Capítulo 13

13.1.2 Ejercicios resueltos para do…while (Continuación)

Ejercicio 13.1.2.3

Elaborar un algoritmo que mida la inflación que han tenido ciertos artículos y que imprima el siguiente reporte:

ANALISIS DE INFLACION

ARTICULO	PRECIO ANTERIOR	PRECIO ACTUAL	PTJE. INFLACION
XXXXXXXXXX	99,999.99	99,999.99	99.99
XXXXXXXXXX	99,999.99	99,999.99	99.99
.	.	.	.
.	.	.	.
XXXXXXXXXX	99,999.99	99,999.99	99.99

Promedio de inflación: 99.99
Artículo con mayor inflación : XXXXXXXXXXXXXXXXXXXXXXXXXX
Porcentaje mayor de inflación : 99.99

Datos disponibles por cada artículo:
- Descripción
- Precio anterior
- Precio actual

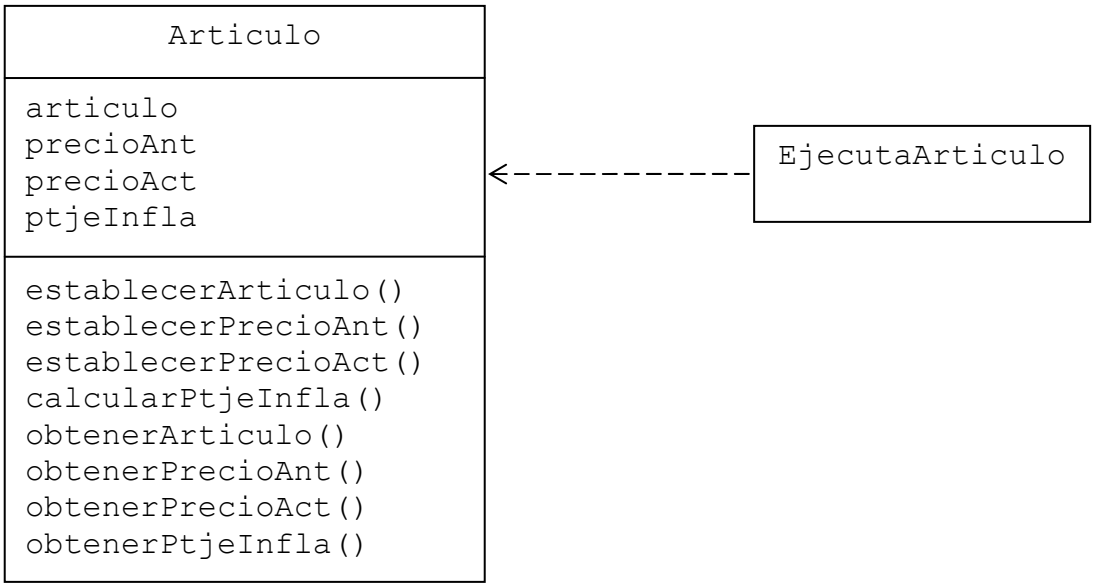
Forma de calcular el porcentaje de inflación:

$$\text{PTJE. INFLACION} = \frac{\text{Precio actual - Precio anterior}}{\text{Precio anterior}} \text{ X } 100$$

A continuación se presenta el algoritmo de la solución:

(Primero hágalo usted....después compare la solución)

Diagrama de clases




```
Algoritmo CALCULA INFLACION DE ARTICULOS
 Clase Articulo
  1. Declarar
       Datos
          articulo : Cadena
          precioAnt: Real
          precioAct: Real
          ptjeInfla: Real

  2. Método establecerArticulo(nom: Cadena)
     a. articulo = nom
     b. Fin Método establecerArticulo

  3. Método establecerPrecioAnt(pAnt: Real)
     a. precioAnt = pAnt
     b. Fin Método establecerPrecioAnt

  4. Método establecerPrecioAct(pAct: Real)
     a. precioAct = pAct
     b. Fin Método establecerPrecioAct

  5. Método calcularPtjeInfla()
     a. ptjeInfla = ((precioAct-precioAnt)/precioAnt)*100
     b. Fin Método calcularPtjeInfla

  6. Método obtenerArticulo() Cadena
     a. return articulo
     b. Fin Método obtenerArticulo

  7. Método obtenerPrecioAnt() Real
     a. return precioAnt
     b. Fin Método obtenerPrecioAnt
```

```
  8. Método obtenerPrecioAct() Real
     a. return precioAct
     b. Fin Método obtenerPrecioAct

  9. Método obtenerPtjeInfla() Real
     a. return ptjeInfla
     b. Fin Método obtenerPtjeInfla
 Fin Clase Articulo

 Clase EjecutaArticulo
  1. Método principal()
     a. Declarar
          Variables
            art, artMayor: Cadena
            totArticulos: Entero
            preAn, preAc, totInfla,
            mayorInfla, promInfla: Real
            otro: Carácter
     b. Imprimir encabezado
     c. totArticulos = 0
        totInfla = 0
        mayorInfla = 0
     d. do
           1. Declarar, crear e iniciar objeto
              Articulo objArti = new Articulo()
           2. Solicitar Articulo, precio actual y
                        precio anterior
           3. Leer art, preAc, preAn
           4. Establecer
                 objArti.establecerArticulo(art)
                 objArti.establecerPrecioAnt(preAn)
                 objArti.establecerPrecioAct(preAc)
           5. Calcular objArti.calcularPtjeInfla()
           6. Imprimir objArti.obtenerArticulo()
                       objArti.obtenerPrecioAnt()
                       objArti.obtenerPrecioAct()
                       objArti.obtenerPtjeInfla()
           7. if objArti.obtenerPtjeInfla() > mayorInfla then
                 a. mayorInfla = objArti.obtenerPtjeInfla()
                 b. artMayor = objArti.obtenerArticulo()
           8. endif
           9. totArticulos = totArticulos + 1
          10. totInfla = totInfla+objArti.obtenerPtjeInfla()
          11. Preguntar "¿Desea procesar otro articulo(S/N)?"
          12. Leer otro
     e. while otro == 'S'
     f. promInfla = totInfla / totArticulos
     g. Imprimir promInfla, artMayor, mayorInfla
     h. Fin Método principal
 Fin Clase EjecutaArticulo
Fin
```

En la zona de descarga de la Web del libro, está disponible:
Programa en Java: Articulo.java y EjecutaArticulo.java

