|  |  |  |
| --- | --- | --- |
| Texto  Descripción generada automáticamente con confianza media | TECNICATURA UNIVERSITARIA EN DISEÑO INTEGRAL DE VIDEOJUEGOS  FACULTAD DE INGENIERÍA  Universidad Nacional de Jujuy |  |

*Profesor:*

*Mg. Ing. Ariel Alejandro Vega*

*Año 2024*

Trabajo Práctico N°1

Angelo Carla TUV000563

**FUNDAMENTOS DE PROGRAMACIÓN ORIENTADA A OBJETOS**

Indice

**Trabajo Práctico N°1: Operadores – Metodología de Programación**

*Sección Expresiones aritméticas y lógicas*

**Ejercicio 1:** Evaluar (obtener resultado) la siguiente expresión para A = 2 y B = 5

**3\* A - 4 \* B / A ^ 2**

**Resolución:**

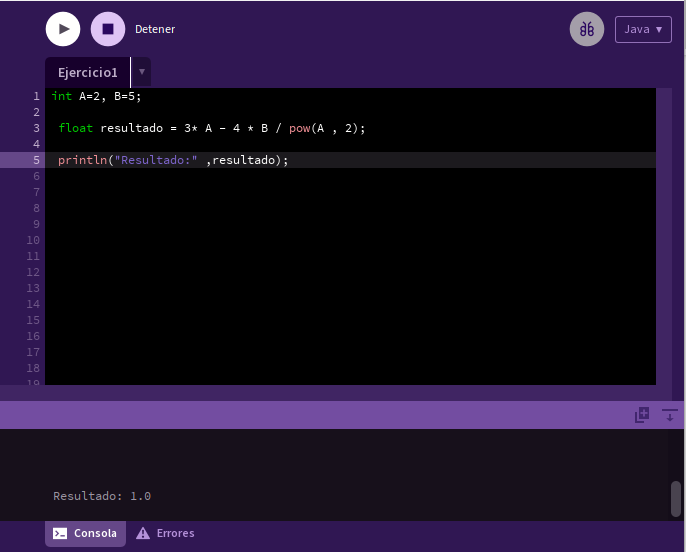
( 3\* 2 ) – ( 4 \* 5 / 2 ^ 2 )

6 - ( 20 / 4 )

6 – 5

**1**

**Processing:**



**Ejercicio 2:** Evaluar la siguiente expresión

**4 / 2 \* 3 / 6 + 6 / 2 / 1 / 5 ^ 2 / 4 \* 2**

**Resolución:**

( 4 / 2 \* 3 / 6 ) + ( 6 / 2 / 1 / 5 ^ 2 / 4 \* 2 )

( 2 \* 3 / 6 ) + ( 6 / 2 / 1 / 25 / 4 \* 2)

( 6 / 6 ) + ( 3 / 1 / 25 / 4 \* 2 )

1 + ( 3 / 25 / 4 \* 2 )

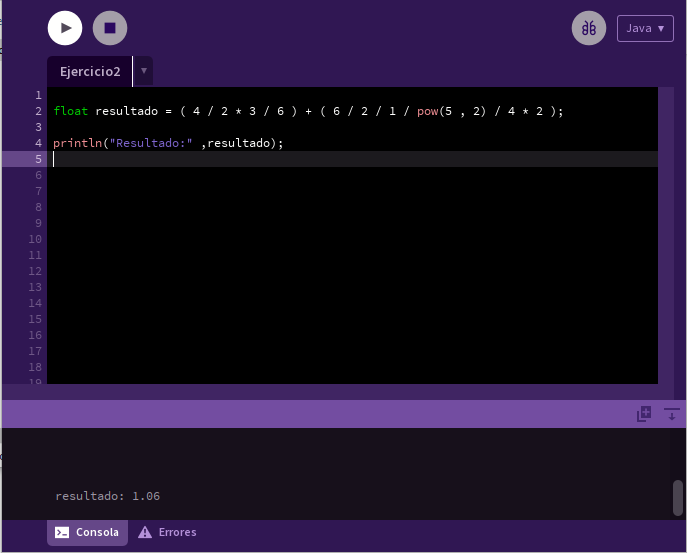
1 + ( 0.12 / 4 \* 2 )

1 + ( 0.03 \* 2 )

1 + 0.06

**1.06**

**Processing:**

****

**Ejercicio 3:** Evaluar las siguientes expresiones aritméticas, para lo cual indicar en el caso de las variables, el valor indicado. Luego escribirlas como expresiones algebraicas.

*a=1, b=2, c=3, d=4, x=2, y=1*

a) b ^ 2 – 4 \* a \* c

b) 3 \* X ^ 4 – 5 \* X ^ 3 + X 12 – 17

c) (b + d) / (c + 4)

d) (x ^ 2 + y ^ 2) ^ (1 / 2)

**Resolución:**

1. b ^ 2 – 4 \* a \* c

( 2 ^ 2 ) − ( 4 \* 1\* 3 )

4 – ( 4 \* 3 )

4 – 12

**-8**

***𝑏² − 4. 𝑎. c***

1. 3 \* X ^ 4 – 5 \* X ^ 3 + X 12 – 17

( 3 \* 2 ^ 4 ) – ( 5 \* 2 ^ 3 ) + ( 2 \* 12 ) – 17

( 3 \* 16 ) – ( 5 \* 8 ) + ( 2 \* 12 ) – 17

48 – 40 + 24 – 17

8 + 24 – 17

32 – 17

**15**

***3x ⁴ - 5x ³ + x12 – 17***

1. (b + d) / (c + 4)

( 2 + 4 ) / ( 3 + 4 )

6 / 7

**o.857**

1. (x ^ 2 + y ^ 2) ^ (1 / 2)

( 2 ^ 2 + 1 ^ 2) ^ ( 1 / 2 )

( 4 + 1 ) ^ ( 1 / 2 )

5 ^ ( 1 / 2 )

**2.236**

**Processing:**

****

**Ejercicio 4:** Si el valor de A es 4, el valor de B es 5 y el valor de C es 1, evaluar las siguientes expresiones:

a) B \* A – B ^ 2 / 4 \* C

b) (A \* B) / 3 ^ 2

c) (((B + C) / 2 \* A + 10) \* 3 \* B) – 6

**Resolución:**

a) ( 5 \* 4 ) – ( 5 ^ 2 / 4 \* 1 )

20 – ( 25 / 4 \* 1 )

20 – ( 6.25 \* 1 )

