en el main que la cadena de entrada tenga menos de 15 caracteres.

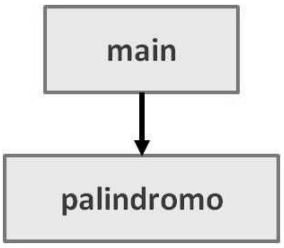
   b) Diseño: Una forma de ver si una cadena es un palíndromo sería dando la vuelta a ésta y comparando la cadena original con la cadena al revés: si son iguales se trata un palíndromo. Sin embargo este método no es el más eficiente. Por ejemplo si el primer y último carácter de la cadena ya son distintos, no es necesario seguir comparando. Por tanto, también se puede saber si una cadena es un palíndromo usando un bucle y comparando el primer carácter con el último, el segundo con el penúltimo y así sucesivamente. Para ello usamos dos índices y mientras uno se va incrementando el otro se decrementará. El proceso debe acabar cuando el índice que va aumentando es mayor o igual que el que

va decrementando. Si al salir del bucle los dos caracteres apuntados por los índices son iguales, se tratará de un palíndromo.

Vamos a hacer el programa considerando una sóla palabra introducida por teclado. Se podría introducir una frase pero siempre teniendo encuenta que los caracteres en blanco cuentan.

El programa tendrá dos métodos: método main y método palindromo (boolean palindromo (String palabra)) que devolverá true o false dependiendo de si la palabra es palíndroma o no. En el main se comprobará si la longitud de la cadena es menor de 15 caracteres, en caso de ser mayor se mostrará un mensaje de error.

El diagrama de estructura correspondiente sería:



El pseudocódigo del algoritmo necesario sería:

1: Leer palabra
2: $palin \Leftarrow Verdadero$
3: $prin \Leftarrow 0$
4: $fin \Leftarrow longitud\_cadena - 1$
5: **while** $prin < fin$ AND palin **do**
6: **if** carácter prin de palabra distinto de carácter fin de palabra **then**
7: $palin \Leftarrow Falso$
8: **else**
9: $prin \Leftarrow prin + 1$
10: $fin \Leftarrow fin - 1$
11: **end if**
12: **end while**
13: **return** palin

c) Implementación

```
import java.io.*;

class Ejercicio {

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

    System.out.print("\nIntroduzca una palabra: ");
    String cadena=leer.readLine();

    if (cadena.length()<15) { //se comprueba si es palindromo
      if (palindromo(cadena))
        System.out.println ("\nLa palabra es un palindromo");
      else
        System.out.println ("\nLa palabra no es "
                  + "un palindromo");
    } else //no se comprueba y se muestra el mensaje
      System.out.println ("\nLa palabra tiene mas"
                                        + " de 15 letras");
  }

  static boolean palindromo(String palabra){
    int prin, fin;
    boolean palin = true;
    prin = 0;
    fin = (palabra.length()) - 1;
    while ( (prin < fin ) && palin ) {
      if (palabra.charAt(prin) != palabra.charAt(fin))
        palin = false;
      else {
        prin++;
        fin--;
      }
    }
    return palin;
  }
}
```

6. Escribir un programa que contenga cuatro métodos, aparte del main. El primero de ellos, llamado suma, debe recibir dos números enteros y devolver la suma de esos dos números. Un segundo método, también llamado suma, recibirá un número como

parámetro y lo sumará a un entero que pedirá por teclado usando la clase Scanner. Dicho método devolverá la suma de los dos números. El tercer método, llamado también suma, recibirá dos cadenas y devolverá la suma del número de caracteres de las dos cadenas. Finalmente, el cuarto método se llamará suma_cadenas, recibirá tres cadenas como parámetros y devolverá una frase que sea la unión (concatenación) de esas 3 cadenas. En ninguno de estos métodos se debe imprimir nada. En el método main se imprimirán los valores resultantes de invocar a cada uno de los métodos. Como parámetros actuales al invocar a los métodos se usarán los siguientes valores:

- Primer método: 3 y 5
- Segundo método: 7
- Tercer método: programa y modular
- Cuarto método: voy , a , aprobar

Solución:

```
import java.util.*;

class Ejercicio {

  static Scanner leer= new Scanner (System.in);
  public static void main (String [] args){
    int resultado1, resultado2;
    int resultado3;
    String resultado4;
    resultado1 = suma (3,5);
    resultado2 = suma(7);
    resultado3 = suma ("programa", "modular");
    resultado4 = suma_cadenas ("voy","a","aprobar");
    System.out.println ("El resultado es: "+resultado1+ " " +
                 resultado2+ " " + resultado3+ " "+ resultado4);
  }

  public static int suma (int a, int b){
    return a+b;
  }

  public static int suma (int a){
    System.out.print("Escriba un numero: ");
    int i=leer.nextInt();
    return i+a;
  }
```

```
  public static int suma (String ca1, String ca2){
    int suma=ca1.length()+ca2.length();
    return suma;
  }
  public static String suma_cadenas (String a, String b, String c){
    return a+b+c;
  }
}
```

4.2.2. Métodos

7. Escribir un programa que contenga un método que no devuelva nada y que no tenga argumentos pero que imprima el resultado de multiplicar dos números. Escribir un método igual que el anterior pero que en lugar de mostrar el resultado de la multiplicación por pantalla, devuelva el número. Por último hacer el mismo método, pero que reciba como argumentos los 2 números, que serán de tipo double, y devuelva su multiplicación. El tipo que se devuelve debe ser int.

Solución:

```
class Ejercicio {
  static void mostrarNumero(){
    double numero1, numero2, resultado;
    numero1 = 3.5;
    numero2 = 5.6;
    resultado = numero1 * numero2;
    System.out.println("El resultado de multiplicar" +
                       numero1+"*"+numero2+"es "+resultado);

  }
 //Metodo que devuelve el resultado de la multiplicacion
  static double devolverResultado(){
    double numero1, numero2, resultado;
    numero1 = 3.5;
    numero2 = 5.6;
    resultado = numero1 * numero2;
    return resultado;
 }
//Metodo que tiene parametros y devuelve un entero
  static int metodoParametros(double numero1, double numero2){
    int resultado;
    resultado = (int) (numero1 * numero2);
    return resultado;
  }
```

```
  public static void main(String args[]){
    double resultado1;
    int resultado2;
    mostrarNumero();
    resultado1 = devolverResultado();
    resultado2 = metodoParametros(3.5, 5.6);
    System.out.println(resultado1);
    System.out.println(resultado2);
  }
}
```

8. Escribir un método que compruebe si un año es bisiesto o no. El método debe recibir como argumento un año y devolver verdadero o falso dependiendo de si el año es bisiesto o no. De acuerdo al calendario Gregoriano un año es bisiesto si es divisible por 4, excepto aquellos divisibles por 100 que para ser bisiestos deben ser también divisibles por 400. Además, imprimir un programa que dado un año imprima si es bisiesto usando el método hecho anteriormente.

   Solución:

```
class Ejercicio {
  public static void main (String[] args) {

    int v=Integer.parseInt(args[0]);
    if (bisiesto(v))
      System.out.println("El agno "+v+ " es bisiesto");
    else
      System.out.println("El agno "+v+ " no es bisiesto");
  }

  public static boolean bisiesto (int v){
    boolean valor=false;
    if (v%400==0) {
      valor=true;
    }
    else {
      if (v%4==0 && v%100!=0){
        valor=true;
      }
    }
    return valor;
  }
}
```

Otra forma de hacerlo:

```
public static boolean bisiesto (int v){
  boolean valor=false;
  valor = (v%4==0);
  valor = valor && (v % 100 != 0);
  valor = valor || (v%400 == 0);
  return valor;
}
```

9. Escribir un método que compruebe si un número pasado como parámetro es impar. Si lo es, el método devolverá el siguiente número impar. Si el número no es impar el método devolverá el número impar anterior.

   Solución:

```
public static int impar (int x){
  if (x%2!=0){
    x=x+2;
  }
  else {
    x--;
  }
  return x;
}
```

10. Escribir un método que evalúe el factorial de un número N. Además, escribir un método que lea por línea de órdenes el valor de N e imprima su factorial usando el método anteriormente escrito.

    Solución:

```
class Ejercicio {
  public static void main (String argv []) {
  int N;
  long resultado;
  N= Integer.parseInt(argv[0]);
  if (N>=0) {
    resultado=factorial(N);
    System.out.println ("El factorial de "+N+" es: "+resultado);
  } else
    System.out.println("No se puede calcular el "
                       + "factorial de numeros negativos");
  }
```

```
  static long factorial (int N) {
    int i=2;
    long producto=1;
    while (i<=N) {
      producto=producto*i;
      i=i+1;
    }
    return producto;
  }
}
```

Cuando $N = 0$ o $N = 1$ no se ejecuta el bucle y $factorial = 1$, como se quería.

11. Construir un programa con dos métodos que usen el mismo identificador (nombre). El primer método determinará el máximo de dos números pasados como parámetro. El segundo, obtendrá el máximo de tres números también pasados como parámetros. En ambos casos, los valores numéricos se introducirán por línea de órdenes.

Solución:

a) Análisis:

Se trata de un ejemplo de sobrecarga de métodos. La lectura se debe realizar por línea de órdenes.

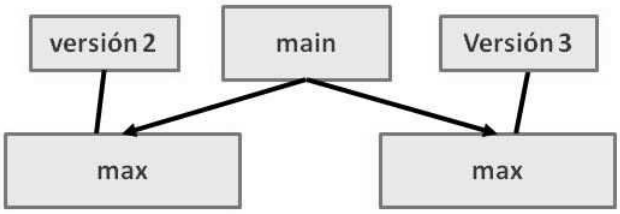
b) Diseño:

1) Estructuras de datos:
Puesto que se tienen que pasar dos o tres parámetros al método no podemos usar una matriz monodimensional. Usaremos variables independientes, dos para el primer método y tres para el segundo.

2) Pseudocódigo:
El algoritmo se ilustra con el caso más complejo, el de tres elementos:

1: $maximo \Leftarrow a$
2: **if** $b > maximo$ **then**
3: $\quad maximo \Leftarrow b$
4: **end if**
5: **if** $c > maximo$ **then**
6: $\quad maximo \Leftarrow c$
7: **end if**
8: **return** maximo

3) Diagrama de estructura:



c) Implementación

```
class Ejercicio {
  public static void main ( String [] args ) {
    double a, b, c, valor ;
    // Asignacion de los datos
    a= Double.parseDouble (args [0]);
    b= Double.parseDouble (args [1]);
    // Eco de los datos
    System.out.println (" Valor 1 : "+a);
    System.out.println (" Valor 2 : "+b);
    if (args.length == 2) {
      valor = maximo (a, b);
    } else {
      c= Double.parseDouble(args [2]);// Caso de tres valores
      System.out.println (" Valor 3 : "+c);
      valor = maximo (a,b,c);
    }
    // Salida de resultados
    System.out.println ("Maximo de los valores : "+ valor );
  }
  static double maximo ( double a, double b, double c) {
    // Version con tres parametros
    double max =a;
    if (b > max) max =b;
    if (c > max) max =c;
    return max ;
  }
  static double maximo ( double a, double b) {
    // Version con dos parametros
    double max =a;
    if (b > max) max =b;
    return max ;
  }
}
```

12. Construir un programa que escriba el triángulo de Pascal con un número determinado de filas. Dicho número se introducirá por teclado. A tal efecto, se utilizará la función combinatoria $c(n, k)$ definida como:

$$c(n,k) = \frac{n!}{(k!(n-k)!)} = \begin{pmatrix} c \\ k \end{pmatrix}$$

donde $k \leq n$.

Solución:

a) Análisis

Los valores de cada fila del triángulo son los coeficientes binómicos, resultado de elevar el binomio (A+B) a la potencia que indica el número de la fila. Por otro lado, el coeficiente $c(n, k)$ indica cuantas formas distintas hay de seleccionar k elementos de un total de n elementos (número de combinaciones de n elementos tomados de k en k). En el triángulo, las filas indican el número total de elementos, n, y las columnas el número de elementos que se seleccionan, k. Tanto filas como columnas se empiezan a contar en cero, ver diagrama a continuación.

	5	4	3	2	1	0	1	2	3	4	5
Fila 0						1					
Fila 1					1		1				
Fila 2				1		2		1			
Fila 3			1		3		3		1		
Fila 4		1		4		6		4		1	
Fila 5	1		5		10		10		5		1

Es necesario indicar el valor de n. k no es necesario, pues n lo determina: $0 \leq k \leq n$. De acuerdo a los requisitos se introducirá por línea de órdenes.

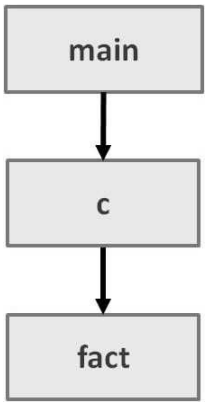
b) Diseño

Para obtener los coeficientes se usará la expresión para c(n, k). Para ello será necesario poder calcular el factorial de un entero.

