**FUNDAMENTOS DE PROGRAMACIÓN**

**CON JAVA**

**PRÁCTICA 03**

**ESTRUCTURAS REPETITIVAS**

**Integrantes:** Llontop Solano, Juan Daniel

Medina Encalada, Alberto José

**Docente:** Coronel Castillo, Eric Gustavo

**INDICE**

1 Desarrollo de los proyectos 3

1.1 Proyecto 01 3

1.1.1 Problema 3

1.1.2 Análisis 3

1.1.3 Codificación 4

1.1.4 Pruebas de la solución 4

1.2 Proyecto 02 5

1.2.1 Problema 5

1.2.2 Análisis 5

1.2.3 Codificación 6

1.2.4 Pruebas de la solución 7

1.3 Proyecto 03 7

1.3.1 Problema 7

1.3.2 Análisis 7

1.3.3 Codificación 8

1.3.4 Pruebas de la solución 8

1.4 Proyecto 04 9

1.4.1 Problema 9

1.4.2 Análisis 9

1.4.3 Codificación 10

1.4.4 Pruebas de la solución 11

1.5 Proyecto 05 12

1.5.1 Problema 12

1.5.2 Análisis 12

1.5.3 Codificación 14

1.5.4 Pruebas de la solución 15

2 Conclusiones 16

3 Recomendaciones 16

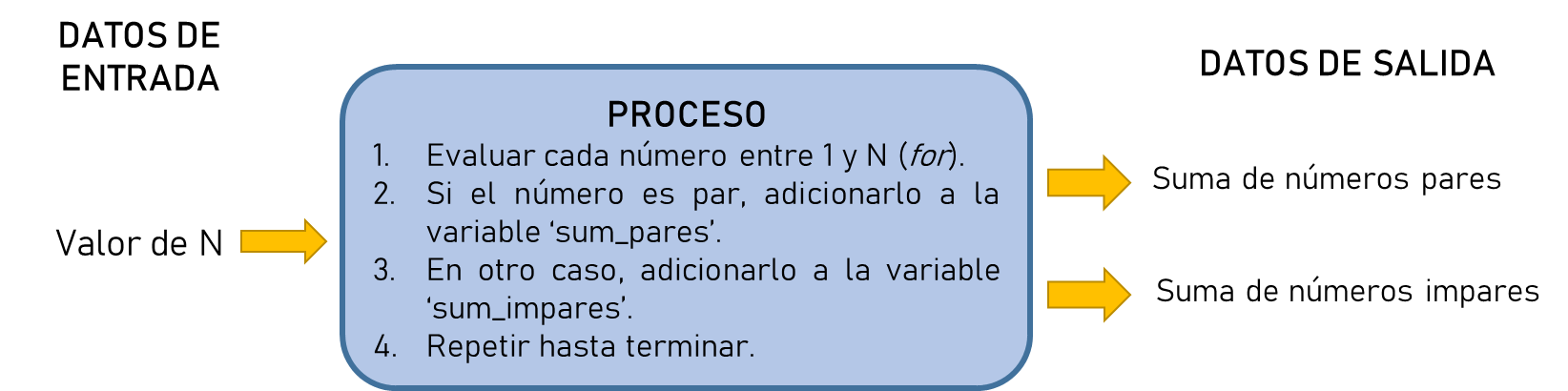
# Desarrollo de los proyectos

## Proyecto 01

### Problema

Desarrollar un programa para encontrar la suma de los números pares e impares comprendidos entre 1 y N.

### Análisis



En este problema, a partir de un valor natural de **N**, se requiere determinar la **suma de los números pares** y de los **números impares** hasta N. El proceso consiste en:

1. Inicializar las variables ‘sum\_pares’ y ‘sum\_impares’ con el valor de 0.
2. Evaluar todos los números desde 1 hasta N. Esto se consigue con el bucle *for*.

for (int i = 1 ; i <= N ; i++)

1. Si el número ‘i’ es par, adicionarlo a la variable ‘sum\_pares’. Si no, adicionarlo a la variable sum\_impares.
2. Repetir el proceso hasta llegar a i = N.

