TP I- TRABAJO DE RESOLUCIONES EN PROCCESING

APELLIDO Y NOMBRE: Catacata Aleman Fátima María del Carmen

DNI: 46598823

LU: 0

Profesor: Ariel Alejandro vega

Diseños de los Algoritmos

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

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

Resolución:

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

6-(4\*B/4)

6-5

1

Ejercicio 2: Evaluar la siguiente expresión

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

|  |
| --- |
| (4 / 2 \* 3 / 6) +( 6 / 2 / 1 /( 5 ^ 2) /( 4 \* 2)) |
| (1)+(3/25/8) |

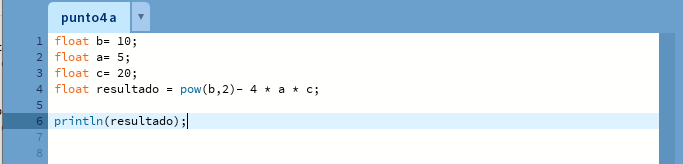
~~Ejercicio 3: Escribir las siguientes expresiones algebraicas como expresiones algorítmicas (en su forma aritmética dentro del algoritmo). En este caso no se pide evaluarlas ni programarlas.~~

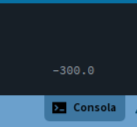
Ejercicio 4: Evaluar las siguientes expresiones aritméticas, para lo cual indicar en el caso de las variables, el valor indicado. Luego escribirlas como expresiones algebraicas.

1. b ^ 2 – 4 \* a \* c

Resolución

|  |  |
| --- | --- |
| (10 ^ 2 )– 4 \* 5 \* 20 | 10 |
| (100)- 400  -300 |  |

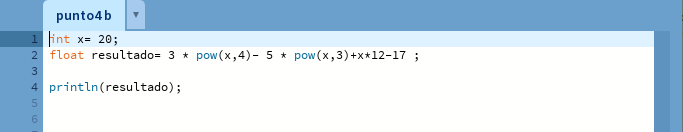


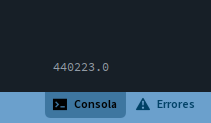


1. 3 \* X ^ 4 – 5 \* X ^ 3 + X 12 – 17

Resolución

|  |  |
| --- | --- |
|  |  |
|  |  |

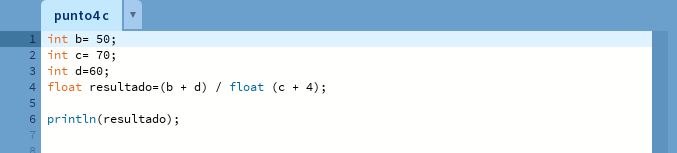
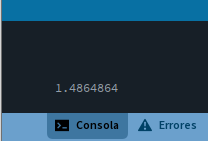




1. (b + d) / (c + 4)

Resolución

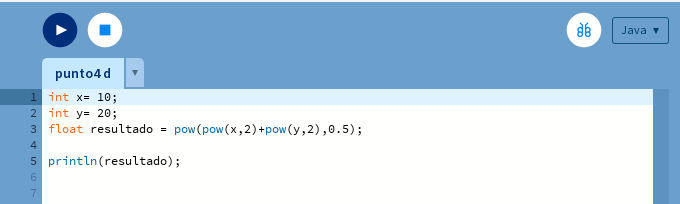
|  |  |
| --- | --- |
|  |  |
|  |  |

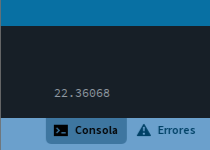


1. (x ^ 2 + y ^ 2) ^ (1 / 2)

Resolución

|  |  |
| --- | --- |
|  |  |
|  |  |

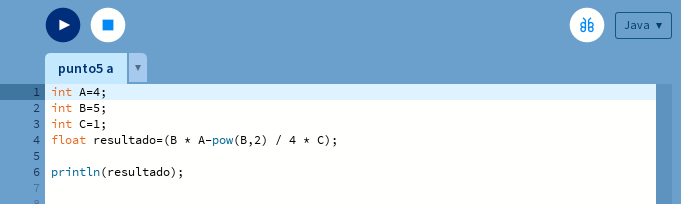


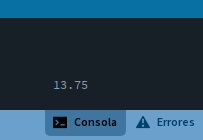


Ejercicio 5: Si el valor de A es 4, el valor de B es 5 y el valor de C es 1, evaluar las siguientes expresiones:

1. B \* A – B ^ 2 / 4 \* C

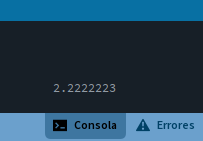
|  |
| --- |
|  |





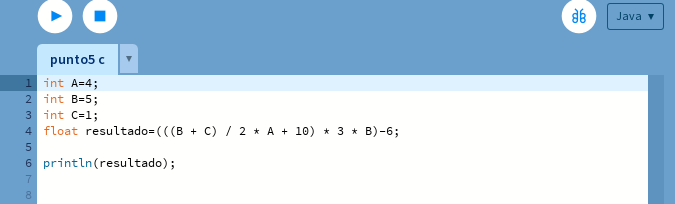
1. (A \* B) / 3 ^ 2

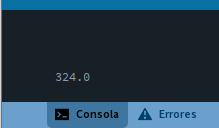
|  |
| --- |
| punto5_b.png |



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

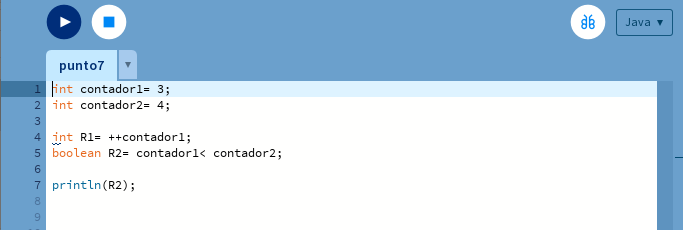
|  |
| --- |
|  |

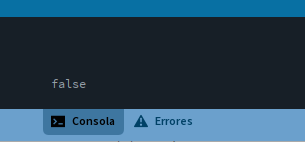




Ejercicio 6: Para x=3, y=4; z=1, evaluar el resultado de

R1 = y+z

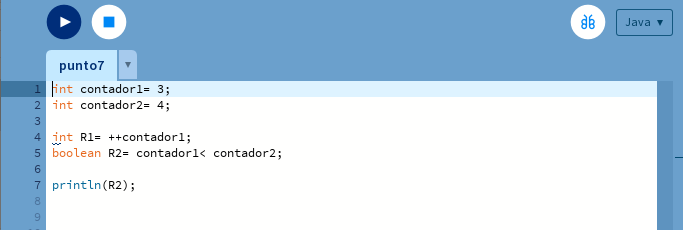
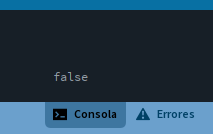
R2 = x >= R1



Ejercicio 7: Para contador1=3, contador3=4, evaluar el resultado de

R1 = ++contador1

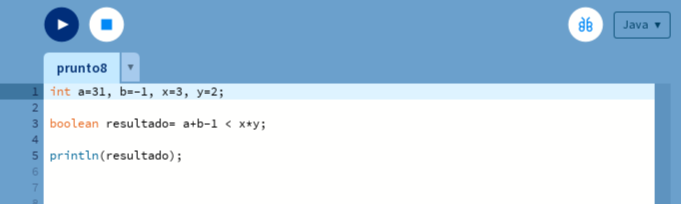
R2 = contador1 < contador2

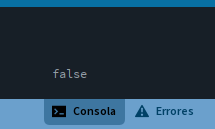


Ejercicio 8: Para a=31, b=-1; x=3, y=2, evaluar el resultado de

a+b-1 < x\*y

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

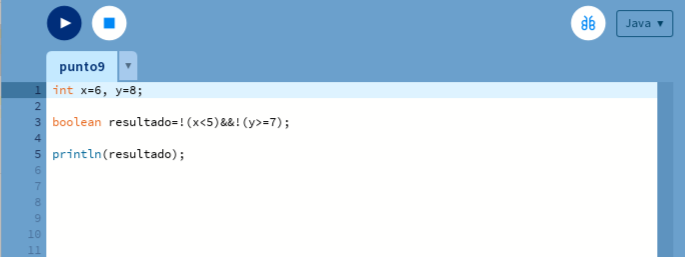
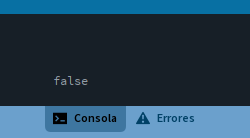