1) Estructuras de datos
   Se usarán variables de tipos primitivos.
2) Algoritmo
   Usando pseudocódigo el algoritmo sería:

1: Leer número de filas
2: **for** n tal que $0 \leq i < filas$ **do**
3: **for** k tal que $0 \leq k \leq n$ **do**
4: $numero \Leftarrow c(n, k)$
5: Imprimir numero sin saltar de linea
6: **end for**
7: Imprimir línea en blanco
8: **end for**

3) Diagrama de estructura
Será necesario un módulo para calcular los coeficientes $c(n, k)$ y otro para evaluar los factoriales. El diagrama de estructura resultante se muestra a continuación.



c) Implementación

```
/*************************************************************
Programa para la obtencion del triangulo de Pascal.
El numero de filas a generar se introduce por teclado
*************************************************************/
import java.io.*;
class Ejercicio {

  public static int c(int n, int k) {
    return fact(n)/(fact(k)*fact(n-k));
  }
  public static int fact(int n) {
    int producto=1;
    for (int i=1; i<=n;i++){
      producto=producto*i;
    }
    return producto;
  }
```

```
  public static void main(String [] args ) throws IOException{
    int filas, numero;
    BufferedReader leer=new BufferedReader(new
                        InputStreamReader(System.in));
    System.out.print ("Introduza el numero de filas: ");
    filas=Integer.parseInt(leer.readLine());
    for (int n=0; n<filas; n++){
      for (int k=0; k<=n;k++){
        numero=c(n,k);
        System.out.print(numero + "  ");
      }
      System.out.println();
    }
  }
}
```

Obsérvese que la salida es un triángulo de Pascal escrito sin formato, es decir, que para las primeras cinco filas se tendría:

1
1 1
1 2 1
1 3 3 1
1 4 6 4 1

13. Escribir un programa que determine el tipo de un triángulo dada la longitud de sus tres lados. El programa deberá incluir los siguientes métodos que devuelven un valor lógico indicando el tipo del triángulo:

    - es_rectangulo (para triángulos rectángulos)
    - es_escaleno (todos los lados distintos)
    - es_isosceles (dos lados iguales y el otro distinto)
    - es_equilatero (los tres lados iguales)

Solución:

a) Análisis

La lectura de los tres lados se va a realizar por teclado. Lo demás está claro en el enunciado.

b) Diseño

La distinción de los diferentes tipos de triángulo se realiza comparando las longitudes de los lados.

1) Estructuras de datos.
No se usa ninguna estructura de datos en particular.

2) Algoritmos.
El caso del triángulo rectángulo necesita determinar si la suma de los cuadrados de los catetos es igual al cuadrado de la hipotenusa. Para distinguir todos los casos hay que distinguir si los lados son todos iguales o distintos o si hay dos iguales. Para el caso del triángulo rectángulo el pseudocódigo sería:

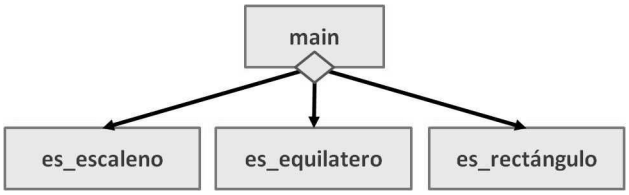
```
1: lado1_2 ⇐ lado1 * lado1
2: lado2_2 ⇐ lado2 * lado2
3: lado3_2 ⇐ lado3 * lado3
4: limite ⇐ valorsuficientemente pequeño
5: rectangulo ⇐ falso
6: if |lado1_2 − lado2_2 − lado3_3| < limite then
7:    rectangulo ⇐ verdadero
8: else
9:    if |lado2_2 − lado1_2 − lado3_3| < limite then
10:      rectangulo ⇐ verdadero
11:   else
12:      if |lado3_2 − lado2_2 − lado1_2| < limite then
13:         rectangulo ⇐ verdadero
14:      end if
15:   end if
16: end if
```

Tengamos en cuenta que si el programa sólo tiene que distinguir el tipo de triángulo podríamos mirar si es escaleno o equilátero, si no lo es ya sabríamos que es isósceles pues son posibilidades excluyentes. Aparte, miraríamos si es o no rectángulo aunque en el caso de un equilátero ya sabríamos que no puede ser rectángulo.

Pseudocódigo:

```
1: if esescaleno then
2:    Imprimir: Es escaleno
3: else
4:    if esequilatero then
5:       Imprimir: Es equilátero
6:    else
7:       Imprimir: Es isósceles
8:    end if
9: end if
```

3) Diagrama de estructura:



c) Implementación

```
/*************************************************************/
/*Este programa distingue el tipo de un triangulo cuyos tres */
/*lados se introducen como argumentos por la linea de ordenes*/
/*************************************************************/

class Ejercicio {
  static final double LIMITE=1.0e-6;  // Equivalencia a cero

  public static void main (String [] args){
    double lado1=0, lado2=0, lado3=0;

// Eco de los datos de entrada
    System.out.println("Longitud de los lados del triangulo:");
    for (int i=0; i<=args.length-1;i++)
      System.out.println ("Lado "+i+" : "+args[i]);
    System.out.println();

// Asignacion de los datos leidos
    lado1=Double.parseDouble(args[0]);
    lado2=Double.parseDouble(args[1]);
    lado3=Double.parseDouble(args[2]);
```

```
// Determinacion del tipo de triangulo
    if (es_escaleno(lado1, lado2, lado3))
      System.out.println ("El triangulo es escaleno");
    else
      if (es_equilatero(lado1, lado2, lado3))
        System.out.println ("El triangulo es equilatero");
      else
        System.out.println ("El triangulo es isosceles");
    if (es_rectangulo(lado1, lado2, lado3))
      System.out.println ("El triangulo es rectangulo");
  }

  static boolean es_escaleno (double a,double b,double c) {
    boolean escaleno=false;
    if (Math.abs (a-b) > LIMITE)
      if (Math.abs (a-c) > LIMITE)
        if (Math.abs (b-c) > LIMITE)
          escaleno=true;
    return escaleno;
  }

/*************************************************************/
/* El metodo es_isosceles se presenta por completitud pero */
/* no se usa en el programa.                                 */
/*************************************************************/

  static boolean es_isosceles (double a,double b,double c) {
    boolean isosceles=false;
    if (Math.abs(a-b) <= LIMITE && Math.abs (a-c) >LIMITE)
      isosceles=true;
    else
      if (Math.abs (a-c) <= LIMITE && Math.abs (a-b) >LIMITE)
        isosceles=true;
      else
        if (Math.abs(b-c) <= LIMITE && Math.abs (a-c) >LIMITE)
          isosceles=true;
    return isosceles;
  }

  static boolean es_equilatero (double a,double b,double c) {
    boolean equilatero=false;
    if (Math.abs(a-b)<LIMITE && Math.abs(a-c)<LIMITE)
      equilatero=true;
    return equilatero;
  }
```

```
  static boolean es_rectangulo (double a,double b,double c) {
    boolean rectangulo=false;
    a=a*a;
    b=b*b;
    c=c*c;

    if (Math.abs (a-b-c) < LIMITE)
      rectangulo=true;
    else
      if (Math.abs (b-c-a) < LIMITE)
        rectangulo=true;
      else
        if (Math.abs (c-a-b) < LIMITE)
          rectangulo=true;
    return rectangulo;
  }
}
```

14. Construir un programa que calcule los valores del exponente, e^x, el coseno, $cos(x)$, y el seno, $sen(x)$ a partir de las series siguientes:

$$e^x = \sum_{i=0}^{n} \frac{x^i}{i!}$$
$$cos(x) = \sum_{i=0}^{n} -1^i * \frac{x^{2i}}{(2i)!}$$
$$sen(x) = \sum_{i=0}^{n} -1^i * \frac{x^{2i+1}}{(2i+1)!}$$

El número de términos de la serie será el necesario para que la diferencia absoluta entre dos valores sucesivos sea menor de 10^{-3}. Úsese un método para cada caso, imprimiendo los distintos resultados en el método principal.

Solución:

```
import java.io.*;
class Ejercicio {
  public static void main (String [] args)throws IOException {
    BufferedReader leer=new BufferedReader
        (new InputStreamReader (System.in));
    System.out.print("Introduzca el valor de x: ");
    double x=Double.parseDouble(leer.readLine());
    System.out.println("e^("+x+ ") = "+e_x(x));
```

```
    System.out.println("cos("+x+")= "+coseno(x));
    System.out.println("sen("+x+")= "+seno(x));
  }

  static double e_x(double x){
    int i;
    double suma,ultimo,termino;
    suma = 0;
    termino = 0;
    i = 0;
    do {
      ultimo = termino;
      termino = Math.pow(x,i)/fact(i);
      suma = suma+termino;
      i++;
    }while(Math.abs(ultimo-termino) >= 0.001);
    return suma;
  }

  static double coseno(double x){
    int i;
    double suma, ultimo,termino;
    i = 0;
    suma = 0;
    termino = 0;
    do {
      ultimo = termino;
      termino = Math.pow(x,2*i)/fact(2*i);
      if (i%2 == 0)
        suma = suma+termino;
      else
        suma = suma-termino;
        i++;
    }while(Math.abs( termino-ultimo ) >= 0.001);
    return suma;
  }

  static double seno(double x){
    int i;
    double suma, ultimo,termino;
    i = 0;
    suma = 0;
    termino = 0;
    do {
      ultimo = termino;
      termino = Math.pow(x,2*i+1)/fact(2*i+1);
```

```
      if (i%2 == 0)
        suma = suma+termino;
      else
        suma = suma-termino;
        i++;
    }while(Math.abs(termino-ultimo) >= 0.001);
    return suma;
  }

  static long fact (int N) {
    if (N == 0)
      return 1;
    else
      return fact (N-1)*N;
  }
}
```

Resultados:

$e^{0,0010} = 1{,}0010005$
$cos(0{,}0010) = 0{,}9999995000000417$
$sen(0{,}0010) = 9{,}999998333333334E-4$

$e^{0,01} = 1{,}0100501666666668$
$cos(0{,}01) = 0{,}9999500004166667$
$sen(0{,}01) = 0{,}009999833334166666$

15. Escribir un programa en Java que permite generar números aleatorios, para simular el lanzamiento de un dado. Para la generación de números aleatorios utilice la clase Random que provee Java.

    Este ejemplo se realiza usando números aleatorios (pseudoaleatorios). En Java existe una clase para esta tarea, es la clase Random (aleatorio). La clase Random está definida en el paquete java.util e implementa un generador de números pseudoaleatorios. Los números aleatorios se usan mucho en juegos y en simulación. El generador de números aleatorios selecciona un valor entre un intervalo de valores. Por ejemplo, tenemos varios métodos de utilidad como son nextInt() que devuelve un entero (int) entre el intervalo de valores del tipo int. Tenemos también un nextFloat() y un nextDouble() que devuelven un real de tipo float o double seleccionado entre el intervalo $0{,}0 - 1{,}0$. Para simular la obtención de un valor entero de una serie de valores se puede usar el nextInt(). Lo que hay que hacer es 'escalar' (cambiar la escala) del

intervalo de valores del tipo int al intervalo que nos interesa. Esto se puede hacer con el operador % dividiendo por el número de valores que se desean. Como al dividir por n los restos posibles son $0, 1, ..., (n-1)$ ya tenemos la escala. Si en lugar de 0 a (n-1) queremos que vaya de 1 a n sumamos 1.

Expresión	Intervalo
Math.abs(nombreObjeto.nextInt()) %6+1	1 a 6
Math.abs(nombreObjeto.nextInt()) %10+1	1 a 10
Math.abs(nombreObjeto.nextInt()) %101	0 a 100
Math.abs(nombreObjeto.nextInt()) %11+20	20 a 30
Math.abs(nombreObjeto.nextInt()) %11-5	-5 a 5

Tabla 4.1: Ejemplos

Para lanzar un dado, se hará el resto con 6 y se sumará 1.

Solución:

```
/*-----------------------------------------------------------------*/
/* Programa que simula el lanzamiento de un dado usando la clase */
/* Random                                                          */
/*-----------------------------------------------------------------*/
import java.io.*;
import java.util.*;
class Ejercicio {
  public static void main (String [] args) throws IOException {
    int N, tirada;

    Random dado = new Random();// Creando el objeto de clase Random
    BufferedReader leer = new BufferedReader
                          (new InputStreamReader(System.in));

    System.out.println ("Numero de tiradas:");
    N=Integer.parseInt(leer.readLine());

    for (int i=1; i<=N; i++) {
      tirada = Math.abs(dado.nextInt())%6 + 1;
      System.out.println ("Tirada "+i+" : "+tirada);
    }
  }
}
```

16. Un número se denomina mágico cuando es divisible entre 3 y 5 y no es divisible entre 10. Escribir un método que reciba como parámetros dos números e imprima los números mágicos comprendidos entre esos dos números.

    Solución:

```
public static void magico (int a, int b){
   for (int i=a; i<=b; i++){
     if (i%3==0 && i%5==0 && i%10!=0) {
       System.out.println("el numero: "+i);
    }
  }
}
```

17. Sean a y b dos números enteros positivos o cero. Escribir un método que calcule la potencia a^b sin usar el método pow de Java. Escribir un método main que dados dos números a y b leídos por línea de órdenes calcule la potencia a^b usando el método escrito anteriormente. Sólo se calculará la potencia si tanto a como b son múltiplos de 2.

    Solución:

```
class Ejercicio {

  public static void main(String args[]) {

    int x = new Integer(args[0]).intValue();
    int y = new Integer(args[1]).intValue();

    if (x%2==0 && y%2==0){

      System.out.print(x + "^" + y + " :\n");
      long f = potenciaf(x,y);
      long w = potenciaw(x,y);
      double ver = Math.pow(x,y);
      System.out.println("\nBucle for: "+f);
      System.out.println("Bucle while: "+w);
      System.out.println("Metodo pow de Java: "+ver);
    }
    else {
      System.out.println("Los numeros x e y no son multiplos de 2");
    }
  }
```

```
// Metodo potencia con bucle while
  static long potenciaw(int x, int y) {
    long potencia=1;
    int contador=0;
    while (contador!=y){
      potencia=potencia*x;
      contador++;
    }
    return potencia;
  }

//Metodo potencia con bucle for
  static long potenciaf(int x, int y) {
    long potencia=1;

    for (int j=0;j<y;j++){
      potencia=potencia*x;

    }
    return potencia;
  }
}
```

18. Construir un método que devuelva el producto de los n primeros números enteros pares a partir de un número x dado. Tanto x como n se pasarán como parámetros al método. Suponer n mayor o igual a 1.