### Codificación

public static void main(String[] args) {

//Variables

int N, sum\_pares, sum\_impares;

Scanner sc = new Scanner(System.in);

//Lectura de datos

System.out.println("Ingrese el valor de N (número natural):");

N = sc.nextInt();

//Proceso

sum\_pares = 0;

sum\_impares = 0;

for (int i = 1; i <= N; i++) {

if (i % 2 == 0)

sum\_pares += i;

else

sum\_impares += i;

}

//Reporte

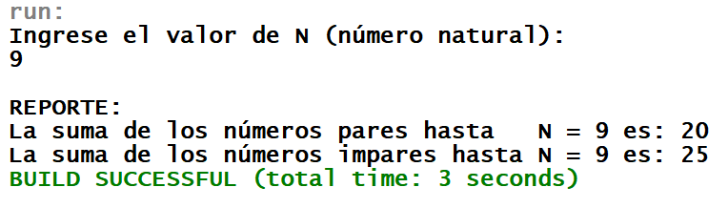
System.out.println("\nREPORTE: ");

System.out.println("La suma de los números pares hasta N = " + N + " es: " + sum\_pares);

System.out.println("La suma de los números impares hasta N = " + N + " es: " + sum\_impares);

}

### Pruebas de la solución

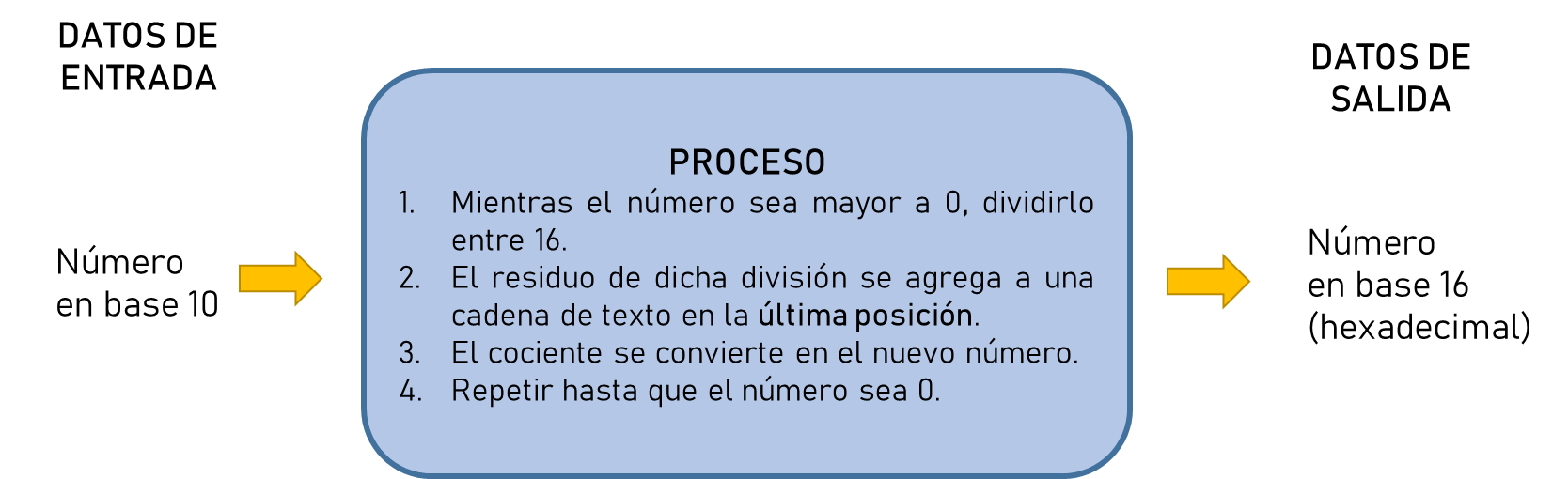


## Proyecto 02

### Problema

Desarrollar un programa para convertir un número de base 10 a base 16.