Explicación:
El algoritmo tiene dos clases; la Clase Articulo y la clase EjecutaArticulo.
En la Clase Articulo:
1. Se declaran los datos que representan la estructura de la clase:
   articulo para el nombre del articulo
   precioAnt para el precio anterior del articulo
   precioAct para el precio actual del articulo
   ptjeInfla para el porcentaje de inflación del articulo
2. Método establecerArticulo(nom: Cadena)
   Recibe en el parámetro nom el valor que luego coloca en el dato articulo.
3. Método establecerPrecioAnt(pAnt: Real)
   Recibe en el parámetro pAnt el valor que luego coloca en el dato precioAnt.
4. Método establecerPrecioAct(pAct: Real)
   Recibe en el parámetro pAct el valor que luego coloca en el dato precioAct.
5. Método calcularPtjeInfla()
   Calcula ptjeInfla (porcentaje de inflación)
6. Método obtenerArticulo() Cadena
   Retorna articulo
7. Método obtenerPrecioAnt() Real
   Retorna precioAnt
8. Método obtenerPrecioAct() Real
   Retorna precioAct
9. Método obtenerPtjeInfla() Real
   Retorna ptjeInfla

Fin de la Clase Articulo

En la Clase EjecutaArticulo; en el Método principal():
a. Se declara:
   La variable art para leer el nombre del artículo.
   La variable preAn para leer el precio anterior.
   La variable preAc para leer el precio actual.
   La variable totArticulos para contar el total de artículos.
   La variable mayorInfla para colocar la mayor inflación.
   La variable artMayor para colocar el artículo con mayor inflación.
   La variable otro para controlar el ciclo repetitivo
b. Imprime encabezado
c. Inicia totArticulos, totInfla y mayorInfla en 0
d. Inicia ciclo do
   1. Se declara el objeto objArti, usando como base a la clase Articulo; dicho objeto se crea e inicializa mediante el constructor por defecto Articulo().
      Observe que cada vez que entra al ciclo crea un nuevo objeto articulo.
   2. Solicita Artículo, precio actual y precio anterior
   3. Lee en art, preAc, preAn

4. Se llama al método establecerArticulo(art) del objeto objArti; para colocar el valor de art en el dato articulo.
   Se llama al método establecerPrecioAnt(preAn) del objeto objArti; para colocar el valor de preAn en el dato precioAnt.
   Se llama al método establecerPrecioAct(preAc) del objeto objArti; para colocar el valor de preAc en el dato precioAct.
5. Se llama al método calcularPtjeInfla() del objeto objArti; para calcular el porcentaje de inflación.
6. Se llama al método obtenerArticulo() del objeto objArti; para acceder e imprimir el valor del dato articulo.
   Se llama al método obtenerPrecioAnt() del objeto objArti; para acceder e imprimir el valor del dato precioAnt.
   Se llama al método obtenerPrecioAct() del objeto objArti; para acceder e imprimir el valor del dato precioAct.
   Se llama al método obtenerPtjeInfla() del objeto objArti; para acceder e imprimir el valor del dato ptjeInfla.
7. Si objArti.obtenerPtjeInfla() > mayorInfla Entonces
   a. mayorInfla = objArti.obtenerPtjeInfla()
   b. artMayor = objArti.obtenerArticulo()
8. Fin del IF
9. Incrementa totArticulos en 1
10. Se incrementa totInfla con ptjeInfla; que se accede llamando al método obtenerPtjeInfla() del objeto objArti.
11. Pregunta "¿Desea procesar otro articulo(S/N)?"
12. Lee en otro la respuesta

e. Fin del ciclo (while desea == 'S'). Si se cumple regresa al do; si no, se sale del ciclo.
f. Calcula el promedio de inflación
g. Imprime el promedio de inflación, el artículo que tiene la mayor inflación y el mayor porcentaje de inflación
h. Fin Método principal
Fin de la Clase EjecutaArticulo
Fin del algoritmo

Ejercicio 13.1.2.4

En una empresa manufacturera de mesas. Se tienen varios obreros, y por cada obrero los datos:

Nombre del obrero: X----------X

Cada obrero puede haber trabajado varios días, por cada día que trabajó, se tiene un dato: Cantidad de unidades fabricadas (construidas).

Cantidad de unidades fabricadas: ---
Cantidad de unidades fabricadas: ---
.
Cantidad de unidades fabricadas: ---

Nota: cada cantidad fabricada es de un día de trabajo.

Nombre del obrero: X----------X

Cantidad de unidades fabricadas: ---
Cantidad de unidades fabricadas: ---
.
Cantidad de unidades fabricadas: ---

Elaborar un algoritmo que lea los datos de cada uno de los obreros y por cada obrero, la cantidad de unidades fabricadas de cada uno de los días que trabajó e imprima el siguiente reporte:

REPORTE DE PRODUCCION

NOMBRE DEL OBRERO	TOTAL PRODUCCION	SUELDO
XXXXXXXXXXXXXXXXXXXXXXXX	999	99,999.99
XXXXXXXXXXXXXXXXXXXXXXXX	999	99,999.99
.		
.		
.		
XXXXXXXXXXXXXXXXXXXXXXXX	999	99,999.99
TOTAL 999 OBREROS	9999	99,999.99

NOMBRE OBRERO MAS PRODUCTIVO: XXXXXXXXXXXXXXXXXXXXXX
PRODUCCION QUE FABRICO: 999

NOMBRE OBRERO MENOS PRODUCTIVO: XXXXXXXXXXXXXXXXXXXX
PRODUCCION QUE FABRICO: 999

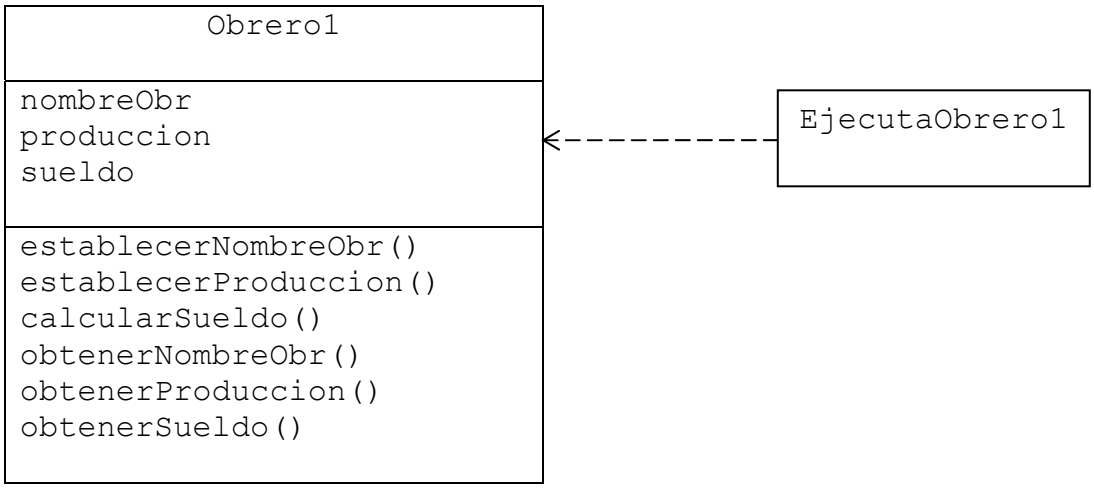
Cálculos:
Se lee el nombre de un obrero, luego cada una de las cantidades producidas por día, se suman estas cantidades para calcular el TOTAL PRODUCCION; en otras palabras, es la sumatoria de las cantidades producidas de todos los días que laboró.
El sueldo se calcula a 20.00 cada unidad fabricada si hizo 500 o menos; a 25.00 si hizo más de 500 y hasta 800; y a 30.00 si hizo más de 800.
Al final se pide el TOTAL de obreros, el TOTAL del TOTAL PRODUCCION y el total de los sueldos de todos los obreros. Además el nombre del obrero más productivo y la producción que fabricó, y el nombre del obrero menos productivo y la producción que fabricó. El obrero más productivo es el que tenga el TOTAL PRODUCCION mayor. El obrero menos productivo es el que tenga el TOTAL

PRODUCCION menor. Se supone que el TOTAL PRODUCCION de los obreros son diferentes.