    Ejemplo: Siendo $n = 3$; $x = 15$, el resultado sería $16 * 18 * 20 = 5760$

    Solución:

```
class Ejercicio {

  public static long metodo(int n, int x){
    long resultado=1;
    while(n>0){
      if(x %2 == 0){
        resultado *= x;
        n--;
      }
      x++;
    }
    return resultado;
  }
```

```
  public static void main (String [] args){
    int n=Integer.parseInt(args[0]);
    int x=Integer.parseInt(args[1]);
    if (n<1){
      System.out.println("Error");
    }
    else {
      System.out.println(metodo(n,x));
    }
  }
}
```

19. Realizar un programa que calcule el perímetro y el área de un círculo. Después de mostrar el resultado, el programa debe preguntar si se quiere calcular el perímetro y área de otro círculo. Utilice la clase Scanner para leer el radio del círculo y el método printf para mostrar la salida del programa por pantalla.

    Solución:

```
import java.util.*;

class Ejercicio {

  public static void main(String args[]) {

    double area, perimetro, radio;
    boolean seguir;
    Scanner sc=new Scanner(System.in);
    sc.useLocale(Locale.US);

    do {
      System.out.printf("Introduzca el radio del circulo:\n");
      radio = sc.nextDouble();
      area = area(radio);
      perimetro = perimetro(radio);
      System.out.printf(Locale.US, "El area es: %.4f\n
      El perimetro es: %.4f\n",area, perimetro);
      System.out.printf("Introduzca la palabra true si"
        + "quiere calcular otros valores, en caso contrario"
                          + "introduzca la palabra false.\n");
      seguir = sc.nextBoolean();
    } while (seguir);

  }
```

```
  // metodo que devuelve el area de un circulo
  static double area(double radio) {
    double resultado;
    resultado = Math.PI * Math.pow(radio, 2);
    return resultado;
  }
  // metodo que devuelve el perimetro de un circulo
  static double perimetro(double radio) {
    double resultado;
    resultado = 2 * Math.PI * radio;
    return resultado;
  }
}
```

20. Un club de baloncesto saca a la venta las entradas para el próximo partido. El precio de las entradas varía dependiendo de la zona del pabellón que el espectador quiere ocupar.

    - El precio de una entrada para la zona de los fondos es de 5 euros. Si un aficionado adquiere más de 7 entradas para esta zona obtiene un descuento total del 6 %.
    - El precio de una entrada para la zona central es de 12 euros. Si un aficionado adquiere más de 5 entradas para esta zona obtiene un descuento total del 5 %.
    - El precio de una entrada para la zona VIP es de 20 euros. Si un aficionado adquiere más de 9 entradas para esta zona obtiene un descuento total del 4 %.

    Diseñar una aplicación (usando las características de la programación estructurada y modular) que muestre un menú con las tres zonas del pabellón, el usuario deberá escoger la zona del pabellón para la que desea adquirir entradas. A continuación la aplicación pedirá al usuario que introduzca el número de entradas que quiere adquirir. Posteriormente el programa mostrará por pantalla el importe en euros que el aficionado debe abonar. Utilice la clase Scanner para leer los datos por teclado y el método printf para mostrar la salida del programa por pantalla.

    Solución:

```
import java.util.Scanner;
import java.util.Locale;
class Ejercicio {
  private static Scanner sc = new Scanner(System.in);
  public static void main(String[] args) {
    int entradas, zona;
```

```
    double precio;
    zona = leerZona();
    entradas = leerEntradas();
    precio=calcularPrecio(zona,entradas);
    mostrarDinero(precio);
  }

  public static int leerZona() {
    int opcion;
    do {
      System.out.printf
        ("Introduzca la zona del pabellon que desea ocupar:\n");
      System.out.printf("1.- Fondos\n");
      System.out.printf("2.- Zona central\n");
      System.out.printf("3.- Zona vip\n");
      opcion = sc.nextInt();
    } while (opcion < 1 || opcion> 3);
    //Solo repetimos el bucle si hay opcion incorrecta
    return opcion;
  }

  public static int leerEntradas() {
    int nentradas;
    System.out.printf("Cuantas entradas quiere adquirir?\n");
    nentradas = sc.nextInt();
    return nentradas;
  }

  public static double  calcularPrecio(int zona, int entradas){
    double dinero=0.0;
    switch (zona) {
    case 1:
      dinero=calculaFondo(entradas);
      break;
    case 2:
      dinero=calculaCentral(entradas);
      break;
    case 3:
      dinero=calculaVip(entradas);
      break;
    }
    //No usamos default porque al leer la zona
    //obligamos a introducir una numero entre 1 y 3

    return dinero;
  }
```

```
  public static double calculaFondo(int nentradas) {
    double precioTotal = 5 * nentradas;
    if (nentradas > 7)
      precioTotal *= 0.94;
    return precioTotal;
  }

  public static double calculaCentral(int nentradas) {
    double precioTotal = 12 * nentradas;
    if (nentradas > 5)
      precioTotal *= 0.95;
    return precioTotal;
  }
  public static double calculaVip(int nentradas) {
    double precioTotal = 20 * nentradas;
    if (nentradas > 9)
      precioTotal *= 0.96;
    return precioTotal;
  }

  public static void mostrarDinero(double dinero){
    System.out.printf(Locale.US,"El importe que debe"
      + " pagar es %.2f euros.\n", dinero);
  }
}
```

21. Utilizando las propiedades y características de la programación estructurada y modular, diseñar un programa que simule el funcionamiento de una sencilla calculadora con un conjunto básico de operaciones. El programa consistirá básicamente, en la presentación de un menú al usuario en el que se le indicarán las operaciones de las que dispone. Estas serán: suma, resta, multiplicación, división y potencia de 2 operandos, que deben ser números reales. Además, existirá una opción dentro del menú para finalizar el programa. Si la opción que introduce el usuario no es la de finalización, el programa le solicitará dos datos necesarios para realizar la operación elegida. El programa debe controlar la introducción errónea de datos, mostrando un mensaje de error cuando esto se produzca y ofreciendo al usuario la posibilidad de que vuelva a introducir los datos correctamente. Utilícese la clase Scanner para leer los datos por teclado y el método printf para mostrar la salida del programa por pantalla.

Solución:

```
import java.util.*;

class Ejercicio {

  static Scanner teclado= new Scanner(System.in);

  public static void main(String args[]) {
    boolean acabar=true;
    System.out.printf("Programa que simula el funcionamiento de "
                                              + "una calculadora\n");
    do{
      mostrarmenu();
      int opcion=leeropcion();
      if (opcion!=6){
        double res=procesar(opcion);
        mostrarresultado (res,opcion);
      } else {
        System.out.printf("Fin del programa\n");
        acabar=false;
      }
    }while (acabar);
  }
  static void mostrarmenu(){
    System.out.println("1. Sumar dos numeros");
    System.out.println("2. Restar dos numeros");
    System.out.println("3. Multiplicar dos numeros");
    System.out.println("4. Dividir dos numeros");
    System.out.println("5. Calcular potencia");
    System.out.println("6. Acabar calculadora");
  }
  static int leeropcion() {
    System.out.printf("\nIntroduce la opcion deseada...\n");
    int entrada=teclado.nextInt();
    while (entrada<1 || entrada>6){
      System.out.printf("Opcion incorrecta. Vuelva a intentarlo");
      entrada=teclado.nextInt();
    }
    return entrada;
  }
  static double procesar(int o) {
    double primero;
    double segundo;
    double resultado=0;
```

```
  primero = leer_argumento(1);
  segundo = leer_argumento(2);

  switch(o){
    case 1:
      resultado=sumar(primero,segundo);
      break;
    case 2:
      resultado=restar(primero,segundo);
      break;
    case 3:
      resultado=multiplicar(primero,segundo);
      break;
    case 4:
      resultado=dividir(primero,segundo);
      break;
    case 5:
      resultado=potencia(primero,segundo);
      break;
  }
  return resultado;

}//fin procesar

static double leer_argumento (int n) {
  double argumento;
  System.out.printf ("Introduzca el argumento %d: ",n);
  argumento=teclado.nextDouble();
  return argumento;
}

static double sumar(double a1,double a2){
  return a1 + a2;
}

static double restar(double a1, double a2) {
  return a1 - a2;
}

static double multiplicar(double a1, double a2) {
  return a1 * a2;
}// fin multiplicar

static double dividir(double a1, double a2) {
  return a1 / a2;
}
```

```
  static double potencia(double a1, double a2) {
    return Math.pow(a1, a2);
  }

  static void mostrarresultado(double r, int o) {
    System.out.printf("\nEl resultado de ");
    switch (o) {
      case 1:
        System.out.printf("la suma");
        break;
      case 2:
        System.out.printf("la resta");
        break;
      case 3:
        System.out.printf("la multiplicacion");
        break;
      case 4:
        System.out.printf("la division");
        break;
      case 5:
        System.out.printf("la potencia");
        break;
    }

    System.out.printf(" es ");

    if (o != 4) {
      System.out.printf(Locale.US, "%.3f", r);
    } else {
      Double rr = new Double(r);
      // Para comprobar que el resultado de la division es
      // infinito e imprimirlo
      if (rr.isInfinite()) {
        System.out.printf("infinito porque ha dividido por cero\n");
      } else {
        System.out.printf(Locale.US, "%.3f", r);
      }
    }
    System.out.printf
       ("\n\n\t\t Introduzca yes y pulse INTRO para continuar\n");
    teclado.next("yes");
  }

}
```

22. Construir un programa que obtenga el término n de la serie de Fibonacci. El valor de n deberá leerse por teclado usando la clase Scanner. La serie de Fibonacci es una secuencia de enteros positivos, cada uno de los cuales es la suma de los dos anteriores. Los dos primeros números de la secuencia son 0 y 1, La serie se define como:

$$\begin{array}{ll} Fibonacci(n) = n & \text{para } n \leq 1 \\ Fibonacc(n) = Fibonacci(n-1) + Fibonacci(n-2) & \text{para } n > 1 \end{array}$$

El programa tendrá, además del método main, un método iterativo para calcular el término de la serie. El método main deberá llamar al método e imprimir, usando printf, el término n de la serie. Si el usuario introduce un valor negativo, el programa debe emitir un aviso.

Solución:

a) Análisis

La serie de Fibonacci es: $0, 1, 1, 2, 3, 5, 8, 13, 21, 34, \ldots$ comienza con 0 y 1 y tiene la propiedad de que cada número de Fibonacci subsecuente es la suma de los dos números de Fibonacci previos. El análisis está claro. Se introduce un número n y el programa nos debe devolver el término n de la serie

b) Diseño

Debemos crear un método que acepta como entrada una variable entera (n) y devuelve otra variable entera que será el elemento n de la serie. Los números de Fibonacci tienden a crecer rápidamente; por tanto, debemos escoger como tipo de dato para el elemento de la serie un entero de tipo long. Por ejemplo, el término 15 vale 610.

El pseudocódigo correspondiente a los algoritmos necesarios para resolver el problema es el siguiente:

- Algoritmo para el programa principal:

```
1: Leer n
2: if  n ≥ 0 then
3:     Calcular termino n de la serie
4:     Imprimir termino de la serie
5: else
6:     Imprimir: n no puede ser negativo
7: end if
```

- Algoritmo para el método (cálculo del término n de la serie)
  La definición de la serie nos lleva a sumar los elementos mediante un bucle que debe ejecutarse desde 2 hasta n. Cada iteración debe guardar el último y el penúltimo término, para lo que se utilizan dos variables *ultimo* y *penultimo*, que irán cambiando sus valores.

```
1: Leer n
2: if  n > 1 then
3:    suma ⇐ 0
4:    ultimo ⇐ 1
5:    penultimo ⇐ 0
6:    for all i tal que 2 ≤ i ≤ n do
7:       suma ⇐ ultimo + penultimo
8:       penultimo ⇐ ultimo
9:       ultimo ⇐ suma
10:   end for
11: else
12:   suma ⇐ n
13: end if
14: return  suma
```

c) Implementación

```
import java.util.Scanner;
class Ejercicio {
  public static void main (String [] args) {
    int n;
    long termino;
    Scanner leer=new Scanner(System.in);
    System.out.print("Introduzca un numero positivo: ");
    n=leer.nextInt() ;
    if (n>=0) {
      termino=fibonacci_iter(n);
      System.out.printf("\nEl termino %d de Fibonacci es %d",
                n,termino);
    }
    else {
      System.out.println("\nEl numero es negativo."
                                + " El programa terminara");
    }
  }
```

```
  static long fibonacci_iter (int n){
    long suma;
    if (n>1) {
      long ultimo,penultimo;
      suma=0;
      ultimo=1;
      penultimo=0;
      for (int i=2;i<=n;i++) {
        suma=ultimo+penultimo;
        penultimo=ultimo;
        ultimo=suma;
      }
    } else {
      suma=n;
    }
    return suma;
  }
}
```

23. Escribir un programa que dados dos enteros introducidos por teclado, evalúe su máximo común divisor usando el algoritmo de Euclides implementado en un método. El algoritmo tal y como lo propuso Euclides en el libro séptimo de los Elementos es el siguiente:

    a) Tómese el resto del cociente m/n

    b) Si el resto es cero, entonces n es el máximo común divisor

    c) Si el resto es distinto de cero se hace m=n y n=resto

    d) Se vuelve al punto primero

Solución:

a) Análisis y diseño

En este ejemplo tan sencillo tanto el análisis como el diseño están prácticamente implícitos en el enunciado. Como labor de diseño únicamente se mostrará el algoritmo de Euclides en pseudocódigo:

1: Leer n,m
2: $resto \Leftarrow 0$
3: $sigue \Leftarrow verdadero$
4: **if** $m < n$ **then**