### Análisis



A partir de un **número en base decimal** se debe conseguir la conversión a **base hexadecimal**. Para efectuar la conversión, creamos un método llamado convertirHexadecimal() que toma como argumento el número (int) y genera el número hexadecimal de tipo String. El proceso de este método consiste en:

1. Generar una lista de las cifras en base hexadecimal.

char cifras[] = {'0', '1', '2', '3', '4', '5', '6', '7', '8',

'9', 'A', 'B', 'C', 'D', 'E', 'F'};

1. Inicializar una cadena de texto vacía para los residuos de la conversión.

String cadenaResiduos = "";

1. Mientras el número sea mayor a 0, dividirlo entre 16. El **residuo** se adiciona a la cadena anterior en su **última posición**. El **cociente** se convierte en el nuevo número.
2. Repetir el proceso hasta que el número sea 0.

### Codificación

public static void main(String[] args) {

//Variables

int numero;

String hexadecimal;

Scanner sc = new Scanner(System.in);

//Lectura de datos

System.out.println("Ingrese un número en base decimal:");

numero = sc.nextInt();

//Proceso

hexadecimal = convertirHexadecimal(numero);

//Reporte

System.out.println("\nEl número en base hexadecimal es:");

System.out.println(hexadecimal);

System.out.println("------Fin------");

}

private static String convertirHexadecimal(int n) {

char cifras[] = {'0', '1', '2', '3', '4', '5', '6', '7',

'8', '9', 'A', 'B', 'C', 'D', 'E', 'F'};

int residuo;

String cadenaResiduos = "";

while (n > 0) {

residuo = n % 16;

cadenaResiduos = cifras[residuo] + cadenaResiduos;

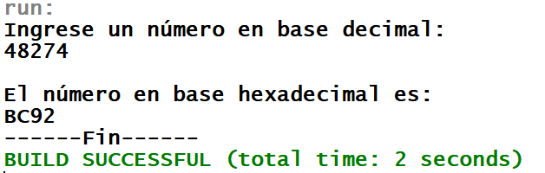
n = n / 16;

}

return cadenaResiduos;

}

### Pruebas de la solución

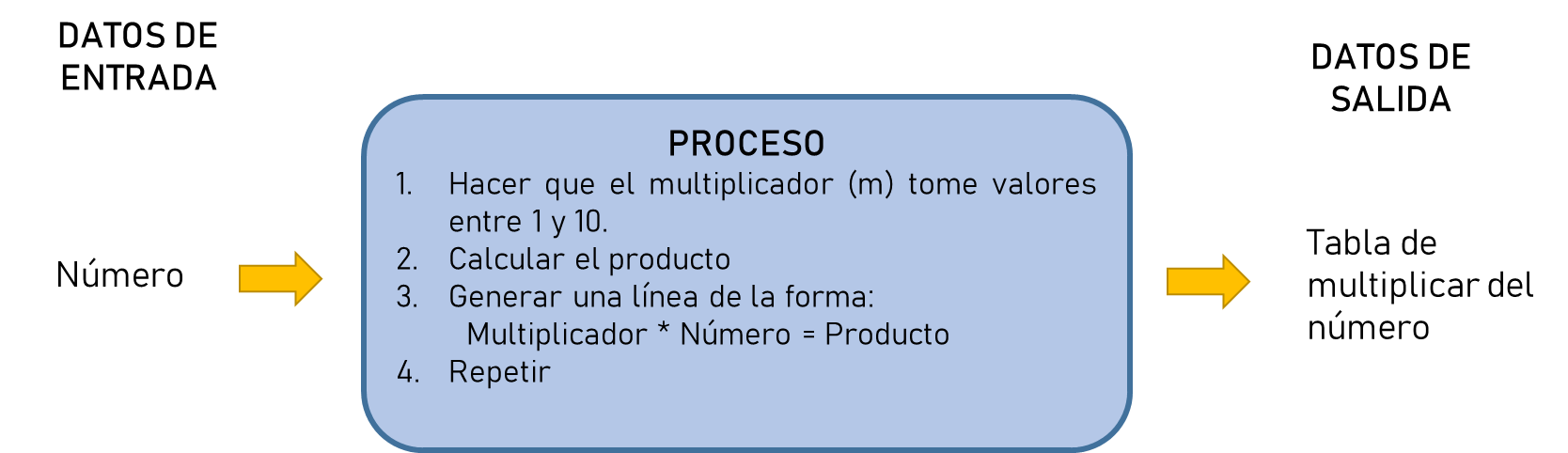
 

## Proyecto 03

### Problema

Desarrollar un programa que permita mostrar la tabla de multiplicar de un número.