Ejercicio 9: Para x=6, y=8, evaluar el resultado de

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

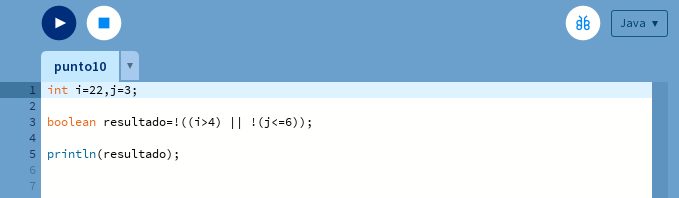
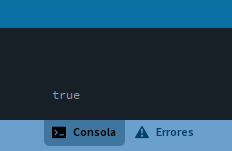
|  |
| --- |
| !(6<5) && !(8>=7)  falso && falso  falso |



Ejercicio 10: Para i=22,j=3, evaluar el resultado de

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

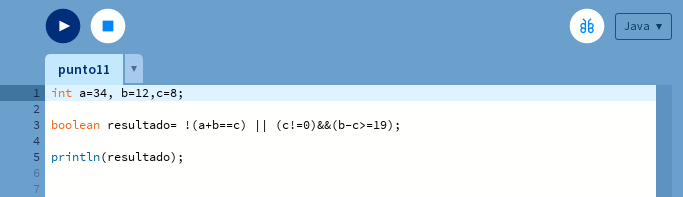
|  |
| --- |
| !((22>4) || !(3<=6))  !(verdadero || falso)  !(verdadero)  Falso |

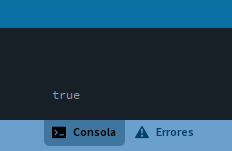


Ejercicio 11: Para a=34, b=12,c=8, evaluar el resultado de

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

|  |
| --- |
| !(34+12==8) || (8!=0)&&(12-8>=19)  !(46==8) || (8!=0)&&(4>=19)  verdadero || verdadero && falso  verdadero|| falso  verdadero |





Ejercicio 12: 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.

Definición del Problema:

• Datos de Entrada:

Nombre<- caracteres

• Datos de Salida:

Bienvenido…<- caracteres

• Proceso:

¿Quién debe realizar el proceso?

El algoritmo

¿Cuál es el proceso que realiza el programa?

Saludar al usuario con el nombre introducido

|  |
| --- |
| ENTIDAD QUE RESUELVE EL PROBLEMA: Algoritmo |
| VARIABLES:  intoduceTuNombre: string // donde podras un nombre  frase: string // dira la frase de bienvenida |
| NOMBRE ALGORITMO: saludo  Algoritmo:  1. inicio  2. Leer  3. intoduceTuNombre← “ Bienvenida” + . intoduceTuNombre;  4. Mostrar saludo  5. fin |
| Captura de pantalla (17).pngpunto12.png |

Ejercicio 13: Será común resolver problemas utilizando variables. Calcule el perímetro y área de un rectángulo dada su base y su altura Definición del Problema:

• Datos de Entrada:

Base<- enteros

Altura<- enteros

• Datos de Salida:

Perímetro<- enteros

Área<-enteros

• Proceso:

¿Quién debe realizar el proceso?

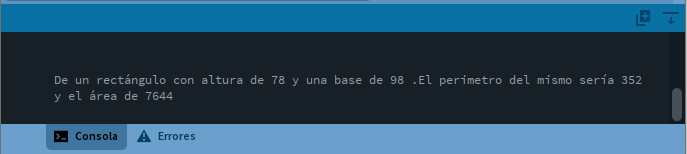
Una claculadora

¿Cuál es el proceso que realiza el programa?