5: $aux \Leftarrow m$
6: $m \Leftarrow n$
7: $n \Leftarrow aux$
8: **end if**
9: **while** $sigue = verdadero$ **do**
10: $resto \Leftarrow resto(m/n)$
11: **if** resto $= 0$ **then**
12: $m \Leftarrow n$
13: $n \Leftarrow resto$
14: **else**
15: $resto \Leftarrow n$
16: $sigue \Leftarrow falso$
17: **end if**
18: **end while**
19: **return** resto

b) Implementación iterativa

```
import java.io.*;
class Ejercicio {
  public static void main(String [] args) throws IOException {
    int m, n, mcd;

    BufferedReader leer =new BufferedReader
                 (new InputStreamReader(System.in));

    System.out.println("Introduzca primer numero:");
    m=Integer.parseInt(leer.readLine());

    System.out.println("Introduzca segundo numero:");
    n=Integer.parseInt(leer.readLine());
    System.out.println();
    if (m<n) {    // Ordenando los valores
      int aux=m;
      m=n;
      n=aux;
    }
    mcd=euclides_iter(n,m);
    System.out.println("El maximo comun divisor es: " + mcd);
  }
```

```
// Algoritmo de Euclides iterativo para obtener el maximo comun
// divisor de dos numeros enteros

  public static int euclides_iter(int n, int m) {
    int resto=0;
    boolean sigue=true;

    while (sigue) {
      resto= m%n;     // Determinando el resto
      if (resto!=0) {
        m=n;
        n=resto;
      }
      else {
        resto=n;
        sigue=false;
      }
    }
    return resto;
  }
}
```

24. Dos números son amigos, si cada uno de ellos es igual a la suma de los divisores del otro. Por ejemplo, 220 y 284 son amigos, ya que:

    - Suma de divisores de 284: $1 + 2 + 4 + 71 + 142 = 220$
    - Suma de divisores de 220: $1 + 2 + 4 + 5 + 10 + 11 + 20 + 22 + 44 + 55 + 110 = 284$

    Construir un método que determine si dos números dados como parámetros son amigos o no. A continuación realizar un programa que muestre todas las parejas de números amigos menores o iguales que n, siendo n un número introducido por teclado. El programa debe usar el método amigo previamente definido.

    Solución:

    a) Análisis

    Averiguar qué son dos números amigos. Hay que hacer un programa que lea un número n y que diga los amigos desde 1 hasta n.

b) Diseño

1) Pseudocódigo de los métodos necesarios.

- Método calcular_suma que evaluará los divisores de un número pasado como argumento y su suma y devolverá dicha suma al método invocante.

```
1: suma ⇐ 1
2: for all i tal que 2 ≤ i ≤ n/2 do
3:    if modulo(n/i) = 0 then
4:       suma ⇐ suma + i
5:    end if
6: end for
7: return suma
```

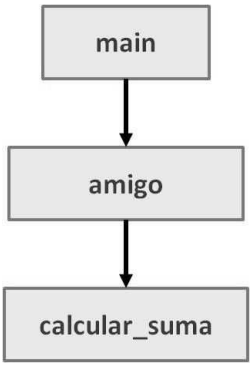
- Método amigo que determinará si dos números dados como argumentos son amigos o no devolviendo un bolean al método invocante. El método comparará la suma de los divisores de uno de los números con el otro número y viceversa. Si la suma de los divisores de los dos números es la misma, el método nos devolverá true. Este método llamará al método calcular_suma que evaluará los divisores de un número y calculará su suma.

```
1: suma1 ⇐ calcular_suma(n1)
2: if suma1 ≠ n2 then
3:    opcion ⇐ falso
4: else
5:    suma2 ⇐ calcular_suma(n2)
6:    opcion ⇐ (suma2 = n1)
7: end if
8: return opcion
```

- Método principal (main) que llamará al método amigo para comprobar que todos los números entre 1 y n son amigos. Para ello se incluye la llamada al método dentro de dos bucles for anidados que generarón todos los números comprendidos entre 1 y n. El bucle interno generará todos los números comprendidos entre 1 a n para cada n gestionado por el bucle externo. Para que no se repitan las parejas, es decir, para no hacer la comprobación 2 veces, por ejemplo 23 y 25 es lo mismo que 25 y 23, hemos hecho que el bucle interno sólo vaya desde n2=n1 hasta n quedándonos con el triángulo superior de la diagonal.

1: Leer n
2: **for all** $n1$ tal que $1 \leq n1 \leq n$ **do**
3: **for all** $n2$ tal que $n1 \leq n2 \leq n$ **do**
4: **if** $amigo(n1, n2) = verdadero$ **then**
5: Imprimir: n1 y n2 son amigos
6: **end if**
7: **end for**
8: **end for**

2) Diagrama de estructura



c) Implementación

```
import java.io.*;

class Ejercicio {
  public static void main(String [] args) throws IOException {
    int n,n1,n2;
    BufferedReader lee=new BufferedReader
                  (new InputStreamReader (System.in));
    System.out.print("Introduzca un numero: ");
    n=Integer.parseInt(lee.readLine());
    for(n1=1;n1<=n;n1++) {
      for(n2=n1; n2<=n; n2++) { //solo recorre la diagonal por
          // (n1,n2)=(n2,n1)
        if(amigo(n1,n2)) {
          System.out.println("\nLos numeros "+ n1 + " y " +
                                          n2 +" son amigos");
        }
      }
    }
  }
```

```
  public static boolean amigo(int n1, int n2) {
    boolean opcion;
    int suman1, suman2=0;
    suman1 = calcular_suma (n1);
    if (suman1!=n2){
      opcion=false;
    } else {
      suman2=calcular_suma(n2);
      opcion=(suman2==n1);
    }
    return opcion;
  }
  public static int calcular_suma (int n){
    int suma;
    suma=1;
    for(int i=2;i<=n/2;i++) {
      if(n%i==0){
        suma=suma+i;
      }
    }
    return suma;
  }
}
```

25. Los factores primos de un número entero son los números primos divisores exactos de ese número entero cuyo producto es igual al número original. Los factores primos no incluyen el 1 pero si cada copia de cada número primo. Por ejemplo, los factores primos de 90 son 2 3 3 5 y los de 1092 son 2 2 3 7 y 13. Para encontrar los factores primos de un número el procedimiento a seguir es:

    - Comprobar si el número es divisible por el menor número primo posible (considerar 2). Si es divisible, este número primo es el primer factor primo.
    - Dividir el número por el primer factor primo y comprobar si el resultado es divisible por dicho factor. Si es así, tendríamos el segundo factor primo (que sería repetido). Esta operación se repite hasta que el resultado no pueda dividirse por ese factor primo.
    - Cuando el resultado no pueda volverse a dividir por ese número, buscar el siguiente número primo posible para continuar dividiendo, repitiendo los pasos anteriores. Desde el punto de vista algorítmico es suficiente con ir probando los siguientes números enteros.
    - Cuando el cociente entre el resultado y el factor primo sea 1 acabar.

Construya un método que reciba como argumento un número y devuelva la suma de los factores primos de ese número, imprimiéndolos en el método main.

Solución:

```
import java.util.*;
class Ejercicio {
  public static void main (String [] args) {
    Scanner leer=new Scanner(System.in);
    int suma;
    System.out.println("Introduce numero");
    int numero=leer.nextInt();
    suma=fac_prim_dowhile(numero);
    System.out.printf("suma=%d \n",suma);
    suma=fac_prim_for(numero);
    System.out.printf("suma=%d\n",suma);
  }
  public static int fac_prim_dowhile (int numero){
    int i;
    int suma=0;
    i=2;
    do {     //tambien se puede hacer con un while
      if (numero%i==0){
        System.out.print(i+" ");
        suma=suma+i;
        numero=numero/i;
      } else
        i++;
    } while (i<=numero);
    return suma;
  }
  public static int fac_prim_for (int n){
    int suma=0;
    for( int i = 2; i <= n ; ){
      if (n % i == 0){
        System.out.print(i+" ") ;
        suma=suma+i;
        n = n / i ;
      } else {
        i++;
      }
    }
    return suma;
  }
}
```

4.2.3. Matrices

26. Escribir un programa que cree una matriz con los caracteres de la 'a' a la 'j' y a continuación sustituya todas las vocales que se encuentren en la matriz por el carácter @.

Solución:

```
class Ejercicio {
  public static void main (String [] args){
    char matrizvocales []={'a','b','c','d','e','f','g','h','i','j'};
    System.out.print("Matriz de vocales antes del cambio: ");
    for (int z=0; z<matrizvocales.length;z++){
      System.out.print(matrizvocales[z]);
    }
    //Recorremos la matriz desde 0 hasta el tamanno de la matriz-1
    for (int z=0; z<matrizvocales.length;z++){
      switch (matrizvocales[z]){
        case 'a':
        case 'e':
        case 'i':
        matrizvocales[z]='@';
      }
    }
    //Imprimimos la matriz cambiada
    System.out.print("\n\nMatriz de vocales despues del cambio: ");

    for (int z=0; z<matrizvocales.length;z++){
      System.out.print(matrizvocales[z]);
    }
    System.out.println();
  }
}
```

27. Construir un programa que obtenga la matriz suma de dos matrices: A (con dimensiones M y N) y B (con dimensiones P y Q).

Solución:

a) Análisis:

La especificación está clara en el enunciado, sólo algunos comentarios respecto a la entrada y salida.

- Datos de salida: suma (matriz suma)

- Datos de entrada: a,b (matrices a sumar), m,n (dimensiones de la matriz a), p,q (dimensiones de la matriz b)

b) Diseño:

Para realizar la suma de dos matrices es necesario que ambas tengan las mismas dimensiones, por lo que lo primero que debemos hacer con los parámetros m,n,p y q es comprobar que m es igual a p y n es igual a q.

En la suma de matrices, cada elemento suma (i,j) es igual a $a(i,j) + b(i,j)$ y se debe, por tanto, recorrer las matrices con dos bucles anidados para hallar la matriz suma.

Diseño del algoritmo:

1: Declarar entero i,j,m,n,p,q
2: matriz suma,a,b
3: Leer m,n,p,q,a,b
4: **if** $m \neq p$ OR $n \neq q$ **then**
5: Imprimir: No se pueden sumar las matrices
6: **else**
7: **for all** i such that $0 \leq i \leq m-1$ **do**
8: **for all** j such that $0 \leq j \leq p-1$ **do**
9: $suma(i,j) \Leftarrow a(i,j) + b(i,j)$
10: **end for**
11: **end for**
12: Imprimir suma(i,j)
13: **end if**

c) Implementación

```
//Suma de dos matrices a(m,n) y b(p,q) para dar suma (m,n)
import java.io.*;
class Ejercicio {
  public static void main (String [] args) throws IOException {
    //Declaraciones
    BufferedReader leer = new BufferedReader
      (new InputStreamReader(System.in));
    int i,j,m,n,p,q;
    //Introduccion de datos
    System.out.println("Suma de dos matrices: a(m,n)+ b(p,q)");
    System.out.println("Recuerde que las dos matrices " +
                       "deben tener las mismas dimensiones");
    System.out.println(
                  "Introduzca la dimension m de la  matriz a");
```

```
    m=Integer.parseInt(leer.readLine());
    System.out.println(
                  "Introduzca la dimension n de la matriz a");
    n=Integer.parseInt(leer.readLine());
    System.out.println(
                  "Introduzca la dimension p de la matriz b");
    p=Integer.parseInt(leer.readLine());
    System.out.println(
                  "Introduzca la dimension q de la matriz b");
    q=Integer.parseInt(leer.readLine());
    if (m!=p || n!= q)
      System.out.println ("No se pueden sumar matrices con "
                                        +"esas dimensiones");
    else {
      int[][] suma=new int [m][n];
      int[][] a=new int [m][n];
      int[][] b=new int [m][n];
      System.out.println("Introduzca la matriz a");
      for (i=0; i<m;i++){
        for (j=0; j<n;j++){
          System.out.print("a("+i+","+j+")=");
          a[i][j]=Integer.parseInt(leer.readLine());
        }
        System.out.println();
      }
      System.out.println("Introduzca la matriz b");
      for (i=0; i<m;i++) {
        for (j=0; j<n;j++) {
          System.out.print("b("+i+","+j+")=");
          b[i][j]=Integer.parseInt(leer.readLine());
        }
        System.out.println();
      }
      //Impresion de las matrices a y b a sumar
      System.out.println ("\nMatriz a\tMatriz b\n");
      for (i=0; i<m;i++) {
        for (j=0; j<n;j++)
          System.out.print(a[i][j]+" ");
          System.out.print("\t\t");
          for (j=0;j<n;j++)
            System.out.print(b[i][j]+" ");
          System.out.println();
      }

      //Suma e impresion de la matriz suma
      System.out.println ("\nMatriz Suma\n");
```

```
      for (i=0; i<m;i++) {
        for (j=0; j<n;j++) {
          suma[i][j]=a[i][j]+b[i][j];
          System.out.print(suma[i][j]+" ");
        }
        System.out.println();
      }
    }
  }
```

28. Realizar un programa que dada una secuencia de enteros separados por espacios en blanco e introducidos por línea de órdenes, visualice por pantalla la subsecuencia creciente de mayor tamaño. Si hay varias secuencias del mismo tamaño se tomará la situada más a la izquierda.