### Análisis



Se requiere imprimir la **tabla de multiplicar** de un número, tomando como dato de entrada el valor entero del **número**. El proceso consiste en:

1. Inicializar un reporte vacío (String reporte = “”).
2. Hacer que el multiplicador (int m) tome valores entre 1 y 10.

for (int m = 1 ; m <= 10 ; m++ )

1. Calcular el producto: producto = m \* numero.
2. Adicionar al reporte una línea de la forma:

m \* numero = producto\n

1. Repetir el proceso hasta llegar a m = 10.

### Codificación

public static void main(String[] args) {

//Variables

int n, producto;

String reporte;

Scanner sc = new Scanner(System.in);

//Lectura de datos

System.out.println("Ingrese un número:");

n = sc.nextInt();

//Proceso

reporte = "";

for (int m = 1; m <= 10; m++) {

producto = n \* m;

reporte += String.format("%2d", m) + " x " + n + " = " + producto + "\n";

}

//Reporte

System.out.println("\nTABLA DE MULTIPLICAR DEL " + n);

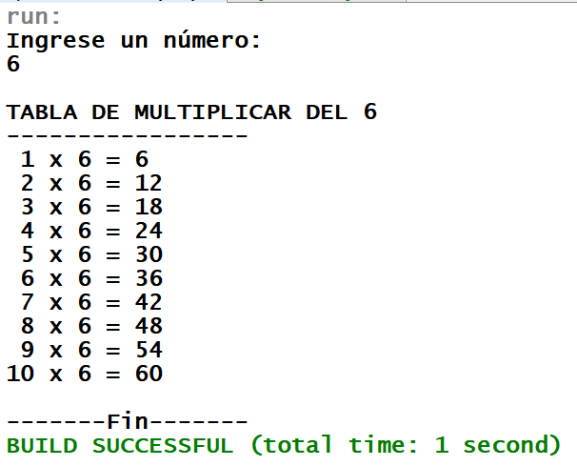
System.out.println("-----------------");

System.out.println(reporte);

System.out.println("-------Fin-------");