Calcular el per y área del rectángulo según los valores de base y altura, con la siguiente formula

𝑃=2(𝑏𝑎𝑠𝑒 +𝑎𝑙𝑡𝑢𝑟𝑎) 𝑦 𝐴=𝑏𝑎𝑠𝑒 ⋅ 𝑎𝑙𝑡𝑢𝑟𝑎

|  |
| --- |
| ENTIDAD QUE RESUELVE EL PROBLEMA: Algoritmo |
| VARIABLES:  base: int//almacena un valor entero  altura: int//almacena un valor entero  resultPer:int / /almacena el resultado entero  resultAr:int///almacena el resultado entero |
| NOMBRE ALGORITMO: perímetro y area de un rectangulo  Algoritmo   1. Inicio 2. Leer base 3. Leer altura 4. resultPer ← 2\*(base + altura) 5. resultAr ← base \* altura 6. perArea ← “De un rectángulo con altura de : ” + altura + “ y una base de “+base+ “.El perímetro del mismo seria”+resultPer+” y el area de”+resultAr; 7. Mostrar perArea 8. Fin |
| punto13.png |



Ejercicio 14: 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.

Definición del Problema: obtener la hipotenusa del triangulo

• Datos de Entrada:

Cateto1<-reales

Cateto2<-reales

• Datos de Salida:

Hipotenusa<- reales

• Proceso:

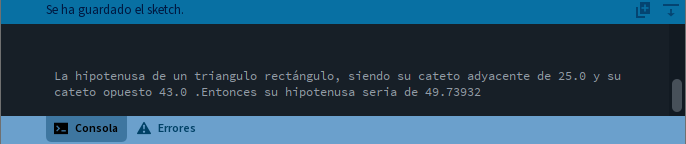
¿Quién debe realizar el proceso?

La computadora

¿Cuál es el proceso que realiza el programa?

Obtener la hipotenusa según el cálculo de los catetos

|  |
| --- |
| ENTIDAD QUE RESUELVE EL PROBLEMA: Algoritmo |
| VARIABLES:  cateto1: float <-almacena un valor decimal  cateto2: float <-almacena un valor decimal  hipotenusa:float<- resultado decimal |
| NOMBRE ALGORITMO: Hipotenusa  Algoritmo   1. Inicio 2. Leer cateto1 3. Leer cateto2 4. Hipotenusa: (a^2 + b^2 ) ^(0.5) 5. Resultado<- “LA hipotenusa de un triangulo rectangulo, siendo su cateto adyacente”+cateto1+…. “su hipotenusa seria”+hipotenusa; 6. Mostrar resultdo 7. Final |
| punto14.png |



Ejercicio 15: 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.

Definición del Problema:

• Datos de Entrada:

Num1<-reales

Num2<-reales

• Datos de Salida:

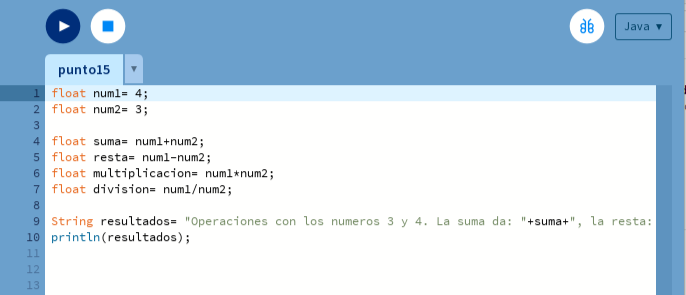
resultado

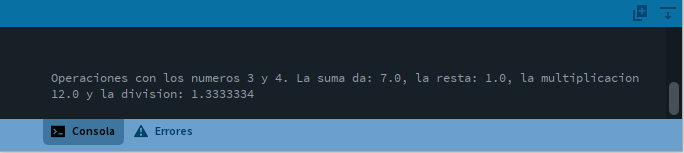
• Proceso:

¿Quién debe realizar el proceso?:

¿Cuál es el proceso que realiza el programa?

|  |
| --- |
| ENTIDAD QUE RESUELVE EL PROBLEMA: Algoritmo |
| VARIABLES: |
| NOMBRE ALGORITMO:  Algoritmo |
| PROCESO DEL ALGORITMO: |





Ejercicio 16: 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

Definición del Problema:

• Datos de Entrada:

• Datos de Salida:

• Proceso:

¿Quién debe realizar el proceso?:

¿Cuál es el proceso que realiza el programa?

|  |
| --- |
| ENTIDAD QUE RESUELVE EL PROBLEMA: Algoritmo |
| VARIABLES: |
| NOMBRE ALGORITMO:  Algoritmo |
| PROCESO DEL ALGORITMO: |