Solución:

```
class Ejercicio {
  public static void main (String argv[]) {
    int longitud=1,longitudmayor=1,fin=0,finmayor=0;
    int numparam = argv.length;
    int [] serie = new int[numparam];
    for (int i=0; i < numparam; i++)
      serie[i] = Integer.parseInt(argv[i]);
    for (int i=0; i < numparam-1; i++) {
      if (serie[i+1] > serie[i]) {
        longitud++;
        if ((i+1)==(numparam-1) && (longitud > longitudmayor)) {
          longitudmayor=longitud;
          finmayor=i+1;
        }
      } else {
        if (longitud > longitudmayor) {
          longitudmayor=longitud;
          finmayor=fin;
        }
        longitud = 1;
      }
      fin=i+1;
    }
    for (int i = (finmayor-longitudmayor+1); i <= finmayor; i++)
      System.out.print(serie[i] + " ");
   }
}
```

4.2.4. Métodos y matrices

29. Escribir un método que reciba como parámetro un vector de enteros A y devuelva un nuevo vector B que sea igual al vector A pero desplazado hacia la izquierda, es decir, todos sus elementos han sido desplazados una posición hacia la izquierda. Ejemplo:

$$A = \{1, 2, 3, 4\}$$
$$B = \{2, 3, 4, 1\}$$

Nota: Téngase en cuenta que el primer elemento del vector A al ser desplazado hacia la izquierda pasa a ser el último elemento del vector B.

Solución:

```
public static int[] rotar (int[] vector1) {
  int [] rotado = new int [vector1.length];
  for (int i = 1; i < vector1.length; i++ ) {
    rotado [i] =  vector1[i-1];
  }
  rotado [0] = vector1[rotado.length - 1];
  return rotado;
}
```

30. Escribir un método que reciba dos vectores A y B de enteros del mismo tamaño y devuelva la multiplicación inversa de ambas matrices, es decir, multiplique el primero elemento de A por el último de B, el segundo elemento de A por el penúltimo de B, y así sucesivamente.

Ejemplo:

$$A = \{1, 2, 3\}$$
$$B = \{6, 7, 8\}$$

El resultado sería: $1 * 8 + 2 * 7 + 3 * 6$

Solución:

```
public static int[] multiplicar (int [] vect1, int [] vect2 ) {
  int [] resultado = new int[vect1.length];
  for (int i = 0; i < vect1.length ; i++) {
    resultado[i] = vect1[i] * vect2[vect2.length -1-i];
  }
  return resultado;
}
```

31. Escribir un método que acepte una matriz monodimensional de enteros y devuelva la suma de los valores almacenados en dicha matriz.

Solución:

```
public static int sum (int [] valores) {
  int suma = 0;
  for (int indice=0; indice < valores.length; indice++)
    suma=suma +valores [indice];
  return suma;
}
```

32. Escribir un método que acepte una matriz monodimensional de enteros e imprima el número mayor, el menor y el número de elementos de la matriz.

Solución:

```
public static void estadistica (int[] valores) {
  int mayor= valores [0];
  int menor=valores [0];

  for (int indice=1; indice <valores.length; indice++) {
    if (valores[indice]>mayor)
      mayor=valores[indice];
    if (valores [indice] < menor)
      menor=valores[indice];
  }

  System.out.println ("Mayor: " + mayor);
  System.out.println ("Menor: " + menor);
  System.out.println ("Numero: " + valores.length);
}
```

33. Construir un programa que cree dos matrices bidimensionales, una de tipo String y otra de tipo int. Cada matriz debe ser de $3x2$. El programa debe leer por teclado los elementos de ambas matrices. Después, deberá imprimir ambas matrices. A continuación deberá sumar un número, que se introducirá por teclado, a cada elemento de la matriz de enteros y concatenar una palabra, que también se introducirá por teclado, a los elementos de la matriz cadena. Finalmente, se volverá a imprimir cada matriz. Deberá utilizarse un método para cada una de las tareas que realiza el programa.

Solución:

a) Análisis:

El enunciado nos indica qué debemos hacer, inicializar dos matrices, imprimir sus valores, modificarlos y volverlos a imprimir.

b) Diseño:

Sabemos por el enunciado que las estructuras que necesitamos son 2 matrices. Debemos recordar dividir el problema en funcionalidades para tener un programa bien estructurado. Los métodos posibles son pideNumero, pideFrase, leer_matrizNumeros, leer_ValoresString, anadirNumero, anadirFrase, mostrar_matriz_numeros, mostrar_matrizString.

c) Implementación

```
import java.io.*;

class Ejercicio {

  static BufferedReader leer=
    new BufferedReader(new InputStreamReader(System.in));

  public static void main (String [] args) throws IOException {
    int [][] numeros = new int [3][2];
    String [][] frases = new String [3][2];
    int numero;
    String frase;
    leer_matrizNumeros(numeros);
    leerValoresString(frases);
    mostrar_matrizNumeros(numeros);
    mostrar_matrizString(frases);
    numero=pideNumero();
    anadirNumero(numeros, numero);
    frase=pideFrase();
    anadirFrase(frases, frase);
    mostrar_matrizNumeros(numeros);
    mostrar_matrizString(frases);
  }

  static int pideNumero () throws IOException{
    int numero;
    System.out.println ("Introduzca un numero");
    numero=Integer.parseInt(leer.readLine());
    return numero;
  }
```

```
  static String pideFrase() throws IOException{
    String frase;
    System.out.println ("Introduzca una frase");
    frase=leer.readLine();
    return frase;
  }

  static void leer_matrizNumeros(int [][] matriz)
                                          throws IOException {
    int f,c,filas=matriz.length,cols=matriz[0].length;
    System.out.println(
               "Introduce los valores de la matriz de numeros");
    for(f=0; f<filas; f++)
      for(c=0; c<cols; c++) {
        System.out.print(f+" "+c+": ");
        matriz[f][c]=Integer.parseInt(leer.readLine());
      }
    }

  static void leerValoresString(String [][] m)
                                           throws IOException {
    int f,c,filas=m.length,cols=m[0].length;
    System.out.println(
           "\n Introduce los valores de la matriz de cadenas");
    for(f=0; f<filas; f++)
      for(c=0; c<cols; c++) {
        System.out.print(f+","+c+": ");
        m[f][c]= leer.readLine();
      }
    System.out.println();
  }

//No hace falta devolver la matriz porque los cambios que se
//hacen en una matriz dentro de un metodo permanecen fuera
//del metodo

  static void anadirNumero(int [][] m, int numero) {
    int f,c,filas=m.length,cols=m[0].length;

    for(f=0; f<filas; f++) {
      for(c=0; c<cols; c++)
        m[f][c]= m[f][c]+numero;
    }
  }
```

```
  static void anadirFrase(String [][] m, String frase) {
    int f,c,filas=m.length,cols=m[0].length;

    for(f=0; f<filas; f++) {
      for(c=0; c<cols; c++)
        m[f][c]= m[f][c]+frase;
    }
  }

  static void mostrar_matrizNumeros(int [][] m) {
    int f,c,filas=m.length,cols=m[0].length;
    System.out.println ("La matriz de numeros contiene:");
    for(f=0; f<filas; f++) {
      for(c=0; c<cols; c++)
        System.out.print(m[f][c]+"\t\t");
      System.out.println();
    }
    System.out.println();
  }

  static void mostrar_matrizString (String [][] m) {
    int f,c,filas=m.length,cols=m[0].length;
    System.out.println ("La matriz de cadenas contiene:");
    for(f=0; f<filas; f++) {
      for(c=0; c<cols; c++)
        System.out.print(m[f][c]+"\t\t");
      System.out.println();
    }
    System.out.println();
  }
}
```

34. Escribir un método que construya y devuelva matrices cuadradas de cualquier dimensión mayor o igual a 1, cuyos elementos sigan el patrón mostrado en las matrices de la figura (cada elemento es la suma de sus índices). La dimensión se pasará como parámetro al método. El resultados se imprimirá en el método main.

0	1	2
1	2	3
2	3	4

Tabla 4.2: Matriz 3x3

0	1	2	3	4
1	2	3	4	5
2	3	4	5	6
3	4	5	6	7
4	5	6	7	8

Tabla 4.3: Matriz 5x5

Solución:

```
class Ejercicio {
  public static void main (String [] args){
    int dimension=Integer.parseInt(args[0]);
    int [][]matriz=metodo(dimension);
    for(int i =0; i< dimension; i++){
      for(int j= 0; j< dimension ;j++){
        System.out.print (matriz[i][j]);
      }
      System.out.println();
    }
  }

  public static int[][]  metodo(int dimension){
    int [][] matriz = new int[dimension][dimension];
    for(int i =0; i< dimension; i++){
      for(int j= 0; j< dimension ;j++){
        matriz[i][j]= i+j;
      }
    }
    return matriz;
  }
}
```

35. Escribir un método que reciba como parámetro una matriz bidimensional con valores numéricos enteros y devuelva una matriz unidimensional de longitud 5 que contenga los 5 primeros elementos pares de la matriz bidimensional, recorriéndola por filas. Si hay menos de 5 elementos pares en la matriz bidimensional rellene las posiciones restantes de la matriz unidimensional con el valor -1. Además, escribir un programa que declare e inicialice una matriz bidimensional y, usando el método anterior, imprima por pantalla la matriz resultante de ejecutar dicho método.

Solución:

a) Análisis:

El enunciado nos indica qué debemos hacer: Declarar e inicializar una matriz bidimensional, a continuación recorrerla por filas para localizar los cinco primeros números pares que serán almacenados en una matriz unidimensional.

b) Diseño:

Sabemos por el enunciado que las estructuras que necesitamos son 2 matrices: Una bidimensional y otra unidimensional. Debemos recordar dividir el problema en funcionalidades para tener un programa bien estructurado. Los métodos posibles son busca_cinco y muestra_resultado.

c) Implementación

```
import java.io.*;
class Ejercicio {
  public static void main (String [] args) throws IOException {
    int[][] mat={{1,2,3},{4,5,6},{7,8,9},{10,11,12},{13,14,15}};
    int[] resultado=busca_cinco(mat);
    muestra_resultado(resultado);
  }
  public static int[] busca_cinco(int[][] mat){
    int []resultado = new int [5];
    int contador = 0;
    boolean var=true;
    for(int i=0;i< resultado.length;i++)
      resultado[i]=-1;
    for(int i = 0; i<mat.length && var; i++){
      for(int j = 0; j< mat[i].length && var;j++){
        if(mat[i][j]%2==0){
          resultado[contador++]=mat[i][j];
          if (contador>=5) var=false;
        }
      }
    }
    return resultado;
  }
  public static void muestra_resultado(int matriz[]){
    for (int i=0; i<5;i++)
      System.out.println (matriz [i]);
  }
}
```

36. Construir un método que reciba una matriz bidimensional de enteros y devuelva un vector de 5 posiciones donde se guarden los 5 primeros elementos pares de una matriz bidimensional recorriéndola de abajo a arriba y de derecha a izquierda. Si hay menos de 5 elementos pares en la matriz rellenar las posiciones restantes del vector con el valor -1.

Solución:

```
public static int[] recorrer(int[][] mat){
  int []resultado = new int [5];
  int contador = 0;

  for(int i=0;i< resultado.length;i++) {
    resultado[i]=-1;
  }

  for(int i = mat.length-1; i >= 0 ; i--){
    for(int j = mat[i].length-1; j>=0;j--){
      if(mat[i][j]%2==0) {
        resultado[contador++] = mat[i][j];
      }
    }
  }
  return resultado;
}
```

37. Construir en Java dos métodos que reciban como único parámetro una matriz de enteros. El primer método debe devolver la media aritmética de los enteros de la matriz. El segundo método debe devolver el valor máximo de la matriz. Construya un programa principal que lea una matriz de enteros, invoque a ambos métodos e imprima los valores devueltos.

Solución:

```
public static int media (int [] matriz){
  int valor=0;

  for (int i=0;i<matriz.length;i++){
    valor=valor+matriz[i];
  }
  valor=valor/matriz.length;
  return valor;
}
```

```
public static int maximo (int [] matriz){
  int valor=matriz[0];
  for (int i=1;i<matriz.length;i++){
    if (valor<matriz[i])
      valor=matriz[i];
  }
  return valor;
}

public static void main (String [] args){
  int[] matriz=null;
  int n,maximo,media;
  Scanner leer=new Scanner (System.in);
  System.out.print("Introduzca la longitud de la matriz: ");
  n=leer.nextInt();
  matriz=new int [n];
  System.out.println("\nIntroduzca los valores de la matriz " +
                                        "separados por blancos ");
  for (int i=0; i<n; i++){
    matriz[i] = leer.nextInt();
  }
  media=media(matriz);
  maximo=maximo(matriz);
  System.out.println ("\nLa media es: "+media);
  System.out.println("El maximo es: "+maximo);
}
```

38. Construir un programa que declare 2 matrices unidimensionales, una en la que se almacenen nombres de alumnos y otra que contenga sus notas. Ambas matrices serán de 10 elementos. El propio programa debe inicializar las matrices de nombres y de notas. El programa debe imprimir la lista de nombres con su nota al lado. Después se debe introducir en una nueva matriz llamada alumnosAprobados los alumnos que hayan aprobado e imprimir esta matriz. Use el método printf para imprimir por pantalla la información.