}

### Pruebas de la solución

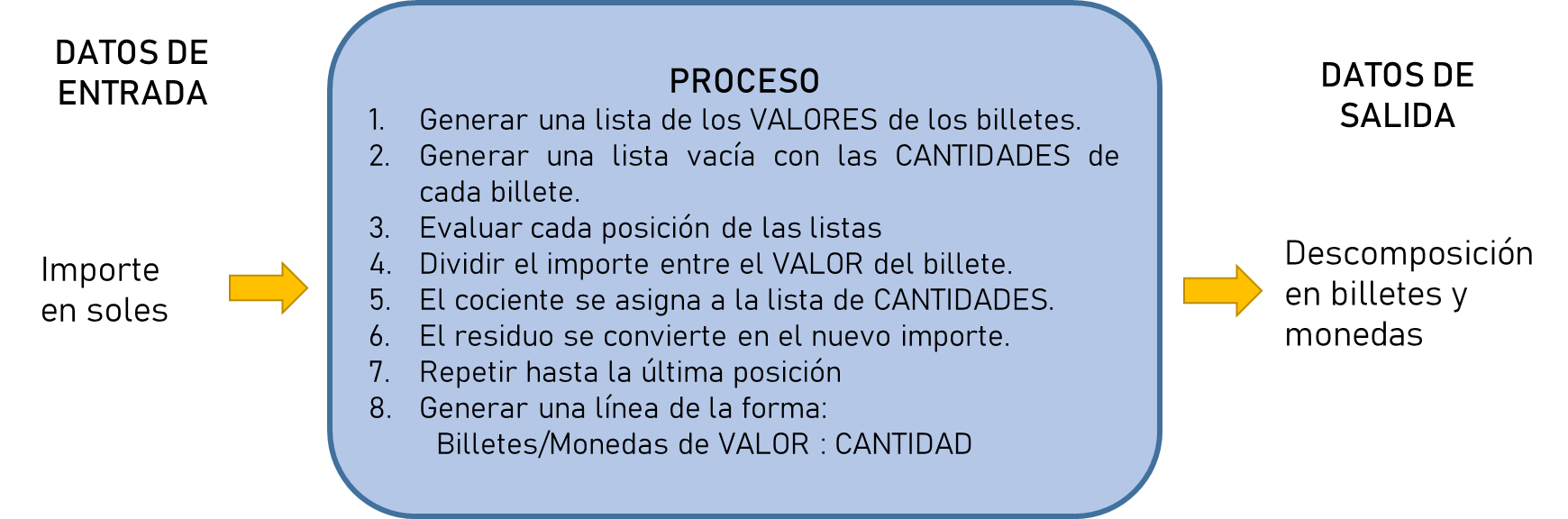


## Proyecto 04

### Problema

Desarrollar un programa que descomponga un importe de soles en billetes de 200, 100, 50, 20 y 10 Soles, el resto lo debe descompones en monedas 5, 2 y 1 sol

### Análisis



Se requiere descomponer un **importe monetario** en **billetes y monedas.** Para efectuar la descomposición, creamos un método llamado descomposicionMonetaria() que toma como argumento el importe (int) y genera el reporte de tipo String. El proceso de este método consiste en:

1. Generar una lista con los VALORES de los billetes y monedas:

int valoresBil[] = {200, 100, 50, 20, 10, 5, 2, 1};

1. Generar una lista vacía para contener las CANTIDADES de cada billete o moneda.

int cantidadBil[] = new int[8];

1. Evaluar cada posición de las listas con el bucle *for*.

for (int i = 0; i < 8; i++)

1. El importe se divide entre el VALOR correspondiente a la posición. El **cociente** de la división se introduce en la lista de CANTIDADES en la posición respectiva. El **residuo** se convierte en el nuevo importe. Repetir.

cantidadBil[i] = N / valoresBil[i];

N = N % valoresBil[i];

1. Inicializar un reporte vacío (String reporte = “”).
2. Evaluar cada posición de la lista de cantidades. Si la cantidad es 0, esta posición se ignora (**continue**). Adicionar al reporte una línea de la forma:

Billetes/Monedas de VALOR: CANTIDAD\n

### Codificación

public static void main(String[] args) {

//Variables

int importe;

String reporte;

Scanner sc = new Scanner(System.in);

//Lectura de datos

System.out.println("Ingrese el importe en soles:");

importe = sc.nextInt();

//Proceso

reporte = descomposicionMonetaria(importe);

//Reporte

System.out.println("\nREPORTE:");

System.out.println(reporte);

}

private static String descomposicionMonetaria(int N) {

int valoresBil[] = {200, 100, 50, 20, 10, 5, 2, 1};

int cantidadBil[] = new int[8];

for (int i = 0; i < 8; i++) {

cantidadBil[i] = N / valoresBil[i];

N = N % valoresBil[i];

}

String reporte = "";

for (int i = 0; i < 8; i++){

if (cantidadBil[i] == 0) {

continue;

}

if (valoresBil[i] >= 10)

reporte += "Billetes de " + valoresBil[i] + " : " + cantidadBil[i] + "\n";

else

reporte += "Monedas de " + valoresBil[i] + " : " + cantidadBil[i] + "\n";

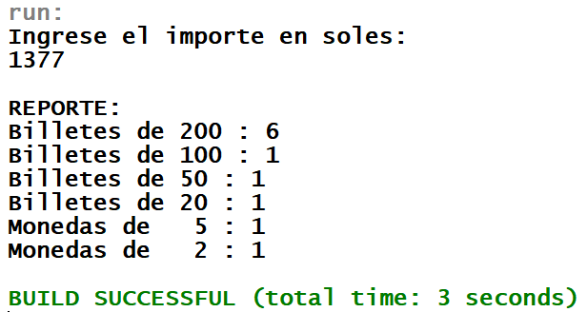
}

return reporte;