A continuación se presenta el algoritmo de la solución:

(Primero hágalo usted...después compare la solución)

Diagrama de clases




```
Algoritmo CALCULA LA PRODUCCION DE VARIOS OBREROS
 Clase Obrero1
  1. Declarar
       Datos
          nombreObr: Cadena
          produccion: Entero
          sueldo: Real

  2. Método establecerNombreObr(nom: Cadena)
     a. nombreObr = nom
     b. Fin Método establecerNombreObr

  3. Método establecerProduccion(prod: Entero)
     a. produccion = prod
     b. Fin Método establecerProduccion

  4. Método calcularSueldo()
     a. if produccion <= 500 then
           1. sueldo = producción * 20.00
     b. endif
     c. if (produccion > 500)AND (produccion <= 800) then
           1. sueldo = producción * 25.00
     d. endif
     e. if produccion > 800 then
           1. sueldo = producción * 30.00
     f. endif
```

```
    g. Fin Método calcularSueldo

  5. Método obtenerNombreObr() Cadena
     a. return nombreObr
     b. Fin Método obtenerNombreObr

  6. Método obtenerProduccion() Real
     a. return produccion
     b. Fin Método obtenerProduccion

  7. Método obtenerSueldo() Real
     a. return sueldo
     b. Fin Método obtenerSueldo
 Fin Clase Obrero1

Clase EjecutaObrero1
  1. Método principal()
     a. Declarar
          Variables
            nombre, obrMayor, obrMenor: Cadena
            proDia, totProdObr, totProd, totObreros,
            mayorProd, menorProd: Entero
            totSueldos: Real
            otro, desea: Carácter
     b. Imprimir encabezado
     c. totObreros = 0
        totProd = 0
        totSuedos = 0
        mayorProd = 0
        menorProd = 10000
     d. do
          1. Declarar, crear e iniciar objeto
             Obrero1 objObrero = new Obrero1()
          2. Solicitar Nombre
          3. Leer nombre
          4. totProdObr = 0
          5. do
               a. Solicitar Producción del dia
               b. Leer proDia
               c. totProdObr = totProdObr + proDia
               d. Preguntar "¿Desea procesar otro dia (S/N)?"
               e. Leer otro
          6. while otro == 'S'
          7. Establecer
                   objObrero.establecerNombreObr(nombre)
                   objObrero.establecerProduccion(totProdObr)
          8. Calcular objObrero.calcularSueldo()
          9. Imprimir objObrero.obtenerNombreObr()
                      objObrero.obtenerProduccion()
                      objObrero.obtenerSueldo()
         10. if objObrero.obtenerProduccion()>mayorProd then
                a. mayorProd = objObrero.obtenerProduccion()
```

```
                b. obrMayor = objObrero.obtenerNombreObr()
          11. endif
          12. if objObrero.obtenerProduccion()<menorProd then
                a. menorProd = objObrero.obtenerProduccion()
                b. obrMenor = objObrero.obtenerNombreObr()
          13. endif
          14. totObreros = totObreros + 1
              totProd = totProd+objObrero.obtenerProduccion()
              totSuedos = totSuedos +objObrero.obtenerSueldo()
          15. Preguntar "¿Desea procesar otro obrero (S/N)?"
          16. Leer desea
      e. while desea == 'S'
      f. Imprimir totObreros, totProd, totSuedos
                  obrMayor, mayorProd, obrMenor, menorProd
      g. Fin Método principal
  Fin Clase EjecutaObrero1
Fin
```

En la zona de descarga de la Web del libro, está disponible:
Programa en Java: Obrero1.java y EjecutaObrero1.java

Explicación:
El algoritmo tiene dos clases; la Clase Obrero1 y la clase EjecutaObrero1.
En la Clase Obrero1:
1. Se declaran los datos que representan la estructura de la clase:
   nombreObr para el nombre del obrero
   produccion para la produccion del obrero
2. Método establecerNombreObr(nom: Cadena)
   Recibe en el parámetro nom el valor que luego coloca en el dato nombreObr.
3. Método establecerProduccion(prod: Entero)
   Recibe en el parámetro prod el valor que luego coloca en el dato produccion.
4. Método calcularSueldo()
   a. Si produccion <= 500 entonces
      1. Calcula sueldo = producción * 20.00
   b. Fin del if
   c. Si (produccion > 500)y (produccion <= 800) entonces
      1. Calcula sueldo = producción * 25.00
   d. Fin del if
   e. Si produccion > 800 entonces
      1. Calcula sueldo = producción * 30.00
   f. Fin del if
   g. Fin Método calcularSueldo
4. Método obtenerNombreObr() Cadena
   Retorna nombreObr
5. Método obtenerProduccion() Real
   Retorna produccion
7. Método obtenerSueldo() Real
   Retorna sueldo
Fin de la Clase Obrero1

En la Clase EjecutaObrero1; en el Método principal():

a. Se declara:
   La variable nombre para leer el nombre del obrero.
   La variable obrMayor para colocar el nombre del obrero más productivo.
   La variable obrMenor para colocar el nombre del obrero menos productivo.
   La variable proDia para leer la producción de cada día.
   La variable totObreros para contar el total de obreros.
   La variable totProdObr para calcular el total de producción del obrero.
   La variable totProd para calcular el total de producción de todos los obreros.
   La variable totSueldos para calcular el total de sueldos de todos los obreros.
   La variable mayorProd para colocar el mayor totProdObr.
   La variable menorProd para colocar el menor totProdObr.
   La variable otro y desea para controlar los ciclos repetitivos.