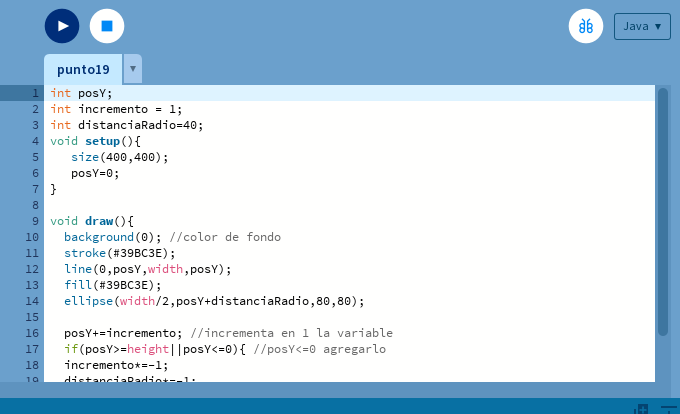
Ejercicio 17: Si queremos representar personajes o power ups (premios) en la pantalla debemos primero ubicarlos en alguna posición dentro de la pantalla. Imagine que está en un juego donde un power up desaparece porque el personaje se acerca a una distancia de x unidades, sin importar por donde se acerque. 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

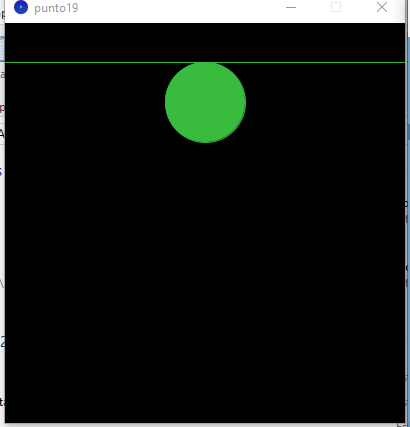
|  |
| --- |
| ENTIDAD QUE RESUELVE EL PROBLEMA: Algoritmo |
| VARIABLES: |
| NOMBRE ALGORITMO:  Algoritmo |
| PROCESO DEL ALGORITMO: |

Ejercicio 18: 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.

Ejercicio 19: 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

|  |
| --- |
| ENTIDAD QUE RESUELVE EL PROBLEMA: Algoritmo |
| VARIABLES: |
| NOMBRE ALGORITMO:  Algoritmo |
| PROCESO DEL ALGORITMO: |





Ejercicio 20: 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.

Definición del Problema: Dibujar rectángulos usando estructuras iterativas

Análisis:

• Datos de Entrada:

coordenadasRect <- coordenadas cartesianas

anchó, alto, distanciaEntreRect <- entero

• Datos de Salida:

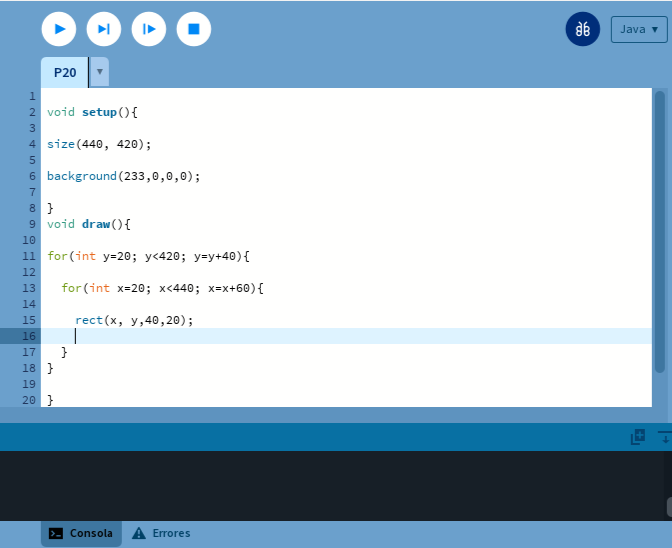
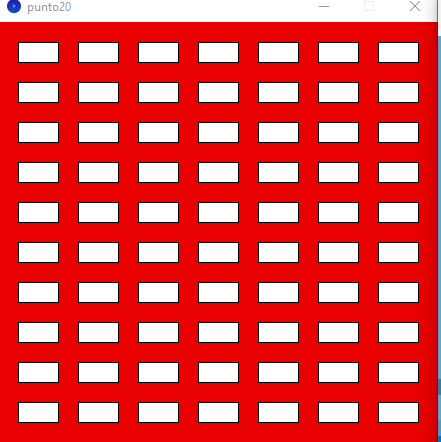
rectangulos\_dibujados