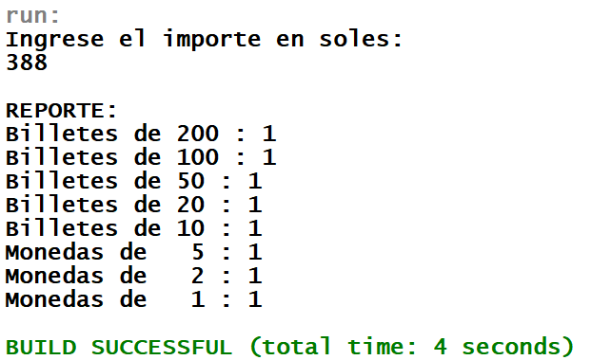
}

### Pruebas de la solución

* Prueba #1:



* Prueba #2:



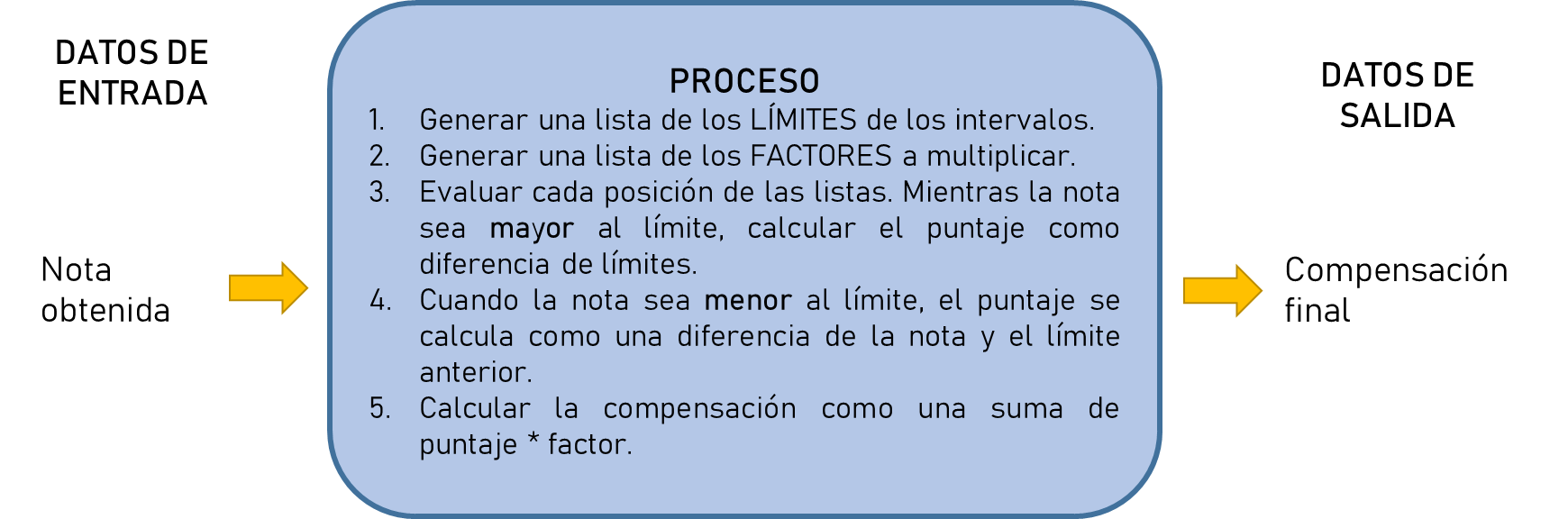
## Proyecto 05

### Problema

Un padre con la intención de motivar el estudio en su hijo, le dice que será compensado según su promedio final del ciclo. La compensación es monetaria, según el cuadro adjunto:



### Análisis



Se necesita calcular la **compensación monetaria** del hijo, teniendo en cuenta la **nota obtenida** y el cuadro superior. El cálculo que debe realizarse para obtener la compensación final es similar a este:

* Sean los límites: {0, 5, 12, 17, 20}
* Suponemos que la nota sea 13.
* Analizamos el primer intervalo (0 - 5). Como la nota es **mayor** al límite superior de este intervalo, el puntaje se obtiene como la diferencia de los límites:
  + El factor de este intervalo es:
  + La compensación de este intervalo será entonces:
* El razonamiento es similar en el segundo intervalo (5 - 12).
* En el tercer intervalo (12 - 17), la nota es **menor** al límite superior del intervalo. Por lo tanto, el puntaje se calcula como la diferencia entre la NOTA y el límite inferior:
  + El factor de este intervalo es:
  + La compensación de este intervalo será entonces:
* El proceso se detiene. Por lo tanto, la compensación total será:

Teniendo en cuenta este razonamiento, el proceso consiste en:

1. Generar una lista de los LÍMITES de los intervalos:

int limites[] = {0, 5, 12, 17, 20};

1. Generar una lista de los FACTORES de los intervalos:

int factores[] = {0, 50, 80, 120, 500};

1. Inicializar la compensación con el valor de 0. (compensación = 0)
2. Evaluar las listas desde la segunda posición. Mientras la nota sea **mayor** al límite, el puntaje se calcula como la diferencia del límite actual menos el anterior:

int i = 1;

while (nota > limites[i])

puntaje = limites[i] - limites[i - 1];

factor = factores[i];

1. La compensación se calcula como el producto del puntaje y el factor. Adicionamos este valor a la variable correspondiente:

compensacion += puntaje \* factor;

1. El bucle termina cuando la nota sea **menor** al límite. En este punto, el puntaje se calcula como la diferencia de la nota con el límite anterior.

puntaje = nota - limites[i - 1];

factor = factores[i];

1. Adicionamos la compensación para hallar el valor final:

compensacion += puntaje \* factor;

### Codificación

public static void main(String[] args) {

//Variables

int nota, puntaje, factor, compensacion;

int limites[] = {0, 5, 12, 17, 20};

int factores[] = {0, 50, 80, 120, 500};

String reporte;

Scanner sc = new Scanner(System.in);

//Lectura de datos

System.out.println("Ingrese la nota obtenida:");

nota = sc.nextInt();

//Proceso

reporte = "";

compensacion = 0;

int i = 1;

while (nota > limites[i]) {

puntaje = limites[i] - limites[i - 1];

factor = factores[i];

compensacion += puntaje \* factor;

reporte += puntaje + " \* " + factor + " = " + (puntaje \* factor) + " +\n";

i++;

}

puntaje = nota - limites[i - 1];

factor = factores[i];

compensacion += puntaje \* factor;

reporte += puntaje + " \* " + factor + " = " + (puntaje \* factor);

//Reporte

System.out.println("\nREPORTE");

System.out.println("-----------------------");

System.out.println(reporte);

System.out.println("-----------------------");