b. Imprime encabezado

c. Inicia totObreros, totProd, mayorProd en 0; y menorProd en 10000

d. Inicia ciclo do

1. Se declara el objeto objObrero, usando como base a la clase Obrero1; dicho objeto se crea e inicializa mediante el constructor por defecto Obrero1().
   Observe que cada vez que entra al ciclo crea un nuevo objeto obrero.
2. Solicita el nombre del obrero
3. Lee en nombre
4. Inicia totProdObr en 0
5. Inicia ciclo do
   a. Solicita la producción del día
   b. Lee en proDia
   c. Incrementa totProdObr con proDia
   d. Pregunta “¿Desea procesar otro dia (S/N)?”
   e. Lee en otro la respuesta
6. Fin del ciclo (while otro == ‘S’). Si se cumple regresa al do; si no, se sale del ciclo.
7. Se llama al método establecerNombreObr(nombre) del objeto objObrero; para colocar el valor de nombre en el dato nombreObr.
   Se llama al método establecerProduccion(totProdObr) del objeto objObrero; para colocar el valor de totProdObr en el dato produccion.
8. Se llama al método calcularSueldo() del objeto objObrero; para acceder e
9. Se llama al método obtenerNombreObr() del objeto objObrero; para acceder e imprimir el valor del dato nombreObr.
   Se llama al método obtenerProduccion() del objeto objObrero; para acceder e imprimir el valor del dato produccion.
   Se llama al método obtenerSueldo() del objeto objObrero; para acceder e imprimir el valor del dato sueldo.
10. if objObrero.obtenerProduccion()>mayorProd Entonces
    a. mayorProd = objObrero.obtenerProduccion()
    b. obrMayor = objObrero.obtenerNombreObr()
11. Fin del if
12. Si objObrero.obtenerProduccion()<menorProd Entonces

a. menorProd = objObrero.obtenerProduccion()
b. obrMenor = objObrero.obtenerNombreObr()

13. Fin del if
14. Incrementa totObreros en 1
    Se incrementa totProd con produccion; que se accede llamando al método obtenerProduccion() del objeto objObrero.
    Se incrementa totSueldos con sueldo; que se accede llamando al método obtenerSueldo() del objeto objObrero.
15. Pregunta "¿Desea procesar otro obrero (S/N)?"
16. Lee en desea la respuesta

e. Fin del ciclo (while desea == 'S'). Si se cumple regresa al do; si no, se sale del ciclo.
f. Imprime totObreros, totProd, totSueldos, obrMayor, mayorProd,
obrMenor, menorProd
g. Fin Método principal
Fin de la Clase EjecutaObrero1
Fin del algoritmo

13.2.1 Ejercicios resueltos para for (Continuación…)

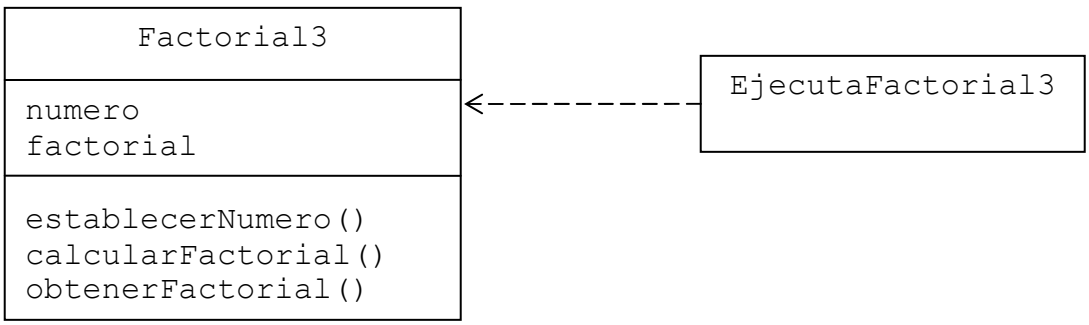
Ejercicio 13.2.1.3

Elaborar un algoritmo que pregunte a cuántos números se desea calcular el factorial; lea la cantidad en N. A continuación, debe leer un número e imprimir su factorial, luego leer otro, y así hasta leer los N números.

A continuación se presenta el algoritmo de la solución:

(Primero hágalo usted....después compare la solución)

Diagrama de clases




```
Algoritmo FACTORIALES DE N NUMEROS
 Clase Factorial3
  1. Declarar
       Datos
         numero: Entero
         factorial: Entero
```

```
  2. Método establecerNumero(nu: Entero)
     a. numero = nu
     b. Fin Método establecerNumero

  3. Método calcularFactorial()
     a. Declarar
          Variables
            i: Entero
     b. if numero == 0 then
         1. factorial = 1
     c. else
         1. factorial = 1
         2. for i=numero; i>=1; i--
              a. factorial = factorial * i
         3. endfor
     d. endif
     e. Fin Método calcularFactorial

  4. Método obtenerFactorial() Entero
     a. return factorial
     b. Fin Método obtenerFactorial
 Fin Clase Factorial3


 Clase EjecutaFactorial3
  1. Método principal()
     a. Declarar
        Variables
            n, j, num: Entero
     b. Solicitar Cantidad de números a calcular factorial
     c. Leer n
     d. for j=1; j<=n; j++
          1. Declarar, crear e iniciar objeto
             Factorial3 objFactorial = new Factorial3()
          2. Solicitar Número
          3. Leer num
          4. Establecer objFactorial.establecerNumero(num)
          5. Calcular objFactorial.calcularFactorial()
          6. Imprimir objFactorial.obtenerFactorial()
     e. endfor
     f. Fin Método principal
 Fin Clase EjecutaFactorial3
Fin
```

En la zona de descarga de la Web del libro, está disponible:
Programa en Java: Factorial3.java y EjecutaFactorial3.java

Explicación:

El algoritmo tiene dos clases; la Clase Factorial3 y la clase EjecutaFactorial3.

En la Clase Factorial3:

1. Se declaran los datos que representan la estructura de la clase:
   numero para el número
   factorial para el factorial
2. Método establecerNumero(nu: Entero)
   Recibe en el parámetro nu el valor que luego coloca en el dato numero.
3. Método calcularFactorial()
   a. Declarar
      Variables
      i: Entero
   b. Si numero == 0 Entonces
      1. factorial = 1
   c. Si no
      1. Inicia factorial en 1
      2. Inicia ciclo for desde i=numero hasta 1 con decrementos de 1
         a. Calcula factorial = factorial * i
      3. Fin del for
   d. Fin del if
   e. Fin Método calcularFactorial
4. Método obtenerFactorial()
   Retorna factorial

Fin de la Clase Factorial3

En la Clase EjecutaFactorial3; en el Método principal():

a. Se declara:
   La variable num para leer el número.
   La variable n para leer cuántos números son.
   La variable j para controlar el ciclo repetitivo.
b. Solicita Cantidad de números a calcular factorial
c. Lee en n
d. Inicia ciclo for desde j=1 hasta n con incrementos de 1
   1. Se declara el objeto objFactorial, usando como base a la clase Factorial3; dicho objeto se crea e inicializa mediante el constructor por defecto Factorial3().
   2. Solicita el número
   3. Lee en num
   4. Se llama al método establecerNumero(num) del objeto objFactorial; para colocar el valor de num en el dato numero.
   5. Se llama al método calcularFactorial() del objeto objFactorial; para calcular el factorial.
   6. Se llama al método obtenerFactorial() del objeto objFactorial; para acceder e imprimir el valor del dato factorial.
e. Fin del for
f. Fin Método principal