• Proceso:

¿Quién debe realizar el proceso?: el programa

¿Cuál es el proceso que realiza el programa? Es dibujar los rectangulos

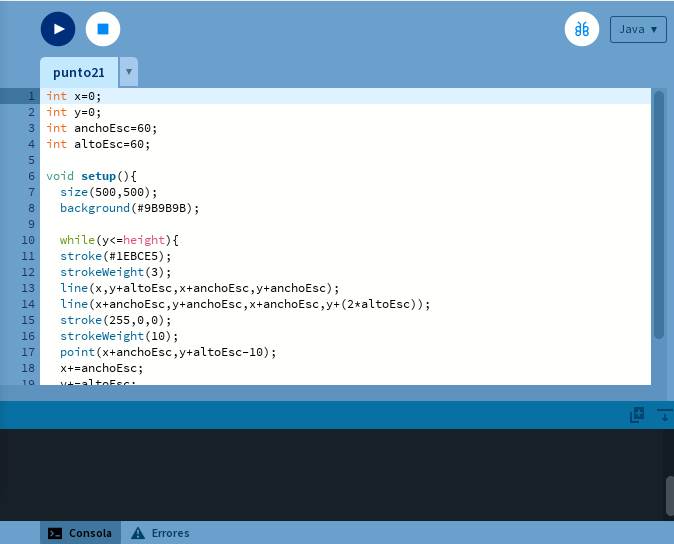
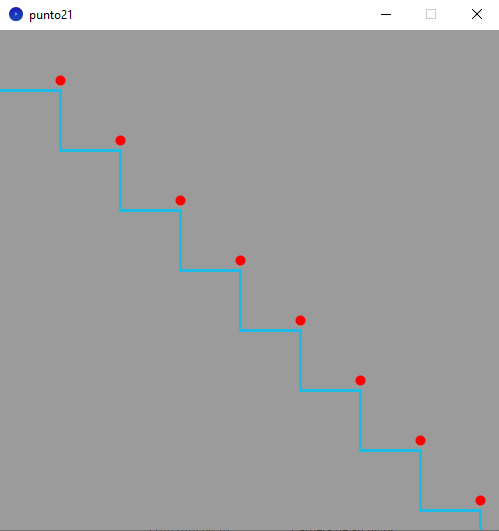
|  |
| --- |
| ENTIDAD QUE RESUELVE EL PROBLEMA: Algoritmo |
| VARIABLES:  coordenadasRect <- coordenadas cartesianas  anchó, alto, distanciaEntreRect <- entero  altoLienzo,anchoLienzo <- entero |
| NOMBRE ALGORITMO: rectangulos-dibujos  Algoritmo  Inicio  anchoLienzo <-440  altoLienzo <-420  distanciaEntreRect <- 20  ancho <- 40  alto <- 20  dibujo.un.rectangulo.en.(distanciaEntreRect,distanciaEntreRect) |
| PROCESO DEL ALGORITMO: |



Ejercicio 21: 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.

El tamaño del lienzo es size(500,500). La estructura while() se ejecuta dentro de la función setup(). La condición es que solo se dibuje dentro del lienzo. Utilice variables que puedan ayudar a la construcción del dibujo, por ej: x, y, anchoEscalon, altoEscalon, etc

|  |
| --- |
| ENTIDAD QUE RESUELVE EL PROBLEMA: Algoritmo |
| VARIABLES: |
| NOMBRE ALGORITMO:  Algoritmo |
| PROCESO DEL ALGORITMO: |



Ejercicio 22: Utilizando la estructura de control repetitiva do-while. Replique la siguiente imagen.

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

|  |
| --- |
| ENTIDAD QUE RESUELVE EL PROBLEMA: Algoritmo |
| VARIABLES: |
| NOMBRE ALGORITMO:  Algoritmo |
| PROCESO DEL ALGORITMO: |