20 – 6.25

**13.75**

b) ( 4 \* 5 ) / 3 ^ 2

20 / 3 ^ 2

20 / 9

**2.2222**

c) ( ( ( 5 + 1 ) / 2 \* 4 + 10 ) \* 3 \* 5 ) – 6

( (6 / 2 \* 4 + 10 ) \* 3 \* 5 ) – 6

( ( 3 \* 4 + 10 ) \* 3 \* 5 ) – 6

( ( 12 + 10 ) \* 3 \* 5 ) – 6

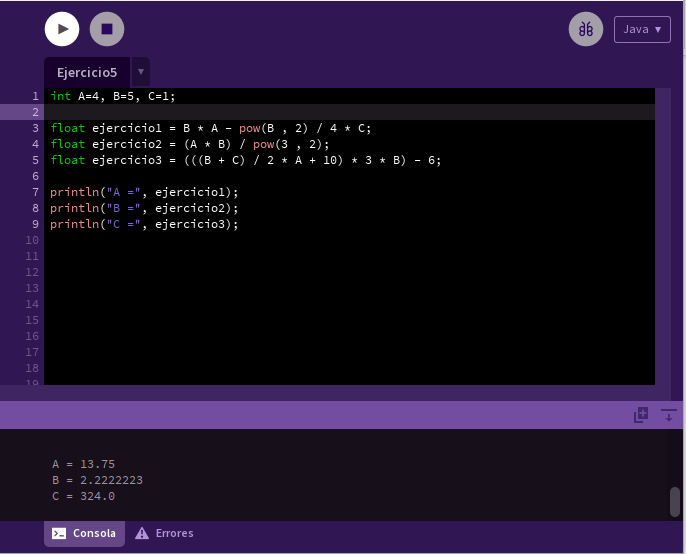
( 22 \* 3 \* 5 ) – 6

66 \* 5 – 6

330 – 6

**324**

**Processing:**



**Ejercicio 5:** Para x=3, y=4; z=1, evaluar el resultado de

R1 = y+z

R2 = x >= R1

**Resolución:**

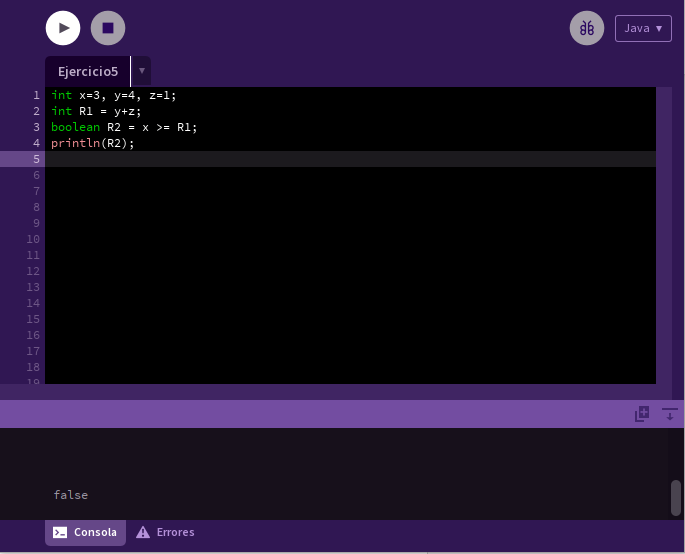
R1 = 4+1

R1 = 5

R2 = 3>= 5

R1 es igual a 5, 3 no es igual o mayor a 5 por lo tanto es **FALSO**

**Processing:**



**Ejercicio 6:** Para contador1=3, contador2=4, evaluar el resultado de

R1 = ++contador1

R2 = contador1 < contador2

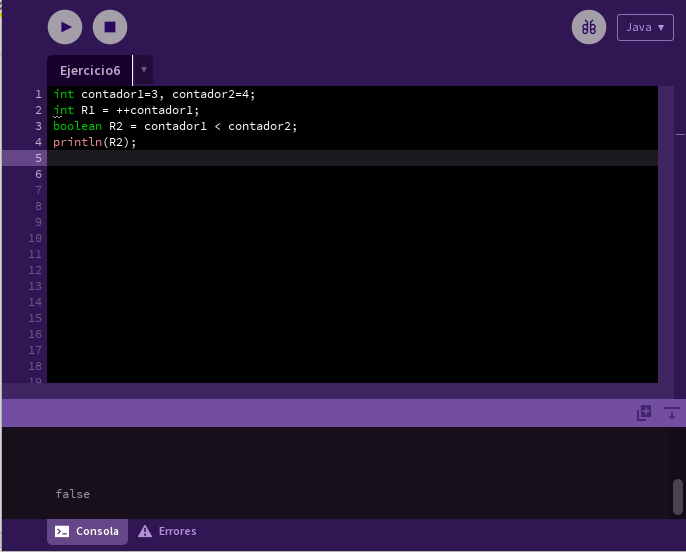
**Resolución:**

R1 = ++3

R2 = 4<4

En R1 se incrementa 1 a 3 lo cual es 4, R2 es igual a 4, 4 no es menor a 4 por lo tanto es **FALSO**

**Processing:**



**Ejercicio 7:** Para a=31, b=-1; x=3, y=2, evaluar el resultado de

a+b-1 < x\*y

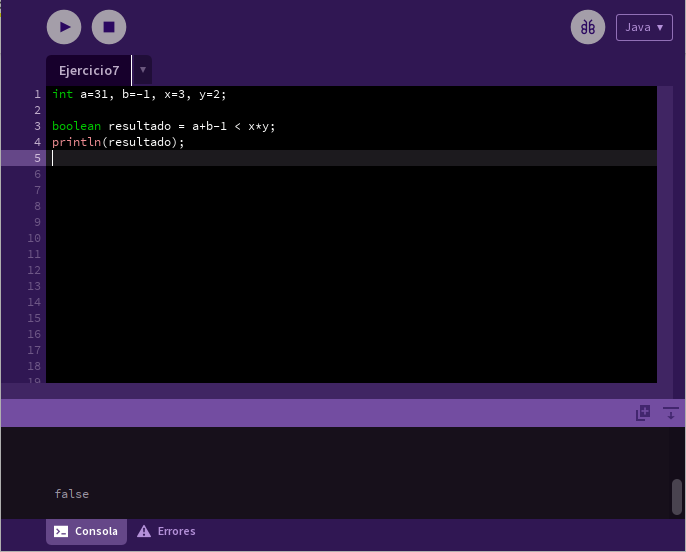
**Resolución:**

31+(-1)-1 < 3\*2

29 < 6

29 no es menor que 6 por lo tanto es **FALSO**

**Processing:**



**Ejercicio 8:** Para x=6, y=8, evaluar el resultado de

!(x<5)&& !(y>=7)

**Resolución:**

!(6<5)&& !(8>=7)

!(FALSO)&& !(VERDADERO)

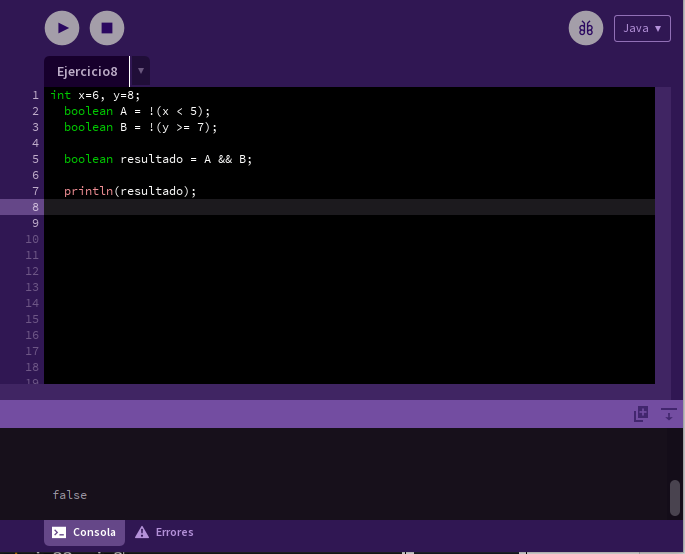
(VERDADERO) && (FALSO)

**FALSO**

*“!(6<5)”* 6 no es menor a 5, por lo tanto es **FALSO**. Con el operador *“!”*, el resultado se convierte en **VERDADERO** y *“!(8>=7)”* 8 si es mayor o igual a 7, por lo tanto es **VERDADERO**. Con el operador *“!”*, el resultado se convierte en **FALSO**.

*!(6<5) && !(8>=7)* se unen ambas expresiones con *“&&”* y por lo tanto **VERDADERO** && **FALSO** resulta **FALSO**

**Processing:**



**Ejercicio 9:** Para i=22,j=3, evaluar el resultado de

!((i>4) || !(j<=6))

**Resolución:**

!((22>4) || !(3<=6))

!((VERDADERO) || !(VERDADERO)

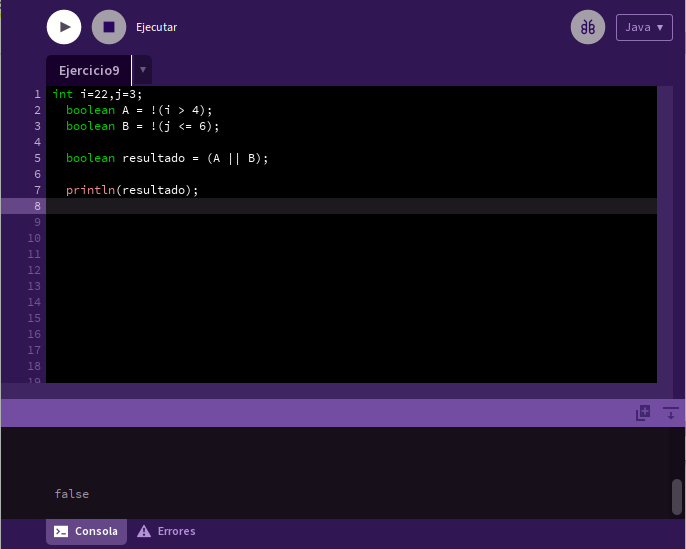
(FALSO)||(FALSO)