Fin de la Clase EjecutaFactorial3

Fin del algoritmo

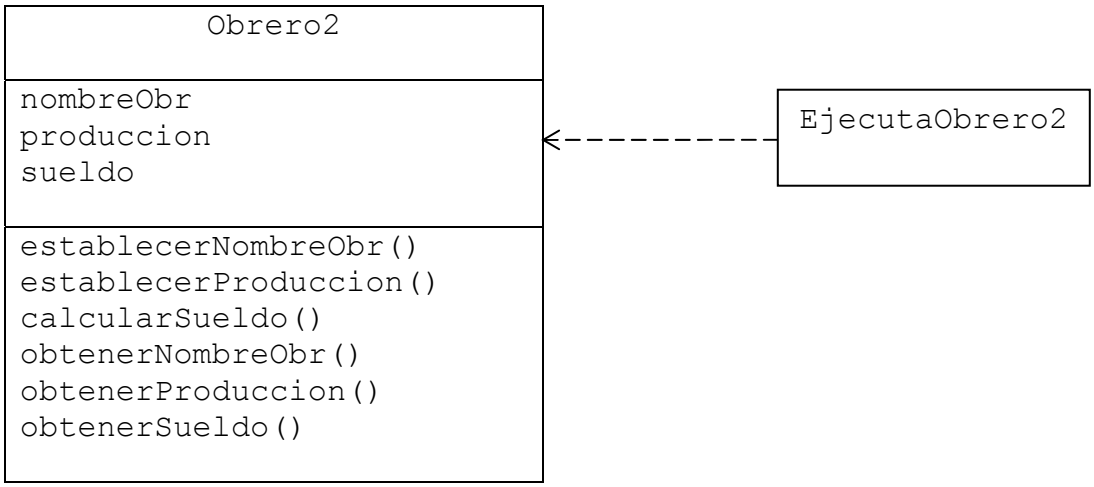
Ejercicio 13.2.1.4

Elaborar un algoritmo similar al Ejercicio 13.1.2.4, del punto anterior; solo que ahora se tienen 15 obreros, y cada obrero trabajó seis días.

A continuación se presenta el algoritmo de la solución:

(Primero hágalo usted...después compare la solución)

Diagrama de clases




```
Algoritmo CALCULA LA PRODUCCION DE 15 OBREROS
 Clase Obrero2
  1. Declarar
       Datos
          nombreObr: Cadena
          produccion: Entero
          sueldo: Real

  2. Método establecerNombreObr(nom: Cadena)
     a. nombreObr = nom
     b. Fin Método establecerNombreObr

  3. Método establecerProduccion(prod: Entero)
     a. produccion = prod
     b. Fin Método establecerProduccion

  4. Método calcularSueldo()
     a. if produccion <= 500 then
           1. sueldo = producción * 20.00
     b. endif
     c. if (produccion > 500)AND (produccion <= 800) then
```

```
          1. sueldo = producción * 25.00
    d. endif
    e. if produccion > 800 then
          1. sueldo = producción * 30.00
    f. endif
    g. Fin Método calcularSueldo

 5. Método obtenerNombreObr() Cadena
    a. return nombreObr
    b. Fin Método obtenerNombreObr

 6. Método obtenerProduccion() Real
    a. return produccion
    b. Fin Método obtenerProduccion

 7. Método obtenerSueldo() Real
    a. return sueldo
    b. Fin Método obtenerSueldo
Fin Clase Obrero2


Clase EjecutaObrero2
 1. Método principal()
    a. Declarar
         Variables
           nombre, obrMayor, obrMenor: Cadena
           obrero, dia, proDia, totProdObr, totProd,
           totObreros, mayorProd, menorProd: Entero
           totSueldos: Real
    b. Imprimir encabezado
    c. totObreros = 0
       totProd = 0
       totSuedos = 0
       mayorProd = 0
       menorProd = 10000
    d. for obrero=1; obrero<=15; obrero++
         1. Declarar, crear e iniciar objeto
            Obrero2 objObrero = new Obrero2()
         2. Solicitar Nombre
         3. Leer nombre
         4. totProdObr = 0
         5. for dia=1; dia<=6; dia++
                a. Solicitar Producción del dia
                b. Leer proDia
                c. totProdObr = totProdObr + proDia
         6. endfor
         7. Establecer
                objObrero.establecerNombreObr(nombre)
                objObrero.establecerProduccion(totProdObr)
         8. Calcular objObrero.calcularSueldo()
         9. Imprimir objObrero.obtenerNombreObr()
                     objObrero.obtenerProduccion()
```

```
                    objObrero.obtenerSueldo()
          10. if objObrero.obtenerProduccion()>mayorProd then
                 a. mayorProd = objObrero.obtenerProduccion()
                 b. obrMayor = objObrero.obtenerNombreObr()
          11. ENDIF
          12. if objObrero.obtenerProduccion()<menorProd then
                 a. menorProd = objObrero.obtenerProduccion()
                 b. obrMenor = objObrero.obtenerNombreObr()
          13. ENDIF
          14. totObreros = totObreros + 1
              totProd = totProd+objObrero.obtenerProduccion()
              totSuedos = totSuedos +objObrero.obtenerSueldo()
      e. endfor
      f. Imprimir totObreros, totProd, totSueldos,
                  obrMayor, mayorProd, obrMenor, menorProd
      g. Fin Método principal
 Fin Clase EjecutaObrero2
Fin
```

En la zona de descarga de la Web del libro, está disponible:
Programa en Java: Obrero2.java y EjecutaObrero2.java

Explicación:

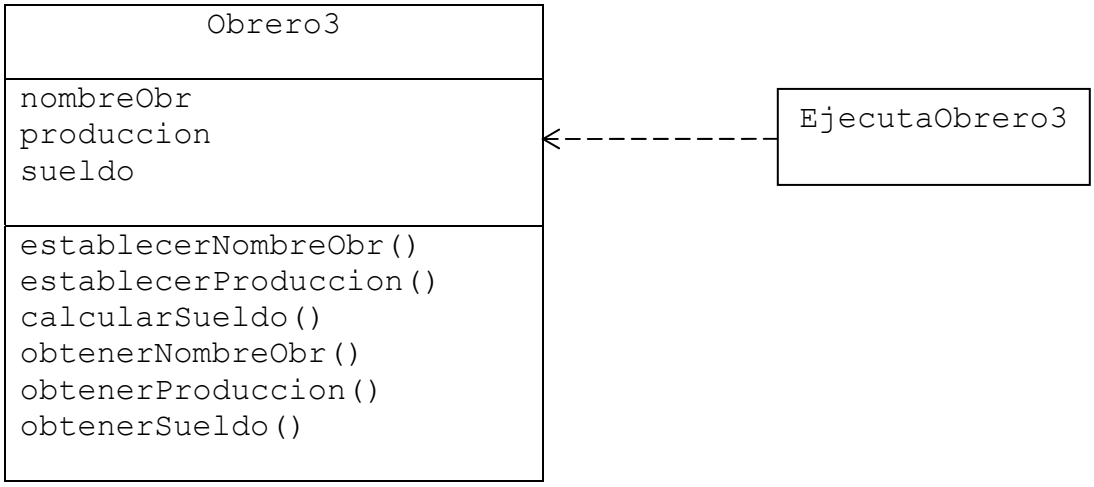
Este algoritmo es similar al Ejercicio 13.1.2.4 lo único que cambia es la forma de controlar los ciclos. Ahora se están usando dos for en lugar de los dos do; en consecuencia, también cambian las variables con las que se controlan, en lugar de desea y otro, ahora se usan obrero y día.

Ejercicio 13.2.1.5

Elaborar un algoritmo similar al Ejercicio 13.1.2.4, del punto anterior; solo que ahora se tienen varios obreros, y cada obrero trabajó seis días.

A continuación se presenta el algoritmo de la solución:

(Primero hágalo usted...después compare la solución)




```
Algoritmo CALCULA LA PRODUCCION DE 15 OBREROS
 Clase Obrero3
  1. Declarar
       Datos
          nombreObr: Cadena
          produccion: Entero
          sueldo: Real

  2. Método establecerNombreObr(nom: Cadena)
     a. nombreObr = nom
     b. Fin Método establecerNombreObr

  3. Método establecerProduccion(prod: Entero)
     a. produccion = prod
     b. Fin Método establecerProduccion

  4. Método calcularSueldo()
     a. if produccion <= 500 then
           1. sueldo = producción * 20.00
     b. endif
     c. if (produccion > 500)AND (produccion <= 800) then
           1. sueldo = producción * 25.00
     d. endif
     e. if produccion > 800 then
           1. sueldo = producción * 30.00
     f. endif
     g. Fin Método calcularSueldo

  5. Método obtenerNombreObr() Cadena
     a. return nombreObr
     b. Fin Método obtenerNombreObr

  6. Método obtenerProduccion() Real
     a. return produccion
     b. Fin Método obtenerProduccion
```

```
 7. Método obtenerSueldo() Real
    a. return sueldo
    b. Fin Método obtenerSueldo
Fin Clase Obrero3


Clase EjecutaObrero3
 1. Método principal()
    a. Declarar
         Variables
           nombre, obrMayor, obrMenor: Cadena
           dia, proDia, totProdObr, totProd,
           totObreros, mayorProd, menorProd: Entero
           totSueldos: Real
           desea: Carácter
    b. Imprimir encabezado
    c. totObreros = 0
       totProd = 0
       totSuedos = 0
       mayorProd = 0
       menorProd = 10000
    d. do
         1. Declarar, crear e iniciar objeto
            Obrero3 objObrero = new Obrero3()
         2. Solicitar Nombre
         3. Leer nombre
         4. totProdObr = 0
         5. for dia=1; dia<=6; dia++
                a. Solicitar Producción del dia
                b. Leer proDia
                c. totProdObr = totProdObr + proDia
         6. endfor
         7. Establecer
                objObrero.establecerNombreObr(nombre)
                objObrero.establecerProduccion(totProdObr)
         8. Calcular objObrero.calcularSueldo()
         9. Imprimir objObrero.obtenerNombreObr()
                     objObrero.obtenerProduccion()
                     objObrero.obtenerSueldo()
        10. if objObrero.obtenerProduccion()>mayorProd then
               a. mayorProd = objObrero.obtenerProduccion()
               b. obrMayor = objObrero.obtenerNombreObr()
        11. ENDIF
        12. if objObrero.obtenerProduccion()<menorProd then
               a. menorProd = objObrero.obtenerProduccion()
               b. obrMenor = objObrero.obtenerNombreObr()
        13. ENDIF
        14. totObreros = totObreros + 1
            totProd = totProd+objObrero.obtenerProduccion()
            totSuedos = totSuedos +objObrero.obtenerSueldo()
        15. Preguntar "¿Desea procesar otro obrero (S/N)?"
```

```
         16. Leer desea
     e. while desea == 'S'
     f. Imprimir totObreros, totProd, totSueldos,
                 obrMayor, mayorProd, obrMenor, menorProd
     g. Fin Método principal
 Fin Clase EjecutaObrero3
Fin
```

En la zona de descarga de la Web del libro, está disponible:
Programa en Java: Obrero3.java y EjecutaObrero3.java

Explicación:

Este algoritmo es similar al Ejercicio 13.1.2.4 lo único que cambia es la forma de controlar el segundo ciclo (el más interno). Ahora se está usando un for en lugar de do; en consecuencia, también cambia la variable con la que se controla, en lugar de otro, ahora se usa día.

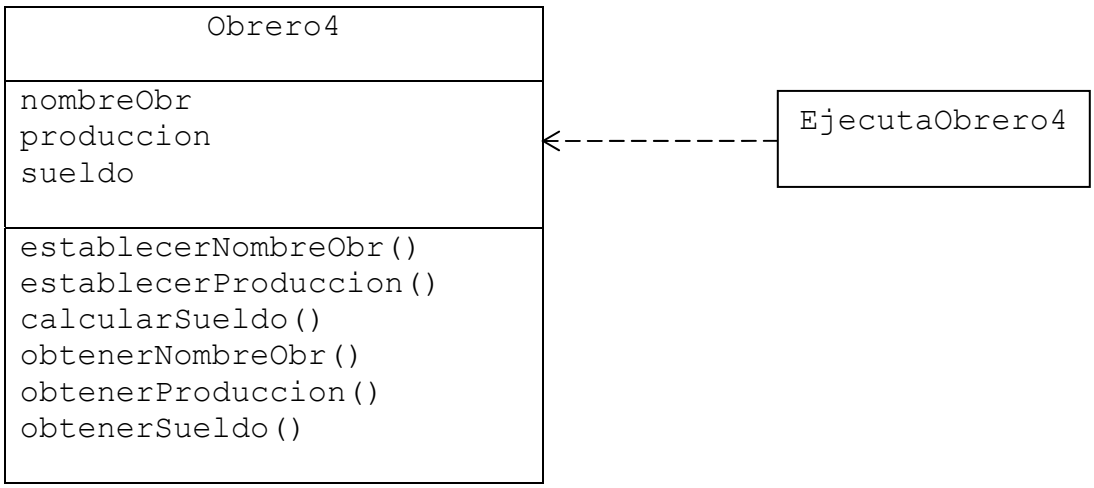
Ejercicio 13.2.1.6

Elaborar un algoritmo similar al Ejercicio 13.1.2.4, del punto anterior; solo que ahora se tienen 15 obreros, y cada obrero trabajó varios días.