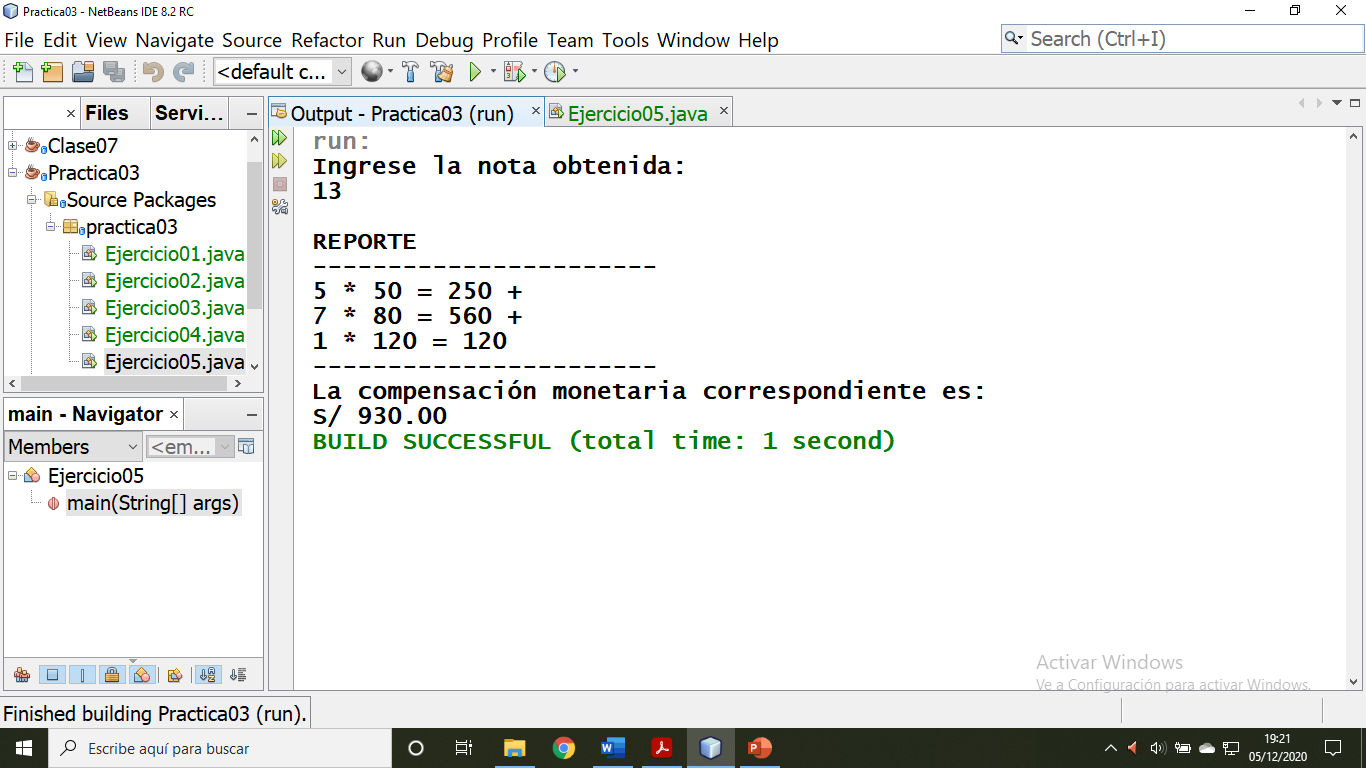
System.out.println("La compensación monetaria correspondiente es:");

System.out.println("S/ " + compensacion + ".00");

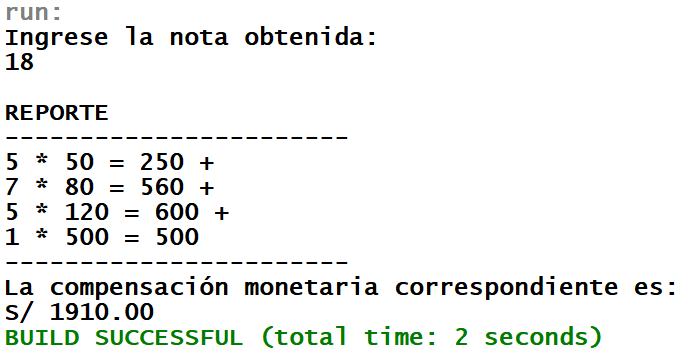
}

### Pruebas de la solución

* Prueba #1:



* Prueba #2:



# Conclusiones

En esta sesión se aprendió a utilizar las estructuras repetitivas o bucles, especialmente las estructuras *for* y *while*, para ejecutar líneas de código mientras se cumpla una cierta condición lógica. La utilidad de estas estructuras se incrementa al utilizar arreglos, ya que los bucles pueden recorrer los elementos de estos arreglos en paralelo utilizando sus índices.

# Recomendaciones

Se recomienda el análisis previo (en papel) de los problemas que involucran bucles, ya que esto facilita la abstracción de las principales partes del proceso. Asimismo, la creación de métodos privados en la misma clase permite estructurar mejor el programa y posibilita una mejor comprensión.