Solución:

a) Análisis:

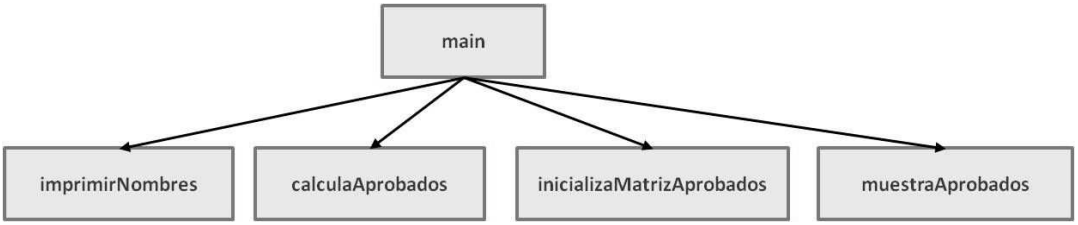
El problema nos pide mostrar por pantalla el nombre de un alumno y al lado su nota. Además debe salir después la lista de alumnos aprobados.

b) Diseño:

El enunciado nos dice que necesitamos 2 matrices unidimensionales, pero al especificar que tenemos otra matriz nueva llamada alumnosAprobados nos in-

dica que necesitaremos otra matriz unidimensional cuyo número de elementos lo podemos saber si calculamos previamente el número de alumnos que han aprobado. El tipo de dato de las matrices será double para la matriz de notas, y String para las 2 restantes. El código debe estar estructurado, por lo que tendremos que dividir el problema en funcionalidades. Un ejemplo puede ser tener un método para imprimir el nombre y la nota de los alumnos. Otro método que nos devuelva el número de alumnos que han aprobado. Un tercer método que copie los alumnos aprobados en la nueva matriz, y un cuarto para imprimir esta última matriz.

El diagrama de estructura del problema sería:



c) Implementación

```
class Ejercicio {

  public static void main(String[] args) {
    double[] notas = { 10, 3.5, 5, 8, 7.8, 10, 9, 9.5, 6, 5.6 };
    String[] nombres = new String[10];
    nombres[0] = "Ana";
    nombres[1] = "Juan";
    nombres[2] = "Pilar";
    nombres[3] = "Antonia";
    nombres[4] = "Jose";
    nombres[5] = "Paula";
    nombres[6] = "Africa";
    nombres[7] = "Javier";
    nombres[8] = "Raul";
    nombres[9] = "Alicia";
    byte contadorDeAprobados = 0;

    imprimirNombres(nombres, notas);
    contadorDeAprobados = calculaAprobados(notas);
    String[] alumnosAprobados = new String[contadorDeAprobados];
    inicializaMatrizAprobados(nombres, notas, alumnosAprobados);
    muestraAprobados(alumnosAprobados);
  }
```

```
  // Se imprimen los nombres de los alumnos y las notas
  public static void imprimirNombres(String[] nombres,
                                              double[] notas) {
    for (int i = 0; i < nombres.length; i++) {
      System.out.printf("%s : %.2f\n", nombres[i], notas[i]);
    }
  }
  // Se calcula el numero de aprobados
  public static byte calculaAprobados(double[] notas) {
    byte contador = 0;
    for (int i = 0; i < notas.length; i++) {
      if (notas[i] >= 5.0) {
        contador++;
      }
    }
    return contador;
  }

  // Se inicializa la matriz alumnosAprobados
  public static void inicializaMatrizAprobados(String[]nombres,
                           double[] nota, String[] aprobados) {
    byte contador = 0;
    for (int i = 0; i < nombres.length; i++) {
      if (nota[i] >= 5.0) {
        aprobados[contador] = nombres[i];
        contador++;
      }
    }
  }
  public static void muestraAprobados(String[] aprobados) {
    System.out.printf("Los alumnos aprobados son:\n");
    for (int i = 0; i < aprobados.length; i++) {
      System.out.printf("%s\n", aprobados[i]);
    }
  }
}
```

39. Construir un programa que muestre por pantalla el resultado de multiplicar dos matrices bidimensionales de números enteros: La matriz A (con dimensiones M y N) y la matriz B (con dimensiones O y P). Para implementar la entrada de datos, diseñe un método que solicite al usuario introducir las dimensiones y los valores de cada matriz. Use la clase BufferedReader para la entrada de datos y el método println para imprimir por pantalla la información.

Solución:

a) Análisis:

El enunciado nos da una especificación suficientemente clara, el problema consiste en la multiplicación de dos matrices que se expresaría matemáticamente como:

$$c(i,j) = \sum_{k=1}^{n} a(i,k) * b(k,j)$$

Los comentarios adicionales sobre la entrada y salida serían:

- Datos de salida: producto (matriz producto)
- Datos de entrada: A,B (matrices a multiplicar), m y n (dimensiones de la matriz a), p y q (dimensiones de la matriz b)

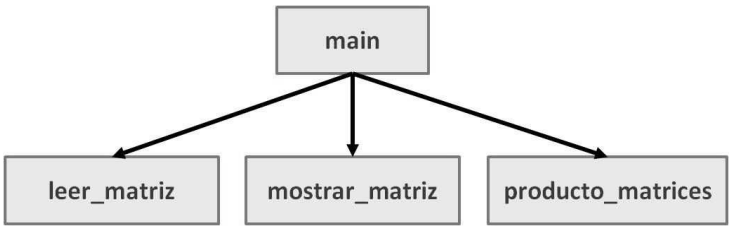
b) Diseño:

Se recorren las matrices con dos bucles anidados siempre que n sea igual a p. La matriz producto tendrá una dimensión de $m \times q$. Cada elemento de la matriz producto: producto (i,j) será:

$$p(i,j) = a(i,0) * b(0,j) + a(i,1) * b(1,j) + ... + a(i,n-1) * b(n-1,j)$$

por lo que dentro del bucle interno se necesita otro bucle que vaya recorriendo los valores de 0 a $n-1$.

El diagrama de estructura del programa sería:



El diseño de los algoritmos sería:

- Algoritmo principal

1: $A \Leftarrow leer\_matriz()$
2: $B \Leftarrow leer\_matriz()$
3: **if** $n \neq p$ **then**

4: Imprimir: No se pueden multiplicar las matrices
5: **else**
6: $producto \Leftarrow producto\_matrices(A, B)$
7: **end if**
8: **return** producto

- Algoritmo para el producto:

1: **for** i tal que $0 \leq i \leq m-1$ **do**
2: **for** j tal que $0 \leq j \leq q-1$ **do**
3: $producto(i, j) \Leftarrow 0$
4: **for** i tal que $0 \leq i \leq n-1$ **do**
5: $producto(i, j) \Leftarrow producto(i, j) + a(i, k) * b(k, j)$
6: **end for**
7: **end for**
8: **end for**

c) Implementación

```
// Producto de dos matrices a(m,n) y b(p,q)
// para dar producto(m,q), siendo n=p.

import java.io.*;

class Ejercicio {
  static BufferedReader leer=
    new BufferedReader(new InputStreamReader(System.in));

  public static void main (String [] args) throws IOException {
    //---Declaraciones---
    int[][] prod;
    int[][] a;
    int[][] b;
    //---Presentacion---
    System.out.println("Producto de dos matrices: "
                                        + "a(m,n) * b(p,q)");
    System.out.println("Recuerde que las dimensiones n de a "+
                                "y p de b deben ser iguales");
    System.out.println();
 //---Introduccion de datos---
    a=leer_matriz("a");
    b=leer_matriz("b");
    System.out.println();
```

```
    //---Realizacion del producto y visualizacion---
    if(a[0].length!=b.length)
      System.out.println("Las dimensiones n y p deben "
                                          + "ser iguales");
    else {
      mostrar_matriz("a",a);
      mostrar_matriz("b",b);
      prod=producto_matrices(a,b);
      mostrar_matriz("Producto",prod);
    }
  }

  static int[][] leer_matriz(String nombre)throws IOException {
    int f,c,filas,cols;
    int [][] m;

    System.out.print("Introduzca el numero de filas de "+
                                "la matriz " + nombre + ": ");
    filas=Integer.parseInt(leer.readLine());
    System.out.print("Introduzca el numero de columnas "+
                              "de la matriz " + nombre + ": ");
    cols=Integer.parseInt(leer.readLine());
    m = new int [filas][cols];
    for(f=0; f<filas; f++)
      for(c=0; c<cols; c++) {
       System.out.print(nombre+"("+f+","+c+"): ");
       m[f][c]=Integer.parseInt(leer.readLine());
      }
    System.out.println();

    return m;
  }

  static void mostrar_matriz(String nombre, int[][] m) {
    int f,c,filas=m.length,cols=m[0].length;

    System.out.println("Matriz " + nombre);
    for(f=0; f<filas; f++) {
      for(c=0; c<cols; c++)
        System.out.print(m[f][c]+"\t\t");
      System.out.println();
    }
    System.out.println();
  }
```

```
  static int[][] producto_matrices(int[][] a, int[][] b) {
    int i,j,k,m=a.length,q=b[0].length,n=a[0].length;
    int [][] prod = new int [m][q];

    for(i=0; i<m; i++)
      for(j=0; j<q; j++) {
        prod[i][j]=0;
        for(k=0; k<n; k++)
        prod[i][j]=prod[i][j]+a[i][k]*b[k][j];
      }
    return prod;
  }
}
```

40. Implementar un programa principal que cree una matriz bidimensional con los siguientes valores: $\{\{1,2\},\{3,4\}\}$ Además, implementar un método que intercambie el primer valor de la matriz con el último, por lo que la matriz quedaría así:

$$\begin{pmatrix} 4 & 2 \\ 3 & 1 \end{pmatrix}$$

El método debe devolver la matriz. Imprimir la matriz (en forma matricial) antes y después de realizar el intercambio de valores, para ello se debe invocar un método que habrá que implementar llamado *imprime*.

Solución:

```
class Ejercicio {

  public static void main (String [] args){
    int [][] matriz = {{1,2},{3,4}};
    imprime(matriz);
    matriz=intercambia(matriz);
    imprime(matriz);
  }

  public static void imprime (int [][] m){
    for (int i=0; i<m.length;i++){
      for (int j=0; j<m.length;j++)
        System.out.print(m[i][j]);
      System.out.println();
    }
    System.out.println();
  }
```

```
  public static int [][] intercambia( int [][] m){
    int aux;
    aux=m[0][0];
    m[0][0]=m[1][1];
    m[1][1]=aux;
    return m;
  }
}
```

41. La matriz transpuesta de una dada se define como aquella que intercambia filas por columnas (el elemento (i, j) pasa a ser el (j, i)). Escribir un método que transponga una matriz cuadrada. El método debe devolver la matriz transpuesta sin sobrescribir la matriz original. Para probar el programa construir un método main que inicialice una matriz y calcule su transpuesta imprimiéndola posteriormente.

Solución:

a) Análisis

Se puede considerar la matriz dividida en un triángulo superior y otro inferior sobre la diagonal. La transposición se realiza intercambiando los elementos simétricos sobre la diagonal. Al hacer la transpuesta sólo hay que transponer un triángulo de la matriz. El otro queda automáticamente cambiado.

b) Diseño

Para construir la transpuesta sólo hay que considerar un triángulo de la matriz. Para cada elemento de ese triángulo se intercambia su valor con el elemento de la matriz que tiene intercambiados los índices. Por eso la j sólo va hasta $j <= i$ (incluyendo la diagonal que se queda como está). El algoritmo en pseudocódigo sería:

1: Leer matriz a
2: **for** i tal que $0 \leq i \leq n$ **do**
3: **for** j tal que $0 \leq j \leq i$ **do**
4: $t(i, j) \Leftarrow a(j, i)$
5: $t(j, i) \Leftarrow a(i, j)$
6: **end for**
7: **end for**
8: **return** t

c) Implementación

```
class Ejercicio {
  public static void main (String [] args ) {
    int n;
    double [][] a = {{1.2 ,4.3 ,8.4} ,{5.1 ,7.2 ,9.1},
                                        {0.9 ,6.7 ,2.6}};
    double [][] transpuesta;
    n=a.length;
    System.out.println("Matriz original");
    for (int i=0; i<n; i++) {
      for (int j=0; j<n; j ++){
        System.out.print (a[i][j]+" ");
      }
      System.out.println ();
    }
    transpuesta=transponer (a);
    System.out.println ("\nMatriz transpuesta");
    for (int i=0; i<n; i++) {
      for (int j=0; j<n; j ++){
        System.out.print (transpuesta[i][j]+" ");
    }
    System.out.println();
    }
  }
  public static double [][] transponer ( double [][] a) {
    double[][] transpuesta = new double[a.length][a[0].length];
    for (int i=0; i<a.length; i ++){
      for (int j=0; j <= i; j++) {
        transpuesta [i][j] = a[j][i];
        transpuesta [j][i] = a[i][j];
      }
    }
    return transpuesta ;
  }
}
```

42. Escribir un programa que dada una matriz que contenga los diez primeros enteros mayores que cero, permita sustituir un elemento de una determinada posición de la matriz por otro introducido por teclado. Esta operación se debe realizar usando un método. Construir otro método que devuelva la suma de todos los elementos de la matriz modificada. En el método principal imprimir la matriz modificada y su suma. El programa debe indicar si se ha introducido una posición de la matriz no válida.