**FALSO**

*“!(22>4)”* 22 es mayor a 4 por lo tanto es **VERDADERO**. Con *“!”* el **VERDADERO** se convierte en FALSO. *“!(3<=6)”* 3 es menor o igual a 6 por lo tanto es VERDADERO. Con *“!”* el **VERDADERO** se convierte en **FALSO**

*!((22>4) || !(3<=6))* se unen ambas expresiones con “||” por lo tanto **FALSO||FALSO** es **FALSO**

**Processing:**



**Ejercicio 10:** Para a=34, b=12,c=8, evaluar el resultado de

!(a+b==c) || (c!=0)&& (b-c>=19)

**Resolución:**

!(34+12==8) || (8!=0)&& (12-8>=19)

!(46==8) || (8!=0)&& (4>=19)

!(FALSO) || (VERDADERO)&& (VERDADERO)

!(FALSO) || (VERDADERO)

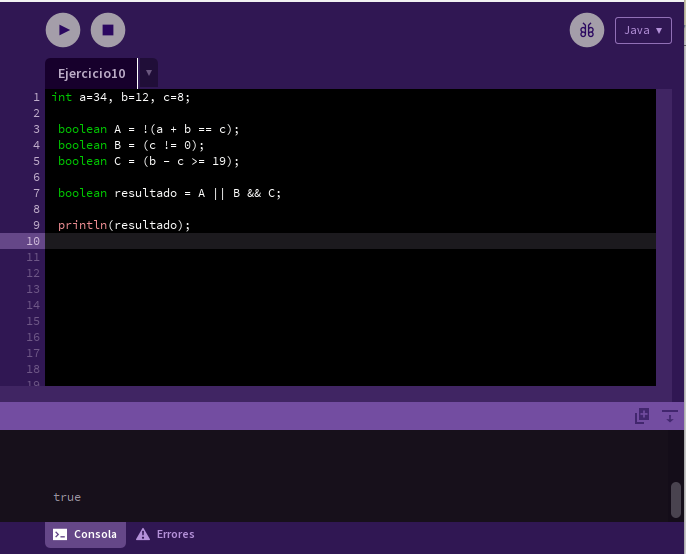
(VERDADERO)||(VERDADERO)

**VERDADERO**

*“!(34+12==8)”*  34 + 12 es igual a 46 y 46 no es igual a 8 por lo tanto seria **FALSO**. Con *“!”* el **FALSO** se convierte en **VERDADERO.** *“(8!=0)”* es distinto 8 de 0 por lo tanto es **VERDADERO. *“****(12-8>=19)”* 12 – 8 es 4 y 4 es menor o igual a 19 por lo tanto es **VERDADERO.**

*“!(34+12==8) || (8!=0)&& (12-8>=19)”* La segunda y tercera se unen con *“&&”* por lo tanto **VERDADERO**&&**VERDADERO** es **VERDADERO** y la primer expresión se une con *“||”* por lo tanto **VERDADERO**||**VERDADERO** es **VERDADERO**

**Processing:**



*Sección Análisis – Diseño y Codificación de algoritmos – Aplicación de estructuras de control*

**Ejercicio 11:** Un problema sencillo. Deberá pedir por teclado al usuario un nombre y posteriormente realizará la presentación en pantalla de un saludo con el nombre indicado

**Resolución:**

*Definición del problema:* A partir de un nombre, saludara con el nombre indicado

**Análisis:**

*Datos de Entrada:* Un nombre

*Datos de Salida:* Un saludo con el nombre indicado

*Proceso:*

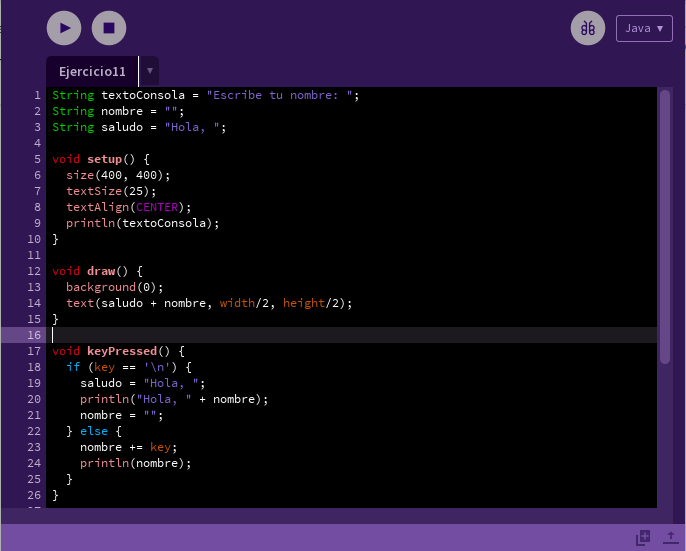
*¿Quién debe realizar el proceso?:* El algoritmo

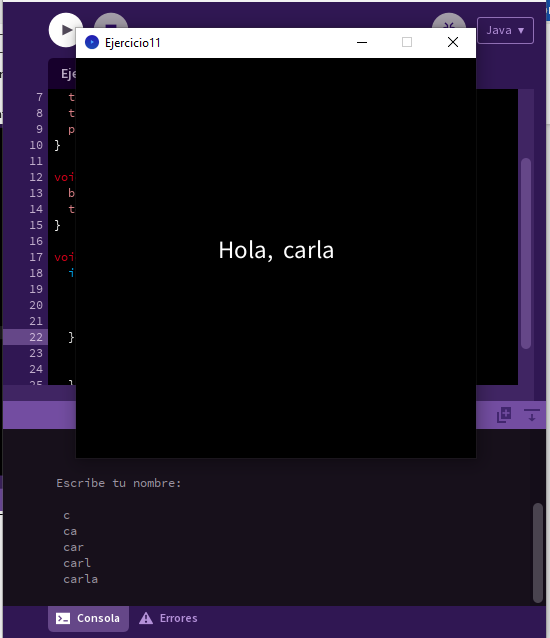
*¿Cuál es el proceso que resuelve?:* se ingresa un nombre y se saluda con el nombre indicado

**Diseño:**

|  |
| --- |
| ***ENTIDAD QUE RESUELVE EL PROBLEMA:*** El algoritmo |
| ***VARIABLES:***  • textoConsola, nombre, saludo: string |
| ***NOMBRE ALGORITMO:*** saludo\_nombre  ***PROCESO DEL ALGORITMO:***   1. *Inicio* 2. *textoConsola⭠ “Escribe tu nombre”* 3. *nombre ⭠ “”* 4. *saludo ⭠ “Hola, “* 5. *mostrar textoConsola* 6. ***si*** *(key == '\n')* ***entonces*** 7. *Saludo ⭠ “hola, “* 8. *Mostrar “hola, “ + nombre”* 9. *Nombre ⭠ “”* 10. ***Demás*** 11. *Nombre ⭠ += key* 12. *Mostrar nombre* 13. ***fin\_si*** 14. *fin* |

**Processing:**

****



**Ejercicio 12:** Será común resolver problemas utilizando variables. Calcule el perímetro y área de un rectángulo dada su base y su altura.