A continuación se presenta el algoritmo de la solución:

(Primero hágalo usted...después compare la solución)

Diagrama de clases




```
Algoritmo CALCULA LA PRODUCCION DE 15 OBREROS
 Clase Obrero4
  1. Declarar
       Datos
         nombreObr: Cadena
         produccion: Entero
```

```
        sueldo: Real

 2. Método establecerNombreObr(nom: Cadena)
    a. nombreObr = nom
    b. Fin Método establecerNombreObr

 3. Método establecerProduccion(prod: Entero)
    a. produccion = prod
    b. Fin Método establecerProduccion

 4. Método calcularSueldo()
    a. if produccion <= 500 then
          1. sueldo = producción * 20.00
    b. endif
    c. if (produccion > 500)AND (produccion <= 800) then
          1. sueldo = producción * 25.00
    d. endif
    e. if produccion > 800 then
          1. sueldo = producción * 30.00
    f. endif
    g. Fin Método calcularSueldo

 5. Método obtenerNombreObr() Cadena
    a. return nombreObr
    b. Fin Método obtenerNombreObr

 6. Método obtenerProduccion() Real
    a. return produccion
    b. Fin Método obtenerProduccion

 7. Método obtenerSueldo() Real
    a. return sueldo
    b. Fin Método obtenerSueldo
Fin Clase Obrero4


Clase EjecutaObrero4
 1. Método principal()
    a. Declarar
         Variables
           nombre, obrMayor, obrMenor: Cadena
           obrero, proDia, totProdObr, totProd,
           totObreros, mayorProd, menorProd: Entero
           totSueldos: Real
           otro: Carácter
    b. Imprimir encabezado
    c. totObreros = 0
       totProd = 0
       totSuedos = 0
       mayorProd = 0
       menorProd = 10000
    d. for obrero=1; obrero<=15; obrero++
```

```
            1. Declarar, crear e iniciar objeto
               Obrero4 objObrero = new Obrero4()
            2. Solicitar Nombre
            3. Leer nombre
            4. totProdObr = 0
            5. do
                    a. Solicitar Producción del dia
                    b. Leer proDia
                    c. totProdObr = totProdObr + proDia
                    d. Preguntar "¿Desea procesar otro dia (S/N)?"
                    e. Leer otro
            6. while otro == 'S'
            7. Establecer
                    objObrero.establecerNombreObr(nombre)
                    objObrero.establecerProduccion(totProdObr)
            8. Calcular objObrero.calcularSueldo()
            9. Imprimir objObrero.obtenerNombreObr()
                        objObrero.obtenerProduccion()
                        objObrero.obtenerSueldo()
           10. if objObrero.obtenerProduccion()>mayorProd then
                  a. mayorProd = objObrero.obtenerProduccion()
                  b. obrMayor = objObrero.obtenerNombreObr()
           11. ENDIF
           12. if objObrero.obtenerProduccion()<menorProd then
                  a. menorProd = objObrero.obtenerProduccion()
                  b. obrMenor = objObrero.obtenerNombreObr()
           13. ENDIF
           14. totObreros = totObreros + 1
               totProd = totProd+objObrero.obtenerProduccion()
               totSuedos = totSuedos +objObrero.obtenerSueldo()
     e. endfor
     f. Imprimir totObreros, totProd, totSueldos,
                 obrMayor, mayorProd, obrMenor, menorProd
     g. Fin Método principal
 Fin Clase EjecutaObrero4
Fin
```

En la zona de descarga de la Web del libro, está disponible:
Programa en Java: Obrero4.java y EjecutaObrero4.java

Explicación:

Este algoritmo es similar al Ejercicio 13.1.2.4 lo único que cambia es la forma de controlar el primer ciclo (el más externo). Ahora se está usando un for en lugar de do; en consecuencia, también cambia la variable con la que se controla, en lugar de desea, ahora se usa obrero.

13.3.1 Ejercicios resueltos para while (Continuación…)

Ejercicio 13.3.1.3

En una fábrica se tienen 10 estaciones de trabajo, y la producción hecha en número de unidades fabricadas en cada uno de los días de la semana; cada estación pudo haber trabajado cualquier cantidad de días en la semana.

Estación 1:
    Producción dia: ---
    Producción dia: ---
        .
        .
    Producción dia: ---

Estación 2:
    Producción dia: ---
    Producción dia: ---
        .
        .
    Producción dia: ---
Estación 10:
    Producción dia: ---
    Producción dia: ---
        .
        .
    Producción dia: ---

Elaborar un algoritmo que lea el nivel de productividad, y los datos de la producción de cada una de las estaciones, e imprima el siguiente reporte:

REPORTE DE PRODUCCION

ESTACION	TOTAL PRODUCCION	NIVEL PRODUCTIVIDAD
1	999	DEFICIENTE
2	999	BUENO
3	-----	
.	-----	
.	-----	
10	999	EXCELENTE
TOTAL	999	

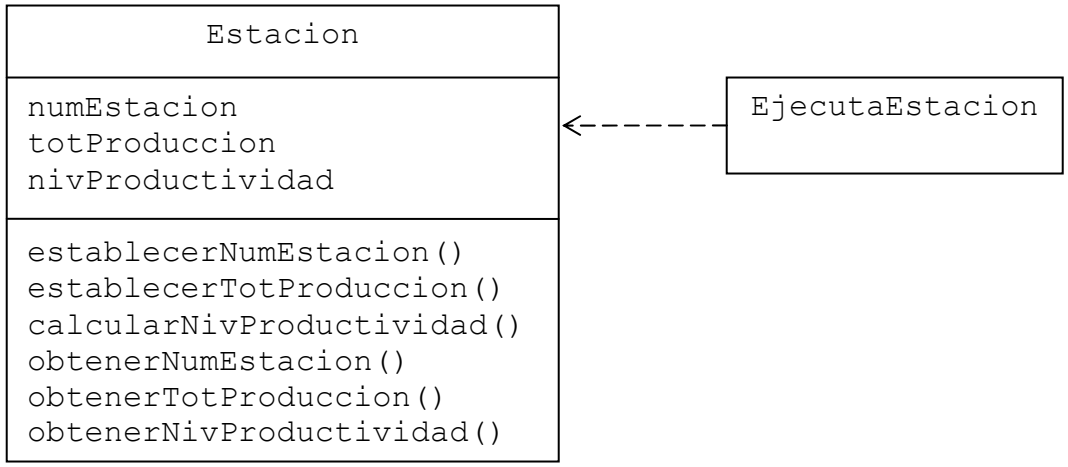
TOTAL PRODUCCION es la sumatoria de la producción de los días laborados.