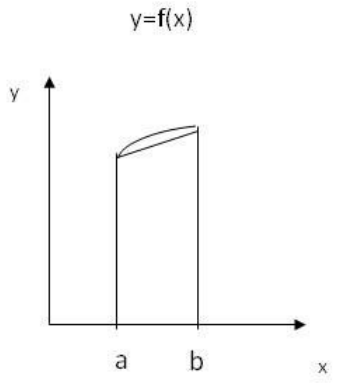
Solución:

```
import java.io.*;
class Ejercicio {
  public static void main ( Stringu [] args ) throws IOException {
    int elemento = 0, posicion = 0, opcion = 0, valor,suma;
    int enteros[] = new int [10];
    BufferedReader leer = new BufferedReader (
                              new InputStreamReader (System .in));
    for (int i=0; i< enteros.length ; i++) {
      enteros [i]=i+1;
      System.out.print(enteros[i]+ " ");
    }
    do {
      System.out.println ("\nIntroduzca el numero a cambiar y " +
                                  " su posicion (respecto a 0) ");
      elemento = Integer.parseInt(leer.readLine());
      posicion = Integer.parseInt(leer.readLine());
      valor = cambiar_elemento (elemento, posicion, enteros);
      opcion=0;
      if ( valor == -1) {
        System.out.println ("Posicion no valida ");
        System.out.println ("Si desea cambiar otro elemento " +
                             "pulse 1, si no pulse otro nÃºmero");
        opcion = Integer.parseInt(leer.readLine());
      }
      else {
        System.out.print("\nMatriz cambiada: ");
        for (int i=0; i< enteros . length ; i++) {
          System.out.print (enteros [i]+" ");
        }
        suma=sumar_matriz(enteros);
        System.out.println("\nSu suma: "+suma);
      }
    } while (opcion==1);
  }
  static int cambiar_elemento (int ele, int pos, int [] matriz ){
    int a=0;
    if (0 <= pos && (pos < matriz.length )) { // posicion valida
      matriz[pos]= ele ;
    }
    else {
      a = -1;
    }
    return a;
  }
```

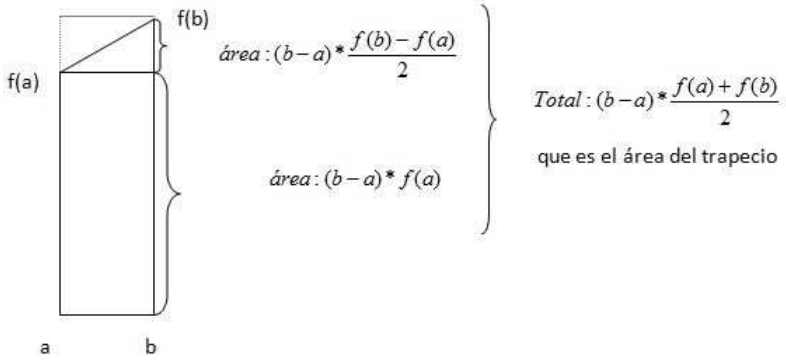
```
  static int sumar_matriz(int [] matriz){
    int suma=0;
    for (int i=0; i<matriz.length; i++){
      suma=suma+matriz[i];
    }
    return suma;
  }
}
```

43. Escribir un método que aplique la regla del Trapecio para calcular numéricamente la integral de $seno(\theta)$ entre los valores $\theta = 0 - \pi$ y devuelva dicho valor.

    Se sabe que el área debajo de una curva es la integral entre dos puntos. Una forma de calcular numéricamente la integral es mediante la regla del trapecio. La regla del trapecio consiste en:

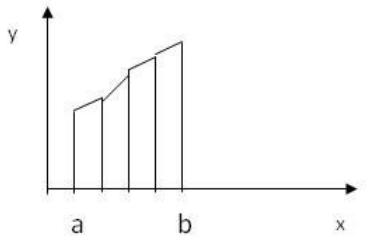


    Si se unen los puntos inicial (a) y final (b) de la curva con una recta obtenemos un trapecio. Después se aproxima la integral por el área del trapecio de la siguiente forma:



Si se pusiera un solo trapecio entonces el error sería enorme, por lo que es necesario poner muchos trapecios pequeños. Cuanto más pequeños sean menos error se cometerá y más se parecerá la línea curva a la recta del trapecio, en el límite coincidirán.

Si se tienen n trapecios igualmente espaciados por h, h=(b-a)/n, siendo n el número de trapecios:



la integral sería la suma de todas las áreas de los trapecios:

$$Integral = h * (\frac{f(a) + f(a+h)}{2} + \frac{f(a+h) + f(a+2*h)}{2} + \\ \ldots + \frac{f(a) + f(a+(n-1)*h)}{2} + \frac{f(b) + f(a+(n-1)*h)}{2})$$

con lo cual al final se tendría,

$$Integral = h * \left[\frac{f(a) + f(b)}{2} + \sum_{i=1}^{n-1} f(a + i * h)\right]$$

Como $f = sen\theta$ y se quiere calcular la función entre 0 y $\pi \Rightarrow h = (\pi - 0)/n$

Solución:

```
// Metodo que aplica el metodo de los trapecios para el calculo
// numerico de la integral seno(x) entre x=0 y x=PI
public static double calcular_integral(int n){
  double integral, h;
  h = (Math.PI-0)/n; // Incremento a usar

  // Ahora se aplica la regla del trapecio
  integral = (Math.sin(0)+ Math.sin(Math.PI))/2;
  for (int i=1; i<n; i++)
    integral = integral + Math.sin (0+i*h);
  integral = integral * h;
  return integral;
}
```

Como se puede observar, cuanto mayor es n más se parece el valor de la integral calculada numéricamente a la calculada analíticamente:

N	Integral
10	1.983523
100	1.999835
1000	1.999998

44. Escribir un método iterativo que localice la única raíz real de la curva $x^3+2x+1=0$ usando el esquema de reestructuración, $x_{n+1} = -(1+x_n^3)/2$ y sabiendo que la raíz está en los alrededores de $x = -0{,}4$, y devuelva su valor. Considerar una precisión de 10^{-4}.

    La idea es partir de un valor razonable para la solución y a partir de ese valor obtener otro más cercano de ese otro, y así sucesivamente, hasta que el proceso converja con tantos decimales como nos interese. Así pues se puede poner la solución en función de sí misma, tomando la ecuación de la curva y de ella despejando x en función de la propia x. Existen varias formas de hacerlo, obteniendo diferentes expresiones de reestructuración. No todas son válidas, se debe cumplir la condición de que la primera derivada de aquello a lo que se iguala la x es menor que 1 en los alrededores del punto considerado.

    La forma propuesta en el enunciado cumple esta condición:

$$x_{n+1} = -(1+x_n^3)/2$$

$$f' = -3x_{n^2}/2 = -(=-3/2) * 0{,}16 = -0{,}24$$

    Solución:

```
public static double calcular_raiz() {
  final double TOL = 1.e-4;
  double x_nueva, x_antigua;
  x_nueva = -0.4; // aproximacion inicial de la raiz
  do {
    x_antigua = x_nueva;
    x_nueva = -(1+ Math.pow (x_antigua, 3))/2;
  } while ( Math.abs (x_nueva - x_antigua )> TOL);
  return x_nueva;
}
```

45. Escribir un programa que calcule el máximo y el mínimo de n números introducidos como argumentos por la línea de órdenes.

Solución:

a) Análisis

Una vez más en este ejemplo tan sencillo los requisitos están claramente establecidos en el enunciado. Así, la lectura será a través de la línea de órdenes lo mismo que la salida. La funcionalidad está clara: obtener un valor máximo y uno mínimo.

b) Diseño:

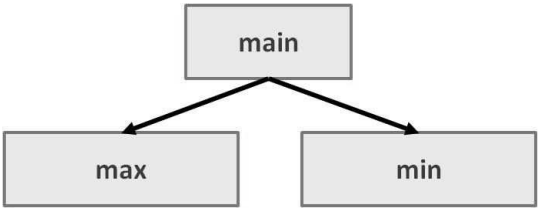
1) Estructuras de datos.
Lo más cómodo es usar una matriz monodimensional para almacenar y procesar los datos. Esto va a permitir usar un bucle para el procesamiento.

2) Algoritmos.
Para obtener el máximo y el mínimo se puede ir seleccionando los valores mayores o menores de la lista que se tenga usando un bucle. El pseudocódigo para la determinación del máximo quedaría de la siguiente forma:

```
1: maximo ⇐ valores[0]
2: for i tal que 1 ≤ i ≤ n do
3:    if maximo < valores[i] then
4:       maximo ⇐ valores[i]
5:    end if
6: end for
```

La determinación del mínimo es trivial visto el ejemplo anterior, basta con sustituir la comparación $maximo < valores[i]$ por $minimo > valores[i]$.

3) Diagrama de estructura.
Se puede modularizar el programa definiendo dos métodos que determinen uno el máximo (método max) otro el mínimo (método min). Con esto el diagrama de estructura sería:



c) Implementación

```
class Ejercicio {
  public static void main (String[] args) {
    double maximo, minimo;
    double [] valores = new double [args.length];
    // Asignacion de los datos leidos y eco de la entrada
    for (int i=0; i <args.length; i++){
      valores [i]= Double.parseDouble (args[i]);
      System.out.println ("valor["+i+"]: "+ valores [i]);
    }
    maximo = max(valores);
    minimo = min(valores);
    // Salida de resultados
    System.out.println ("valor maximo: " + maximo );
    System.out.println ("valor minimo: " + minimo );
  }
  static double max( double[] valores ) {
    double max;
    max = valores [0];
    for (int i=1; i <valores.length; i++){
      if ( max < valores [i]) max = valores [i];
    }
    return max ;
  }
  static double min ( double[] valores ) {
    double minimo ;
    minimo = valores[0];
    for (int i=1; i <valores.length; i ++){
      if ( minimo > valores[i]) minimo = valores [i];
    }
    return minimo ;
  }
}
```

46. Escribir un programa que genere un boleto de lotería primitiva con el número de apuestas elegido por el usuario. El programa hará uso del concepto de modularización, conteniendo los métodos necesarios para la generación del boleto. En el método main se leerá el número de apuestas con la clase Scanner y se imprimirá el boleto (números de cada apuesta, reintegro y precio) con printf. Un boleto de lotería primitiva se compone de n apuestas de 6 números enteros elegidos al azar en el intervalo de 1 a 49 y un reintegro elegido al azar entre 0 y 9. El coste de cada apuesta es de 1 euro.

Solución:

a) Análisis

Se trata de una simulación. Nuestro programa debe realizar la simulación de la primitiva generando uno de los valores enteros comprendidos entre 1 y 49 para un número determinado de apuestas. También se debe generar un número entre 0 y 9 para el reintegro.

b) Diseño

El método random() de la clase Math devuelve un double mayor o igual que 0 y menor que 1. Nosotros queremos que nos devuelva los valores 1-49. Para ello debemos cambiar la escala del resultado del método. Como queremos que el valor máximo sea 49 y el origen sea 1, multiplicamos por 49 el resultado del número aleatorio. Obtendremos como máximo el valor 48,999999 y como valor mínimo 0. Como queremos que el valor mínimo sea 1, sumamos 1 al resultado obtenido. Tendremos entonces valor mínimo 1 y valor máximo 49,9999999. Como sólo queremos números enteros nos quedamos con la parte entera, usando un molde. La simulación se encapsulará en un método. Para asegurarnos que los 6 números generados de la lotería sean distintos construimos un método que compruebe si el número generado ya ha sido generado en las iteraciones anteriores. En el caso del reintegro vamos a hacer uso de la clase Random con el método nextInt(int n) que devuelve un número aleatorio entre 0 y n, siendo n exclusivo.

1) Estructuras de datos: Matriz monodimensional
2) Diagrama de estructura:
   NOTA: En esta solución se usan dos formas distintas de generar los números aleatorios. Con el método nextInt(int n) de la clase Random es mucho más fácil en ambos casos. Con el método random de la clase Math se aprende como escalar cuando el intervalo de números a generar no es directamente el que nos da el método aleatorio.
3) Pseudocódigo:
   - Algoritmo principal

```
1: for all numero de apuestas do
2:    for 6 numeros por cada apuesta do
3:       Generar numero aleatoriamente
4:       if numero ya existe then
5:          Generar otro numero
6:       end if
```

7: Escribir numero
8: **end for**
9: Ordenar numeros apuesta
10: **end for**
11: Generar reintegro
12: Calcular precio del boleto

- ComprobarNumero

1: $existe \Leftarrow falso$
2: **for all** i tal que $0 \leq i \leq j$ AND $existe \neq falso$ **do**
3: **if** $valor = matriz(i)$ **then**
4: $existe \Leftarrow verdadero$
5: **end if**
6: **end for**

c) Implementación

```
import java.util.Scanner;
import java.util.Random;

class Ejercicio {
  public static void main(String [] args) {

    int napuestas, reintegro, numero;
    boolean existe;
    int precio=1; //1 euro por apuesta
    int []apuesta;
    apuesta= new int [6];
    Scanner leer = new Scanner(System.in);

    System.out.println("Numero de apuestas a realizar:");
    napuestas=leer.nextInt();

    //Obtencion de las n apuestas

    for (int i=0;i<napuestas;i++) {
      System.out.print("\nResultado de la apuesta "
                                        + (i+1) + ": ");
      for (int j=0;j<=5;j++){
        existe=true;
        numero=generar_numero();
        if (j!=0) {
          while (existe) {
            existe=comprobar_numero(numero,apuesta,j);
```

```
            if (existe){
              numero=generar_numero();
            }
          }
        }
        apuesta[j]=numero;
        System.out.printf(" %2d", apuesta[j]);
      }

/*Aqui podria venir una llamada a un metodo de ordenacion
 para colocar de menor a mayor los 6 numeros de cada apuesta*/

    }//fin for externo

    //Obtencion del reintegro
    reintegro=generar_reintegro();
    System.out.printf("\n\nReintegro: "+reintegro);
    //Obtencion del precio del boleto
    precio=precio*napuestas;
    System.out.printf("\n\nPrecio: %d euros", precio);

  }

  public static int generar_numero() {
    int resultado;
    resultado=((int)(1+49*Math.random()));
    return resultado;

  }

  public static boolean comprobar_numero(int valor,
                                  int[] matriz, int j){
    boolean existe=false;
    for (int i=0; i<j && !existe; i++){
      if (valor == matriz[i]) existe = true;
    }
    return existe;
  }

  public static int generar_reintegro() {
    Random rd=new Random();
    //Genera un nuumero aleatorio entre 0 y 9
    return rd.nextInt(10);
  }
}
```

47. Escribir un método que reciba como parámetro una matriz bidimensional de cadenas. El método debe devolver una matriz de caracteres, donde en las columnas pares se colocará el primer carácter de la cadena correspondiente y en las columnas impares el último carácter de la palabra correspondiente. Construir un programa de prueba.

    Nota: Se considerará como orden de las columnas el propio de Java para numerar las filas. Se considera 0 como par.