**Resolución:**

*Definición del problema:* Calcular el perímetro y el área de un rectángulo dada su base y altura

**Análisis:**

*Datos de Entrada:* La base y la altura

*Datos de Salida:* El perímetro y el Área

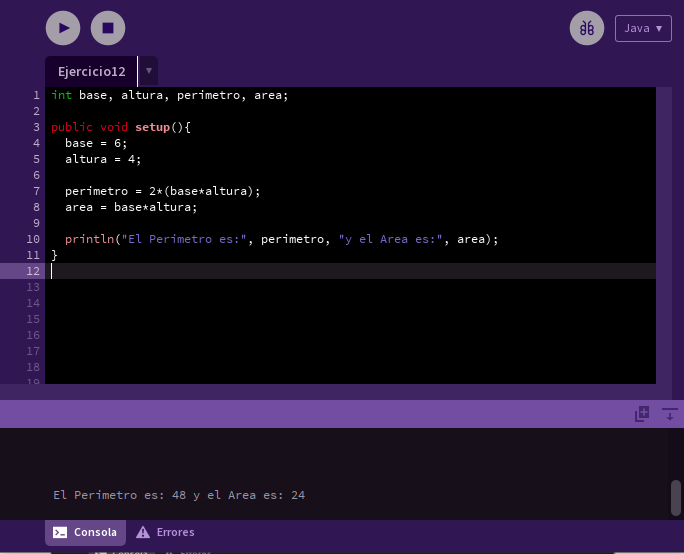
*Proceso:*

*¿Quién debe realizar el proceso?:* El algoritmo

¿Cuál es el proceso que resuelve?: calcula el perímetro y el área de un rectángulo utilizando las formulas

**Diseño:**

|  |
| --- |
| ***ENTIDAD QUE RESUELVE EL PROBLEMA:*** El algoritmo |
| ***VARIABLES:***  • base, altura, perímetro, area: int |
| ***NOMBRE ALGORITMO:*** perímetro\_area\_rectangulo  ***PROCESO DEL ALGORITMO:***   1. *Inicio* 2. *base ⭠ 6 //Ejemplo base* 3. *altura ⭠ 2 //Ejemplo altura* 4. *perimetro⭠ 2\*(base\*altura)* 5. *area⭠ base\*altura* 6. *mostrar “El Perimetro es:” + perímetro + “y el Area es:” + área* 7. *fin* |

**Processing:**

**Ejercicio 13:** Una ayuda importante al momento de resolver problemas con algoritmos es asumir que su gran amigo son las matemáticas. Obtenga la hipotenusa de un triángulo rectángulo conociendo sus catetos

**Resolución:**

*Definición del problema:* Obtener la hipotenusa de un triángulo rectángulo

**Análisis:**

*Datos de Entrada:* El valor de los catetos del triángulo rectángulo

*Datos de Salida:* La hipotenusa del triángulo rectángulo

*Proceso:*

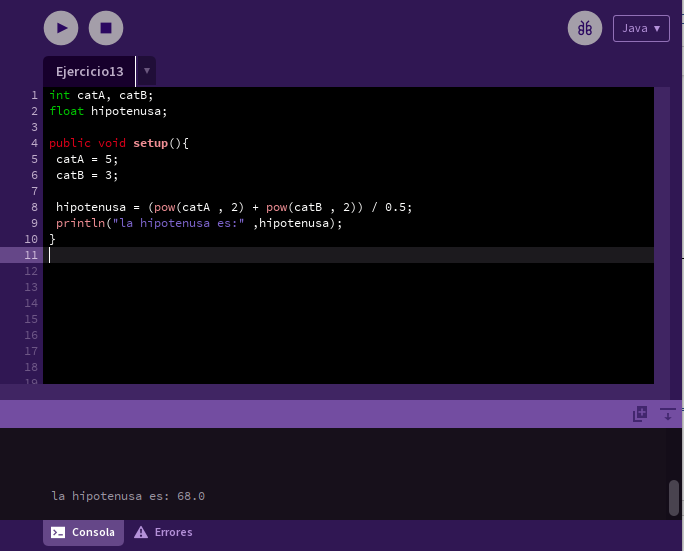
*¿Quién debe realizar el proceso?:* El algoritmo

¿Cuál es el proceso que resuelve?: Se ingresan los valores de ambos catetos del triángulo y se resuelve mediante la formula para obtener la hipotenusa

**Diseño:**

|  |
| --- |
| ***ENTIDAD QUE RESUELVE EL PROBLEMA:*** El algoritmo |
| ***VARIABLES:***  • catA, catB: int  • hipotenusa: float |
| ***NOMBRE ALGORITMO:*** hipotenusa\_triangulo  ***PROCESO DEL ALGORITMO:***   1. *inicio* 2. *catA⭠5* 3. *catB⭠3* 4. *hipotenusa⭠ (a^2 + b^2 ) ^(0.5)* 5. *mostrar hipotenusa* 6. *fin* |

**Processing:**

****

**Ejercicio 14:** Si viste algo de los apuntes y vídeos, esto debería ser muy fácil de resolver. Dados dos números permita calcular la suma, resta, multiplicación y división de estos. Considere que cada una de estas operaciones es un algoritmo cuando realice el diseño. Obviamente muestre los resultados

**Resolución:**

*Definición del problema:* Dados dos números, calcular la suma, resta, multiplicación y división de los mismos

**Análisis:**

*Datos de Entrada:* Dos números

*Datos de Salida:* Resultado de cada operación matemática

*Proceso:*

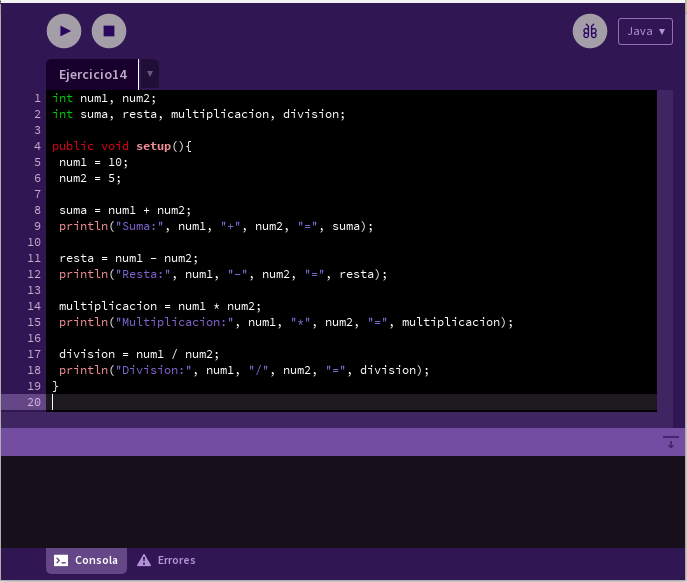
*¿Quién debe realizar el proceso?:* El algoritmo

¿Cuál es el proceso que resuelve?: Se ingresan dos números y se realizan los cálculos correspondientes para cada operación, en este caso suma, resta, multiplicación y división

**Diseño:**

|  |
| --- |
| ***ENTIDAD QUE RESUELVE EL PROBLEMA:*** El algoritmo |
| ***VARIABLES:***  • num1, num2: int  • suma, resta, multiplicación, división: int |
| ***NOMBRE ALGORITMO:*** operaciones\_matematicas  ***PROCESO DEL ALGORITMO:***   1. *inicio* 2. *num1⭠ 10* 3. *num2⭠ 5* 4. *suma⭠ num1 + num2* 5. *mostrar "Suma:", num1, "+", num2, "=", suma* 6. *resta⭠ num1 – num2* 7. *mostar "Resta:", num1, "-", num2, "=", resta* 8. *multiplicacion⭠ num1 \* num2* 9. *mostrar "Multiplicacion:", num1, "\*", num2, "=", multiplicacion* 10. *division⭠ num1 / num2* 11. *mostrar "Division:", num1, "/", num2, "=", division* 12. *fin* |

**Processing:**

****

**Ejercicio 15:** Necesitamos convertir una temperatura Fahrenheit en grados Celsius. Si no conoce la forma en la que se realiza esta conversión, debería investigarlo; para eso sirve la etapa de análisis. Pero como somos buenos, daremos una ayuda

**Resolución:**

*Definición del problema:* Pasar de una temperatura Fahrenheit en grados Celsius

**Análisis:**

*Datos de Entrada:* Grados Fahrenheit

*Datos de Salida:* Grados Celsius

*Proceso:*

*¿Quién debe realizar el proceso?:* El algoritmo

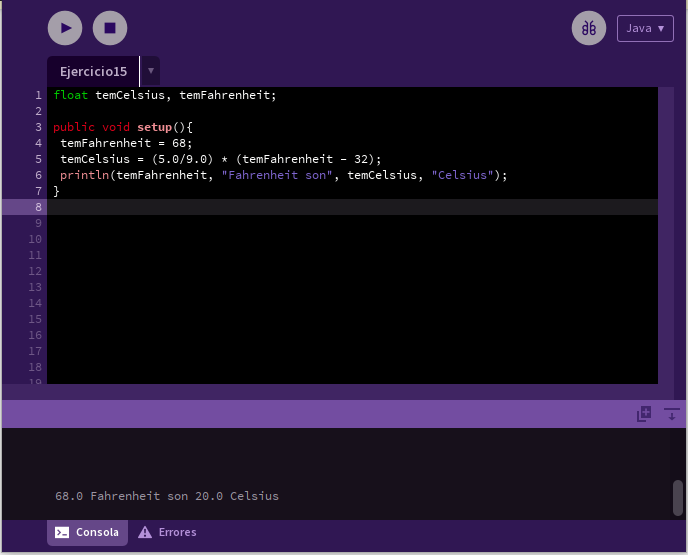
¿Cuál es el proceso que resuelve?: Convertir, mediante la formula C=95​×(F−32),

, de gados Fahrenheit a grados Celsius.