NIVEL PRODUCTIVIDAD es un comentario que indica DEFICIENTE, BUENO o EXCELENTE si el TOTAL PRODUCCION es menor, es igual o es mayor a 300 respectivamente.
TOTAL es el total del TOTAL PRODUCCION de todas las estaciones de trabajo.

A continuación se presenta el algoritmo de la solución:

(Primero hágalo usted...después compare la solución)

Diagrama de clases




```
Algoritmo ESTACIONES DE TRABAJO
 Clase Estacion
  1. Declarar
       Datos
          numEstacion: Entero
          totProduccion: Entero
          nivProductividad: Cadena

  2. Método establecerNumEstacion(num: Entero)
     a. numEstacion = nom
     b. Fin Método establecerNumEstacion

  3. Método establecerTotProduccion(tProd: Entero)
     a. totProduccion = tProd
     b. Fin Método establecerTotProduccion

  4. Método calcularNivProductividad()
     a. if totProduccion < 300 then
           1. nivProductividad = "DEFICIENTE"
     b. else
           1. if totProduccion = 300 then
                 a. nivProductividad = "BUENO"
           2. else
                 a. nivProductividad = "EXCELENTE"
           3. endif
     c. endif
     d. Fin Método calcularNivProductividad
```

```
  5. Método obtenerNumEstacion() Entero
     a. return numEstacion
     b. Fin Método obtenerNumEstacion

  6. Método obtenerTotProduccion() Entero
     a. return totProduccion
     b. Fin Método obtenerTotProduccion

  7. Método obtenerNivProductividad() Cadena
     a. return nivProductividad
     b. Fin Método obtenerNivProductividad
 Fin Clase Estacion


 Clase EjecutaEstacion
  1. Método principal()
     a. Declarar
          Variables
            hay: Carácter
            estacion, proDia, totProd, toTotProd: Entero
     b. Imprimir encabezado
     c. toTotProd = 0
     d. for estacion=1; estacion<=10; estacion++
          1. Declarar, crear e iniciar objeto
             Estacion objEstacion = new Estacion()
          2. totProd = 0
          3. Preguntar "¿hay dia de produccion (S/N)?"
          4. Leer hay
          5. while hay == 'S'
               a. Solicitar Producción del día
               b. Leer proDia
               c. totProd = totProd + proDia
               d. Preguntar "¿hay dia de produccion (S/N)?"
               e. Leer hay
          6. endwhile
          7. Establecer
                objEstacion.establecerNumEstacion(estacion)
                objEstacion.establecerTotProduccion(totProd)
          8. Calcular objEstacion.calcularNivProductividad()
          9. Imprimir objEstacion.obtenerNumEstacion()
                      objEstacion.obtenerTotProduccion()
                      objEstacion.obtenerNivProductividad()
         10. toTotProd = toTotProd +
                         objEstacion.obtenerTotProduccion()
     e. endfor
     f. Imprimir toTotProd
     g. Fin Método principal
 Fin Clase EjecutaEstacion
Fin
```

En la zona de descarga de la Web del libro, está disponible:
Programa en Java: Estacion.java y EjecutaEstacion.java

Explicación:
El algoritmo tiene dos clases; la Clase Estacion y la clase EjecutaEstacion.
En la Clase Estacion:
1. Se declaran los datos que representan la estructura de la clase:
   numEstacion para el número que identifica a la estación
   totProduccion para la producción hecha por la estación
   nivProductividad para el nivel de productividad de la estación
2. Método establecerNumEstacion(num: Entero)
   Recibe en el parámetro num el valor que luego coloca en el dato numEstacion.
3. Método establecerTotProduccion(tProd: Entero)
   Recibe en el parámetro tProd el valor que luego coloca en el dato totProduccion.
4. Método calcularNivProductividad()
   a. Si totProduccion < 300 Entonces
      1. Coloca en nivProductividad "DEFICIENTE"
   b. Si no
      1. Si totProduccion = 300 Entonces
         a. Coloca en nivProductividad "BUENO"
      2. Si no
         a. Coloca en nivProductividad "EXCELENTE"
      3. Fin del if
   c. Fin del if
   d. Fin Método calcularNivProductividad
5. Método obtenerNumEstacion() Entero
   Retorna numEstacion
6. Método obtenerTotProduccion() Entero
   Retorna totProduccion
7. Método obtenerNivProductividad() Cadena
   Retorna nivProductividad

Fin de la Clase Estacion

En la Clase EjecutaEstacion; en el Método principal():
a. Se declara:
   La variable estacion para controlar el ciclo de estaciones de 1 a 10
   La variable proDia para leer la cantidad de unidades fabricadas por cada día
   La variable totProd para calcular el total de producción de cada estación
   La variable toTotProd para calcular el total de producción de todas las estaciones
   La variable hay para controlar el ciclo repetitivo
b. Imprime el encabezado
c. Inicia toTotProd en 0
d. Inicia for desde estacion=1 hasta 10 con incrementos de 1
   1. Se declara el objeto objEstacion, usando como base a la clase Estacion; dicho objeto se crea e inicializa mediante el constructor por defecto Estacion().
      Observe que cada vez que entra al ciclo crea un nuevo objeto empleado.
   2. Inicia totProd en 0
   3. Pregunta "¿Hay dia de produccion (S/N)?"

4. Lee en hay la respuesta
5. Inicia ciclo while mientras hay == ‘S’
    a. Solicita la Producción del día
    b. Lee en proDia
    c. Incrementa totProd con proDia
    d. Pregunta “¿Hay dia de produccion (S/N)?”
    e. Lee en hay la respuesta
6. Fin del while
7. Se llama al método establecerNumEstacion(estacion) del objeto objEstacion; para colocar el valor de estacion en el dato numEstacion.
   Se llama al método establecerTotProduccion(totProd) del objeto objEstacion; para colocar el valor de totProd en el dato totProduccion.
8. Se llama al método calcularNivProductividad() del objeto objEstacion; para calcular el nivel de productividad.
9. Se llama al método obtenerNumEstacion() del objeto objEstacion; para acceder e imprimir el valor del dato numEstacion.
   Se llama al método obtenerTotProduccion() del objeto objEstacion; para acceder e imprimir el valor del dato totProduccion.
   Se llama al método obtenerNivProductividad() del objeto objEstacion; para acceder e imprimir el valor del dato nivProductividad.
10. Se incrementa toTotProd con el totProduccion de la estación; mismo que se accede llamando al método obtenerTotProduccion() del objeto objEstacion.

e. Fin del for
f. Imprime toTotProd
g. Fin Método principal
Fin de la Clase EjecutaEstacion
Fin del algoritmo