    Solución:

```
class Ejercicio {
  public static void main (String [] args){
    String [][] matrizA = {{"casa","duende","todo"},
             {"domo", "oro", "casa"}, {"rap", "pata", "nino"}};
    char [][] matrizB;
    matrizB=metodo(matrizA);
    for (int i = 0; i < matrizA.length; i++){
      for (int j = 0; j < matrizA[0].length; j++){
        System.out.print(matrizB[i][j]);
      }
      System.out.println();
    }
  }

  public static char[][] metodo( String[][] matrizA){
    char[][] matrizB = new
    char[matrizA.length][ matrizA[0].length];
    for (int i = 0; i < matrizA.length; i++){
      for (int j = 0; j < matrizA[i].length; j++){
        if ( j%2 == 0 ){
          matrizB[i][j] = matrizA[i][j].charAt(0);
        } else {
          matrizB[i][j] =
          matrizA[i][j].charAt(matrizA[i][j].length()-1);
        }
      }
    }
    return matrizB;
  }
}
```

48. Escribir un método que reciba como parámetro una matriz A que sea cuadrada y real. En dicho método debe construirse una matriz B, también real, que guarde en las filas pares la parte entera de las filas pares de la matriz A, y en las impares

incremente en 5 unidades el valor de las filas impares de la matriz A. Al finalizar el método, la matriz que se pasó como parámetro debe conservar sus valores originales. El método debe devolver la matriz B. Construir un programa de prueba.

Nota: Se considerará como orden de las filas el propio de Java para numerar las filas. El valor 0 se considerará como número par.

Solución:

```
class Ejercicio {
public static void main (String [] args){
  double [][] matrizA =
    {{2.4,4.5,6.7},{1.2,2.3,3.4}, {1.2,3.4,6.3}};
  double [][] matrizB;

  matrizB=modificar(matrizA);

  for (int i = 0; i < matrizA.length; i++){
    for (int j = 0; j < matrizA[0].length; j++){
      System.out.print(matrizB[i][j]+" ");
    }
    System.out.println();
    }
  }

  public static double[][] modificar( double[][] matrizA){
    double[][] matrizB = new
              double[matrizA.length][matrizA[0].length];
    for (int i = 0; i < matrizA.length; i++){
      for (int j = 0; j < matrizA[i].length; j++){
        if ( i%2 == 0 ){
          matrizB[i][j] = (int)matrizA[i][j];
        } else {
          matrizB[i][j] = matrizA[i][j] + 5;
        }
      }
    }
    return matrizB;
  }
}
```

49. Escribir un método que invierta un vector de enteros. El método debe recibir como argumento el vector a invertir y devolver, en otro vector distinto, el vector invertido. Nota: un vector inicial [1, 2, 3, 4] debería quedar después de la inversión como [4, 3 ,2 ,1].

Escribir un programa principal que lea el vector de enteros e imprima el vector resultante de la inversión.

Solución:

```
import java.util.Scanner;
class Ejercicio {
  public static void main (String [] args){
    int n;
    int [] vector=null;
    int [] vector_inverso=null;
    Scanner leer=new Scanner (System.in);
    System.out.print("Introduzca el numero de elementos: ");
    n=leer.nextInt();
    vector=new int [n];
    System.out.println("Introduzca los "+n
       +" elementos del vector separados por blancos");
    for (int i=0;i<n;i++){
      vector [i]=leer.nextInt();
    }
    vector_inverso=cambiar(vector);

    System.out.print ("\nEl vector invertido es: ");
    for (int i=0; i<vector.length;i++){
      System.out.print (vector_inverso[i]+" ");
    }
  }

//Metodo para invertir el vector

  public static  int[] cambiar(int[] mat){
    int []resultado = new int [mat.length];
    for(int i = 0;i<mat.length;i++){
      resultado [mat.length-i-1]=mat[i];
    }
    return resultado;
  }
}
```

50. Escribir un método que reciba como parámetro un vector de caracteres. En dicho método debe construirse un nuevo vector, también de caracteres. Este nuevo vector debe ser igual al vector original salvo porque los caracteres 'a' hayan sido sustituidos por el caracteres '1'. Dicho método debe devolver el nuevo vector.

    Ejemplo: $casa \Rightarrow c1s1$

Solución:

```
public static char[] vectorizar(char[] vector){
  char[] nuevoVector = new char[vector.length];
  for(int i = 0; i < vector.length; i++){
    if(vector[i] == 'a')
      nuevoVector[i] = '1';
    else
      nuevoVector[i] = vector[i];
  }
  return nuevoVector;
}
```

51. Dadas dos matrices reales a y b de dimensiones $n \times n$, es decir, dos matrices cuadradas, escribir un método en Java que acepte las dos matrices como parámetros (no se deben pasar como parámetros las dimensiones) y devuelva la matriz suma de estas dos.

Solución:

a) Análisis: Para sumar matrices ambas tienen que tener las mismas dimensiones. Los elementos de la matriz suma se obtienen como $suma(i,j) = a(i,j)+b(i,j)$.

b) Diseño: Se recorren las matrices con dos blucles anidados uno para filas y otro para columnas

1: Leer a,b
2: **for all** i tal que $0 \leq i \leq longitud\_fila - 1$ **do**
3: **for all** j tal que $0 \leq j \leq longitud\_columna - 1$ **do**
4: $suma(i,j) \Leftarrow a(i,j) + b(i,j)$
5: **end for**
6: **end for**
7: **return** suma

c) Implementación

```
public static double[][] sumar (double[][] a, double[][] b) {
  double [][] suma = new double [a.length] [a[0].length];
   for (int i=0; i<a.length; i++) {
    for (int j=0; j<a[0].length; j++) {
    suma[i][j] = a[i][j]+b[i][j];
    }
  }
  return suma;
}
```

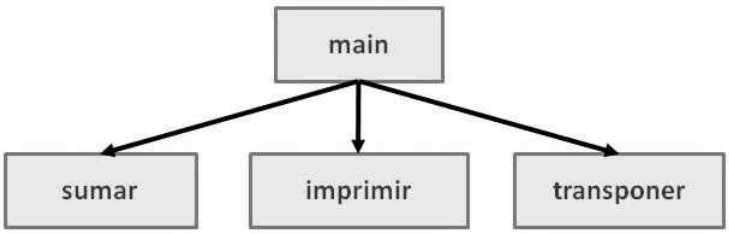
52. Escribir un programa que usando el métodos método sumar del ejercicio 51 y el transponer del ejercicio 41, sume dos matrices y transponga la matriz suma. El programa debe imprimir la suma de ambas matrices y su transpuesta usando el método printf. Para probar el programa simplemente inicialice dos matrices usando una lista de inicialización.

Solución:

a) Diseño

Vamos a diseñar un método main que invoque a los dos métodos el de suma y el de transposición y adicionalmente a un método imprimir que servirá para imprimir las dos matrices.

El diagrama de estructura sería:



b) Implementación

```
import java.util.Locale;
class Ejercicio {
  public static void main (String [] args) {
    double [][] a = {{0.1,1.1,2.1},{3.2,4.2,5.2},{6.3,7.3,8.3}};
    double [][] b = {{1.1,2.1,3.1},{4.2,5.2,6.2},{7.7,8.7,9.7}};
    double [][] c, d;

    c=sumar(a,b);
    System.out.println("\n\nMatriz Suma\n------------\n");
    imprimir(c);
    d=transponer(c);
    System.out.println("\n\nMatriz Transpuesta\n---------\n");
    imprimir (d);
  }
```

```
  public static double[][] sumar (double[][]a, double[][]b) {
  double [][] suma=new double [a.length] [a[0].length];
    for (int i=0; i<a.length; i++) {
      for (int j=0;j<a[0].length; j++) {
        suma[i][j]=a[i][j]+b[i][j];
      }
    }
    return suma;
  }

  public static double [][] transponer(double [][] a) {
    double[][] transpuesta=new double [a.length][a[0].length];
    for (int i=0; i<a.length; i++){
      for (int j=0; j<=i; j++) {
        transpuesta[i][j]=a[j][i];
        transpuesta[j][i]=a[i][j];
      }
    }
    return transpuesta;
  }

  public static void imprimir (double [][] matriz){
    for (int i=0; i<matriz.length;i++){
      for (int j=0; j<matriz[0].length;j++){
        System.out.printf(Locale.US,"%6.2f  ",matriz[i][j]);
      }
      System.out.println();
    }
  }

}
```

53. La criba de Eratóstenes es un algoritmo que permite hallar todos los números primos menores que un número natural dado N. Para ello, se forma una tabla con todos los números naturales comprendidos entre 2 y N y se van eliminando los números que no son primos de la siguiente manera: cuando se encuentra un número entero que no ha sido eliminado ese número se considera primo y se procede a eliminar todos sus múltiplos. El proceso termina cuando el cuadrado del mayor número primo encontrado es mayor que N. Escribir un programa en Java que lea un entero N y escriba los números primos existentes entre 1 y N. El programa debe implementar el cálculo de los primos en un método que aplique la criba de Eratóstenes.

Solución:

```
import java.util.*;
class Ejercicio {
  public static void main (String[] args){
    int n;
    Scanner leer = new Scanner (System.in);
    System.out.println("Introduzca un entero:");
    n=leer.nextInt();
    boolean [] primos = new boolean [n+1];
    eratostenes(primos);
    for (int i=1; i<=n; i++){
      if (primos[i]) System.out.printf ("%5d",i);
    }
  }

  public static void eratostenes (boolean [] primos){
    int j, n=primos.length-1;
    boolean seguir=true;
    for (int i=1; i<=n; i++){
      primos[i]= true;
    }
    j=2;
    while (seguir){
      if (j*j>n) {
        seguir=false;
      }
      else{
        for (int k=2; j*k<=n; k++){
          primos[j*k]=false;
        }
        j++;
      }
    }
  }
}
```

54. Dados dos vectores de longitud N representados por dos matrices monodimensionales (a y b), escribir un programa que calcule la distancia euclídea entre ellos. La distancia euclídea se define como la raíz cuadrada de las sumas de los cuadrados de las diferencias entre los elementos de cada uno de los vectores.

$$Distancia\_euclidea = \sqrt{(a[0]-b[0])^2 + (a[1]-b[1])^2 + \cdots + (a[n-1]+b[n-1])^2}$$

Requisitos del programa:

- Debe tener un método que calcule la matriz diferencia (matriz_dif) entre las matrices a y b y que devuelva dicha matriz.
- Debe tener otro método que calcule la raíz cuadrada de la suma de los cuadrados de los elementos de la matriz diferencia (matriz_dif) y que devuelva ese valor (que será la distancia euclídea).
- Los dos métodos anteriores se deben invocar desde el método main.
- En el método main se debe imprimir el valor de la distancia euclídea con cuatro decimales de precisión.
- Las matrices a y b deben ser reales.
- En el método main se debe leer por teclado la longitud de las matrices y crear y declarar las matrices usando la longitud leída. Posteriormente se deben leer los elementos de cada matriz.

Solución:

a) Análisis:

Directamente serán los requisitos del programa.

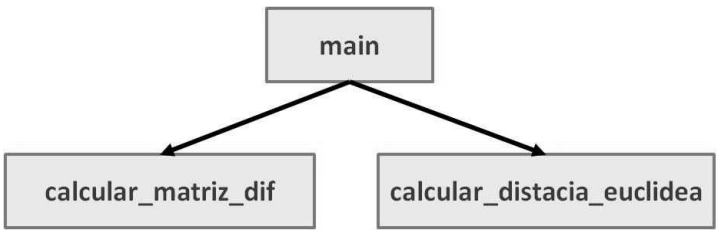
b) Diseño

- Pseudocódigo

1: Leer n
2: Dimensionar matrices a y b a longitud n
3: Leer a,b {Calcular matriz_diferencia}
4: $matriz\_dif \Leftarrow calcular\_matriz\_dif(a, b)$ {Calcular distancia_euclidea}

5: $matriz\_dif \Leftarrow calcular\_matriz\_dif(a, b)$
6: $euclidea \Leftarrow calcular\_distancia\_euclidea(matriz\_dif)$
7: Imprimir euclidea

- Diagrama de estructura



c) Implementación

```
import java.util.Scanner;
import java.util.Locale;
import java.util.Scanner;
import java.util.Locale;
class Ejercicio {
  public static void main(String [] args) {
    Scanner  leer=new Scanner(System.in);
    leer.useLocale(Locale.US);
    int n;
    double[] matriz_dif;
    double euclidea;
    double[] a, b;
    System.out.print ("Introduzca la longitud de "
                                    + "las matrices: ");
    n=leer.nextInt();
    a=new double [n];
    b=new double [n];

  //Lectura de las matrices a y b
    System.out.println("Introduzca los elementos de "
                                     + "la matriz a");
    for (int i=0; i<n;i++){
      a[i]=leer.nextDouble();
    }
    System.out.println("Introduzca los elementos de "
                                    + "la matriz b");
    for (int i=0; i<n;i++){
      b[i]=leer.nextDouble();
    }
    matriz_dif=calcular_matriz_dif(a,b);
    euclidea= calcular_distancia_euclidea(matriz_dif);
    System.out.print("\nLa distancia euclidea entre " +
                                       "a y b es: ");
    System.out.printf(Locale.US,"%.4f", euclidea);
  }

  public static double [] calcular_matriz_dif(double []a,
                                              double []b) {
    double [] c=new double[a.length];
    for (int i=0; i<c.length;i++){
      c[i]=a[i]-b[i];
    }
    return c;
  }
```

```
  public static double calcular_distancia_euclidea(
                                        double[] matriz_dif) {

    double suma=0;
    for (int i=0; i<matriz_dif.length; i++) {
      suma=suma+matriz_dif[i]*matriz_dif[i];
    }
    suma=Math.sqrt(suma);
    return suma;
  }
}
```