**Diseño:**

|  |
| --- |
| ***ENTIDAD QUE RESUELVE EL PROBLEMA:*** El algoritmo |
| ***VARIABLES:***  • temCelsius, temFahrenheit: int |
| ***NOMBRE ALGORITMO***: convertir\_Fahrenheit\_Celsius  ***PROCESO DEL ALGORITMO:***   1. *inicio* 2. *temFahrenheit ⭠ 68* 3. *temCelsius ⭠ (5.0/9.0) \* (temFahrenheit - 32)* 4. *mostrar temFahrenheit, "Fahrenheit son", temCelsius, "Celsius"* 5. *fin* |

**Processing:**



**Ejercicio 16:** Si queremos representar personajes o power ups (premios) en la pantalla debemos primero ubicarlos en alguna posición dentro de la pantalla. Imagine que está en un juego donde un power up desaparece porque el personaje se acerca a una distancia de x unidades, sin importar por donde se acerque. Por tanto, para que desaparezca, en primer lugar, hay que determinar esa distancia. La forma de representar la posición de un objeto en la pantalla es a través de las coordenadas de un punto. Suponga que la posición de Link está representada por la coordenada (𝑥1, 𝑦1) , mientras que las de la caja de tesoro se halla en la posición (𝑥2, 𝑦2). Si observa con detenimiento se observa la conformación de un triángulo rectángulo, por lo que es posible aplicar Pitágoras para obtener la distancia.

Para esto debe calcular el tamaño de los catetos y luego aplicar el teorema. Halle la distancia entre ambos objetos. Cuando programe, represente a lLnk con un Circulo, y al tesoro con un cuadrado. Además, mueva a Link mediante el mouse.

**Resolución:**

*Definición del problema:* Calcular el tamaño de los catetos y luego aplicar Teorema de Pitágoras y hallar la distancia entre Link y el tesoro.

**Análisis:**

*Datos de Entrada:* Las coordenadas de Link y el Tesoro

*Datos de Salida:*

*Proceso:*

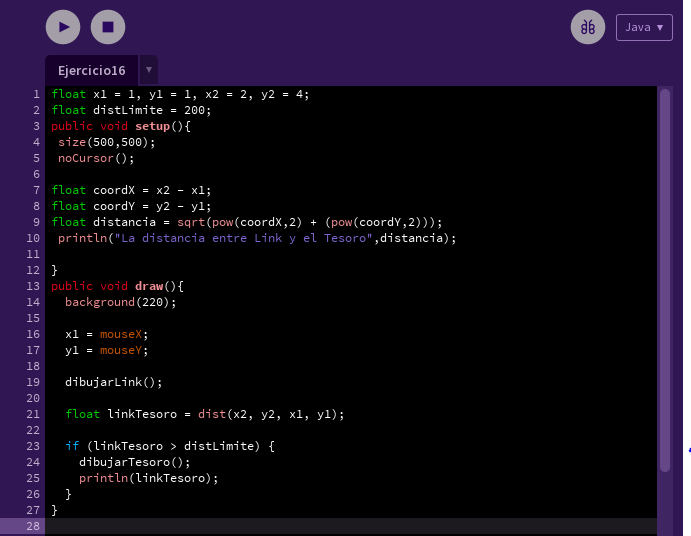
*¿Quién debe realizar el proceso?:* El algoritmo

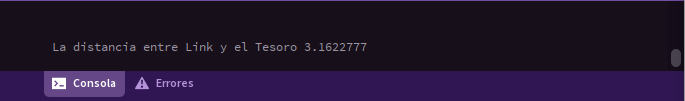
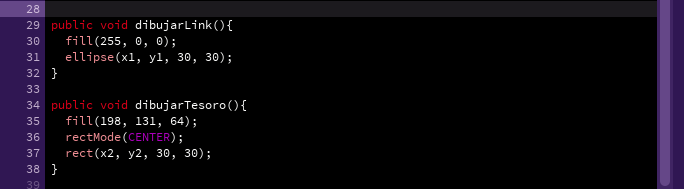
¿Cuál es el proceso que resuelve?:

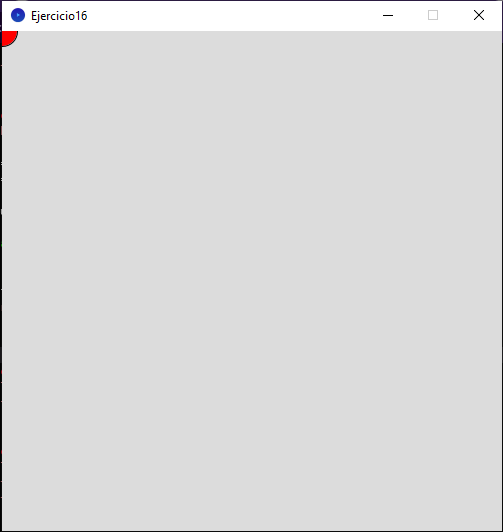
**Diseño:**

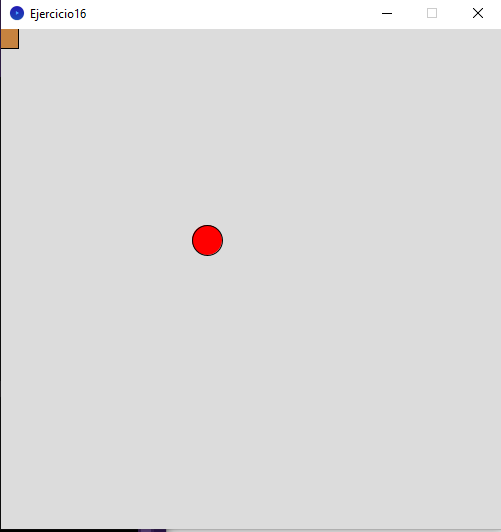
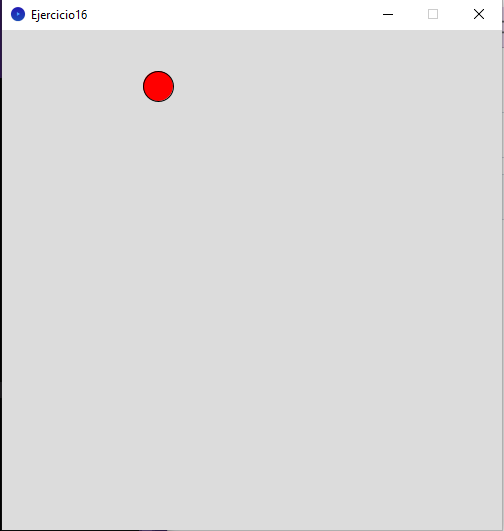
|  |
| --- |
| ***ENTIDAD QUE RESUELVE EL PROBLEMA:*** El algoritmo |
| ***VARIABLES:***  • |
| ***NOMBRE ALGORITMO:***  ***PROCESO DEL ALGORITMO:***  *⭠* |

**Processing:**









**Ejercicio 17:** Desarrolle el análisis y diseño de un algoritmo que permita obtener las raíces de una ecuación de segundo grado. Además, utilice la estructura según para el análisis de la discriminante de la ecuación cuadrática. Obviamente codifique en Processing

**Resolución:**

*Definición del problema:* Un algoritmo que permita obtener las raíces de una ecuación de segundo grado

**Análisis:**

*Datos de Entrada:* Los valores de a, b y c

*Datos de Salida:* Las raíces de la ecuación cuadrática

*Proceso:*

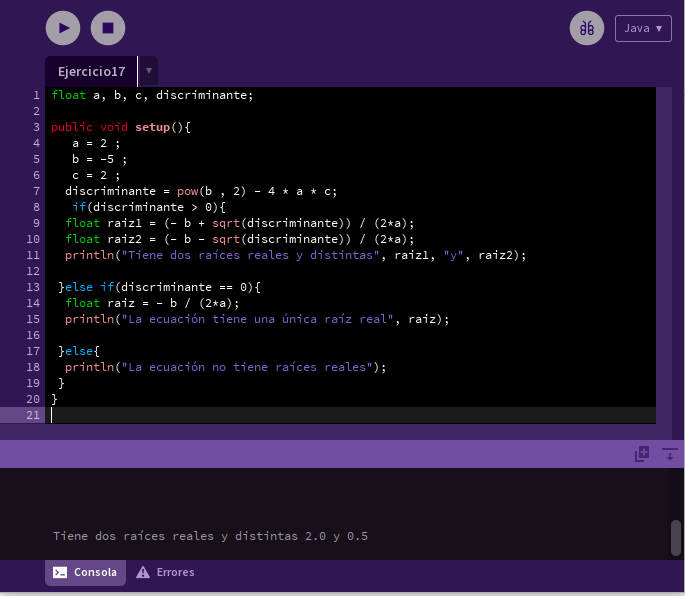
*¿Quién debe realizar el proceso?:* El algoritmo

*¿Cuál es el proceso que resuelve?:* Mediante la formula obtener la discriminante de dicha ecuación

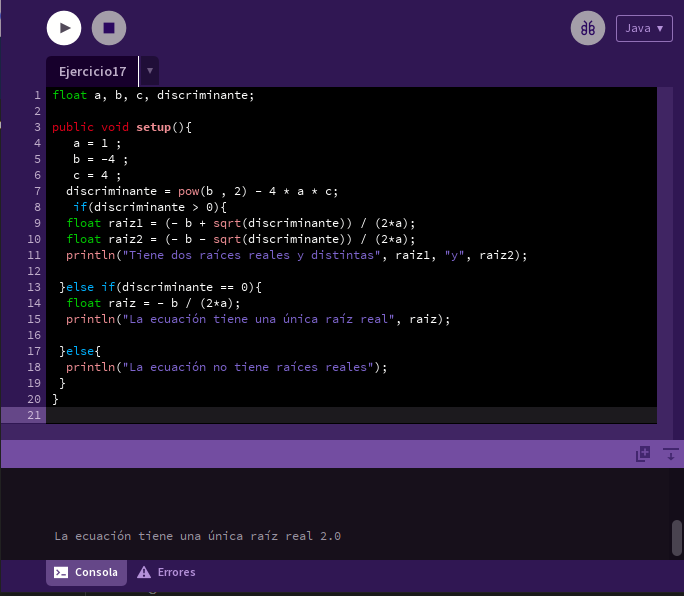
**Diseño:**

|  |
| --- |
| ***ENTIDAD QUE RESUELVE EL PROBLEMA:*** El algoritmo |
| ***VARIABLES:***  • a, b, c, discriminante: float |
| ***NOMBRE ALGORITMO:*** obtener\_raices  ***PROCESO DEL ALGORITMO:***   1. *inicio* 2. *a ⭠ 2* 3. *b ⭠ -5* 4. *c ⭠ 2* 5. *discriminante ⭠ b^2 – 4\*a\*c* 6. *si (discriminante > 0) entonces* 7. *raíz1 ⭠ (-b + (discriminante) ^0.5) / 2\*a* 8. *raiz2 ⭠ (-b – discriminante) ^0.5) / 2\*a* 9. *mostrar "Tiene dos raíces reales y distintas", raiz1, "y", raiz2* 10. *si\_no si(discriminante = 0) entonces* 11. *raiz ⭠ - b / (2\*a)* 12. *mostrar "La ecuación tiene una única raíz real", raíz* 13. *si\_no* 14. *mostrar "La ecuación no tiene raíces reales"* 15. *fin* |

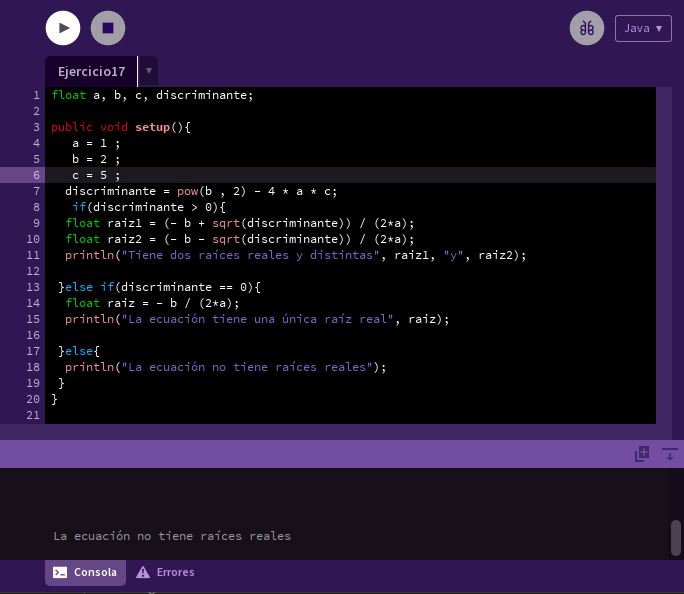
**Processing:** Ejemplo 1



Ejemplo 2

****

Ejemplo 3



**Ejercicio 18:** Declare las variables necesarias para dibujar una línea que se dibuja desde las coordenadas iniciales del lienzo y se extiende por todo el ancho. Sobre el punto medio de la línea y a una distancia de 40px (en sentido vertical desde la línea) dibuje una elipse que tenga como ancho 80px y de alto 80px. Dentro de la función draw(), actualice las variables necesarias para que la línea desde su inicio se mueva en dirección hacia abajo arrastrando la elipse. Mantenga en cero el valor para background(). Cuando la línea supere la posición de la altura del lienzo, debe invertir su sentido, es decir dirigirse hacia arriba arrastrando la elipse. Cuando la línea alcance nuevamente el valor 0 para su posición en y, el desplazamiento debe ser hacia abajo y así sucesivamente. El lienzo debería verse como en las siguientes figuras

**Resolución:**

*Definición del problema:* Definir variables que dibujen una línea y un eclipse el cual arrastre la línea de arriba a abajo

**Análisis:**

*Datos de Entrada:* Los valores de la línea y del eclipse

*Datos de Salida:* El bucle de la línea y el eclipse subiendo y bajando por el lienzo

*Proceso:*

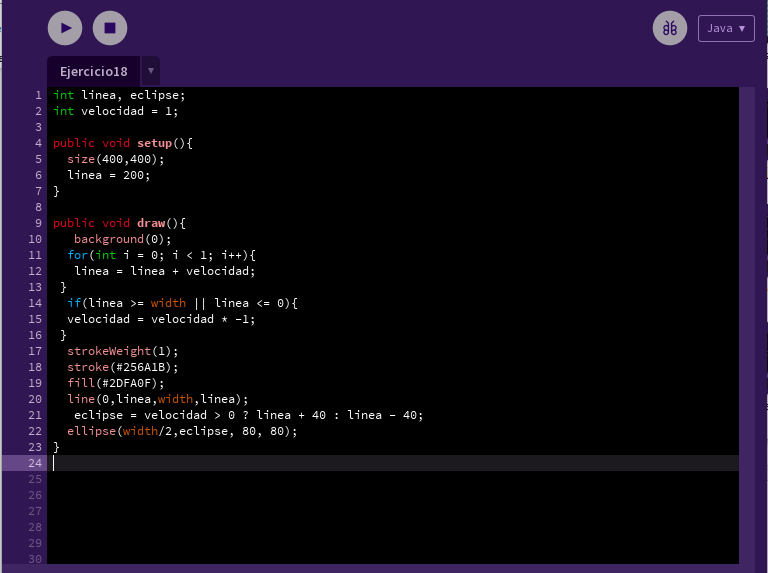
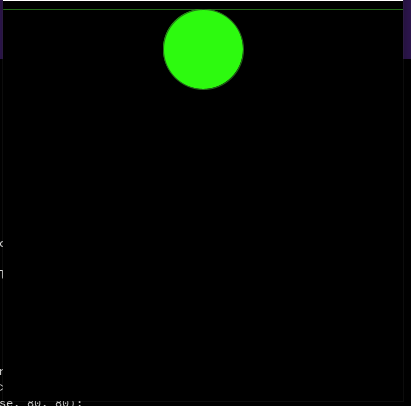
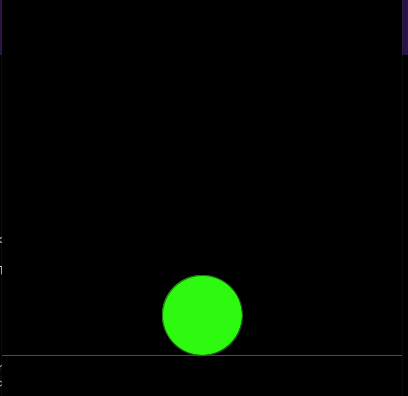
*¿Quién debe realizar el proceso?:* El algoritmo

*¿Cuál es el proceso que resuelve?:* Mediante las variables y los bucles realizar dicho pedido

**Diseño:**

|  |
| --- |
| ***ENTIDAD QUE RESUELVE EL PROBLEMA:*** El algoritmo |
| ***VARIABLES:***  • línea, eclipse: int  • velocidad: int |
| ***NOMBRE ALGORITMO:*** línea\_eclipse  ***PROCESO DEL ALGORITMO:***   * + - 1. *inicio*       2. *velocidad ⭠ 1*       3. *línea ⭠ 200*       4. ***para i*** *⭠ 0* ***hasta*** *1 incremento 1* ***hacer***       5. *línea ⭠ línea + velocidad*       6. ***fin\_para***       7. ***si*** *((linea >= anchoLienzo) O (linea <= 0))* ***entonces***       8. *velocidad ⭠ velocidad \* -1*       9. ***fin\_si***       10. *dibujar línea (0, línea, anchoLienzo, línea*       11. *eclipse ⭠ velocidad > 0 ? linea + 40 : linea - 40*       12. *dibujar eclipse (anchoLienzo/2, eclipse, 80, 80)*       13. *fin* |

**Processing:**



**Ejercicio 19:** Dibuje en toda la extensión del lienzo de (440, 420) rectángulos de idénticas medidas (40 ancho y 20 de alto) y que mantengan una distancia de 20 pixeles entre ellos tanto horizontal como verticalmente. Utilice la estructura de control repetitiva for

**Resolución:**

*Definición del problema:* Dibujar rectángulos a lo largo y ancho del lienzo utilizando la estructura iterativa for

**Análisis:**

*Datos de Entrada:* el alto y ancho de los rectángulos y la distancia entre ellos que es de 20 pixeles

*Datos de Salida:* Los rectángulos dibujados

*Proceso:*

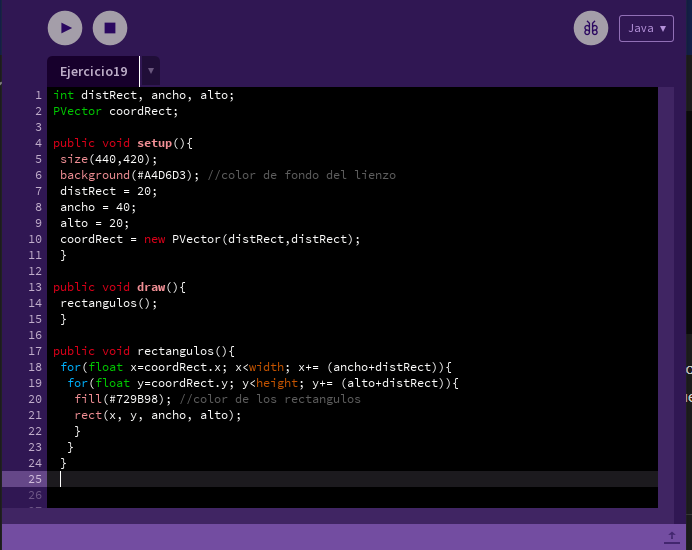
*¿Quien debe realizar el proceso?:* El programa processing

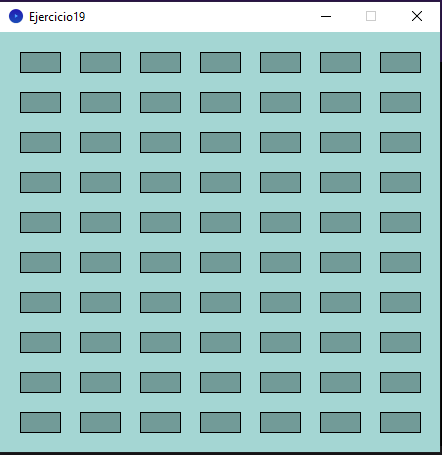
*¿Cual es el proceso que resuelve?:* dibujar rectángulos a lo largo y ancho del lienzo, distanciados de una forma específica utilizando un bucle for

**Diseño:**

|  |
| --- |
| **ENTIDAD QUE RESUELVE EL PROBLEMA*:*** El algoritmo |
| **VARIABLES:**  • coordRect: float  • distRect, anchoLienzo, altoLienzo, ancho, alto : entero |
| **NOMBRE ALGORITMO:** dibujar\_rectangulos  **PROCESO DEL ALGORITMO:**   1. *Inicio* 2. *anchoLienzo ⭠ 440* 3. *altoLienzo ⭠ 420* 4. *distRect ⭠ 20* 5. *ancho ⭠ 40* 6. *alto ⭠ 20* 7. *para.x ⭠ coordRect.x* ***hasta*** *anchoLienzo* ***con paso*** *(ancho+distRect)* ***hacer*** 8. *para.y⭠ coordRect.y* ***hasta*** *altoLienzo* ***con paso*** *(alto+distRect)* ***hacer*** 9. *dibujar rectángulo (x, y,ancho,alto)* 10. *fin\_para* 11. *fin\_para* 12. *fin* |

**Processing:**





**Ejercicio 20:** Utilizando la estructura de control repetitiva while() dibuje la siguiente imagen utilizando líneas que forman escalones y sobre cada borde de escalón se dibuje un punto de color rojo

**Resolución:**

*Definición del problema:* Realizar, mediante un bucle while, una líneas formando escalones y sobre cada escalón un punto color rojo

**Análisis:**

*Datos de Entrada:* El puntoA, puntoB, puntoC, puntoD que representan las coodenadas y la distancia

*Datos de Salida:* dibujado en el lienzo una escalera con puntos rojos en cada esquina

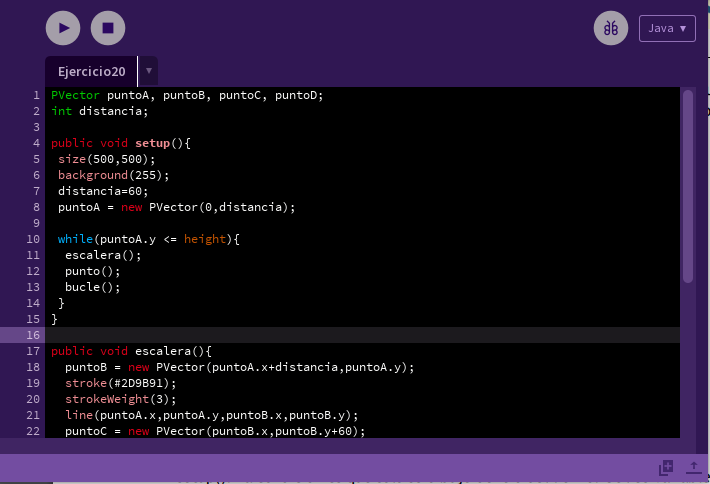
*Proceso:*

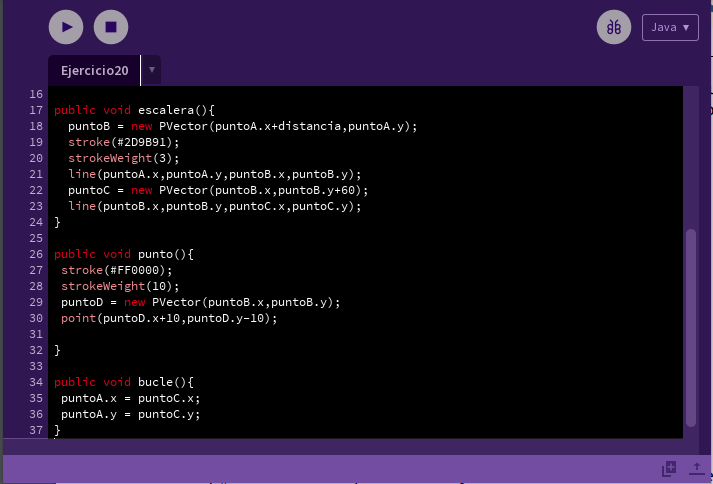
*¿Quien debe realizar el proceso?:* El algoritmo

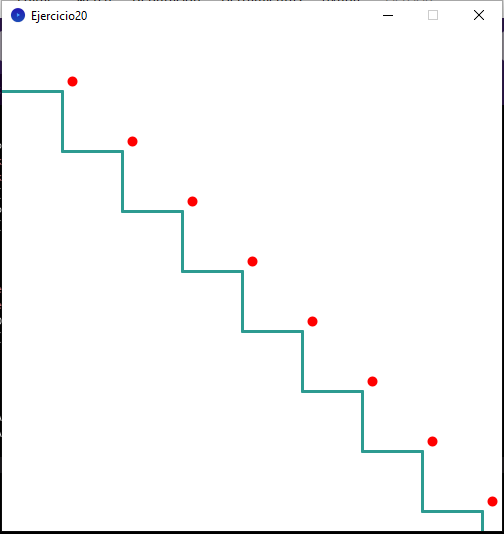
*¿Cual es el proceso que resuelve?:* dibujar unas líneas formando una escalera y puntos rojos en cada esquina mediante un bucle while

**Diseño:**

|  |
| --- |
| ***ENTIDAD QUE RESUELVE EL PROBLEMA:*** El algoritmo |
| ***VARIABLES:***  • puntoA, puntoB, puntoC, puntoD: PVector  • distancia: int |
| ***NOMBRE ALGORITMO:***  ***PROCESO DEL ALGORITMO:***   1. *inicio* 2. *distancia ⭠ 60* 3. ***mientras*** *(puntoA.y sea menor o igual que anchoLienzo)* ***Hacer*** 4. *dibujar línea (puntoA.x, puntoA.y, puntoB.x, puntoB.y)* 5. *dibujar línea (puntoB.x, puntoB.y, puntoC.x, puntoC.y)* 6. *dibujar punto (puntoD.x, puntoD.y)* 7. *puntoA.x ⭠ puntoC.x* 8. *puntoA.y ⭠ puntoC.y* 9. ***fin\_mientras*** 10. *Fin* |

**Processing:** 





**Ejercicio 21:** Utilizando la estructura de control repetitiva do-while. La imagen debe ser construida desde la función setup(). Defina el tamaño del lienzo en size(600,600), verticalmente se divide el lienzo en franjas de igual medida, se deben dibujar los círculos sobre cada línea de por medio es decir en la línea 1 se dibujan círculos con distanciamiento, en la línea 2 no se dibuja y así sucesivamente. Las líneas tienen un color fijo, los círculos asumen colores aleatorios.

**Resolución:**

*Definición del problema:* dibujar líneas a lo largo del lienzo y sobre ellas una fila de círculos en línea de por medio de colores aleatorios mediante un bucle do-while

**Análisis:**

*Datos de Entrada:* x, y, distEclipse y eclipseY

*Datos de Salida:* dibujado líneas a lo largo del lienzo y sobre ellas una fila de círculos, en líneas de por medio de colores aleatorios

*Proceso:*

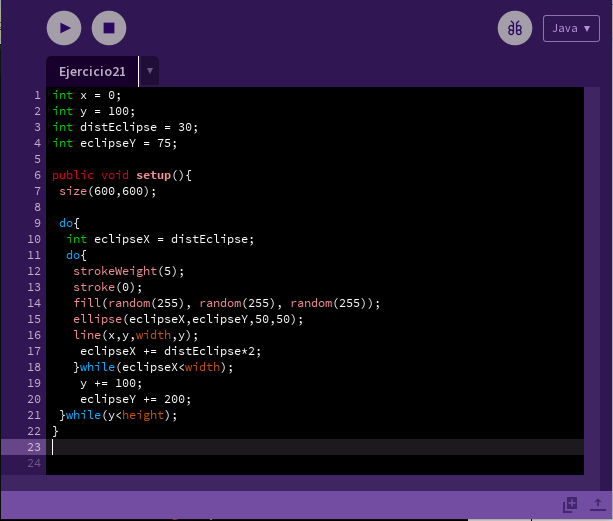
*¿Quien debe realizar el proceso?:* El algoritmo

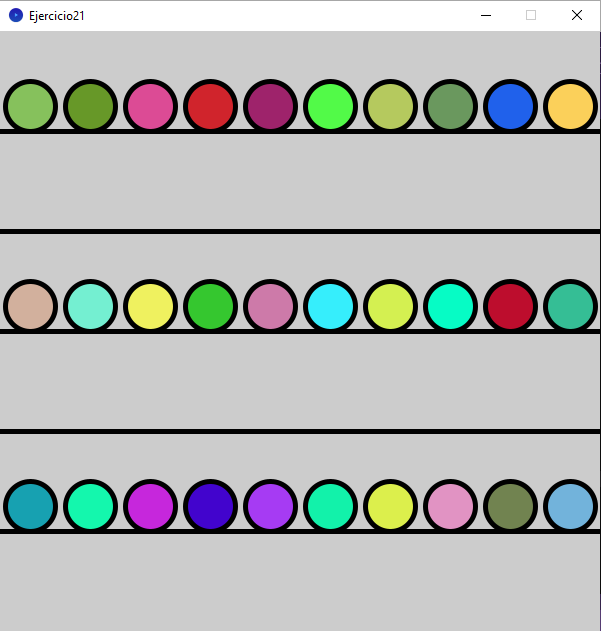
*¿Cual es el proceso que resuelve?:* Mediante un bucle do-while, en el lienzo se dibujan líneas verticales de igual medida. En cada línea de forma alternada, se dibujan círculos con colores aleatorios

**Diseño:**

|  |
| --- |
| ***ENTIDAD QUE RESUELVE EL PROBLEMA: El algoritmo*** |
| ***VARIABLES:***  • x, y, distEclipse, eclipseY: int |
| ***NOMBRE ALGORITMO:***  ***PROCESO DEL ALGORITMO:***   1. ***Inicio*** 2. ***X*** *⭠ 0* 3. ***Y*** *⭠ 100* 4. ***distEclipse*** *⭠ 30* 5. ***eclipseY*** *⭠ 75* 6. ***hacer*** 7. *int EclipseX ⭠ distEclipse* 8. ***hacer*** 9. ***dibujar circulo (eclipseX,eclipseY,50,50)*** 10. ***dibujar línea (x,y,anchoLienzo,y)*** 11. ***eclipseX*** *⭠ += distEclipse \* 2* 12. ***fin\_hacer*** 13. ***mientras(eclipseX sea menor a anchoLienzo)*** 14. ***y*** *⭠ += 100* 15. ***eclipseY*** *⭠ += 200* 16. ***fin\_hacer*** 17. ***mientras(y sea menor a altoLienzo)*** 18. ***fin\_mientras*** 19. ***fin*** |

**Processing:**





Conclusión

Párrafos de las conclusiones

Fuentes bibliográficas

Se deben enunciar las fuentes (apuntes de la materia, páginas web, videos de youtube, libro (nombre, autores, año), etc)