TECNICATURA UNIVERSITARIA EN DISEÑO
INTEGRAL DE VIDEOJUEGOS

FACULTAD DE INGENIERÍA
Universidad Nacional de Jujuy

FUNDAMENTOS DE PROGRAMACIÓN ORIENTADA A OBJETOS

Trabajo Práctico

N° 1

Gomez Juan Gonzalo

LU: TUV000612

Profesores:
Mg. Ing. Ariel Alejandro Vega
Año 2024

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

3* A - 4 * B / A ^ 2

Resolución necesaria en Word:

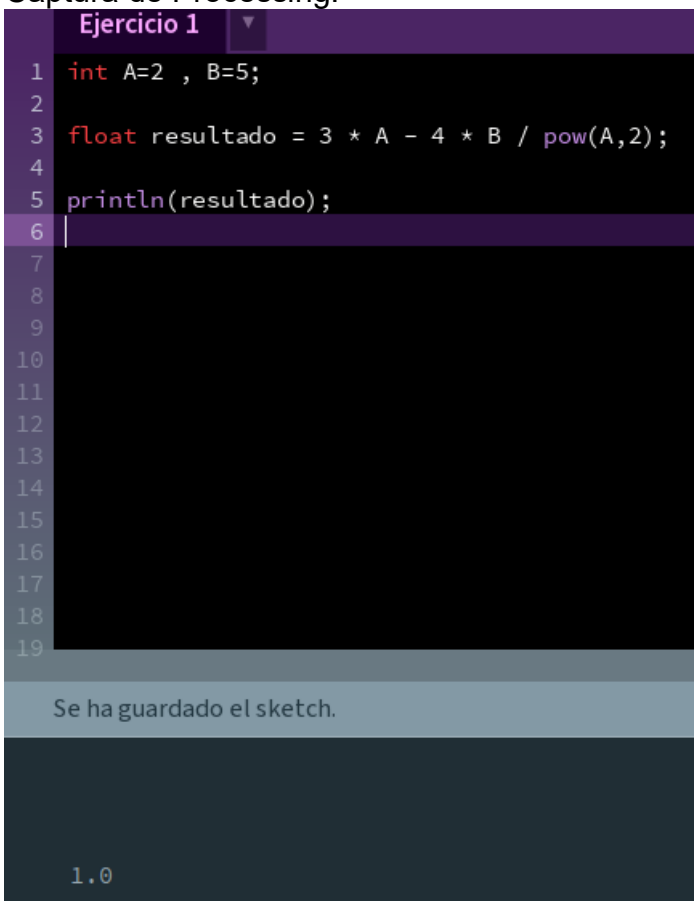
(3*A)-(4*B/(A^2))

6-(4*B/4)

6-5

1

Captura de Processing:



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

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

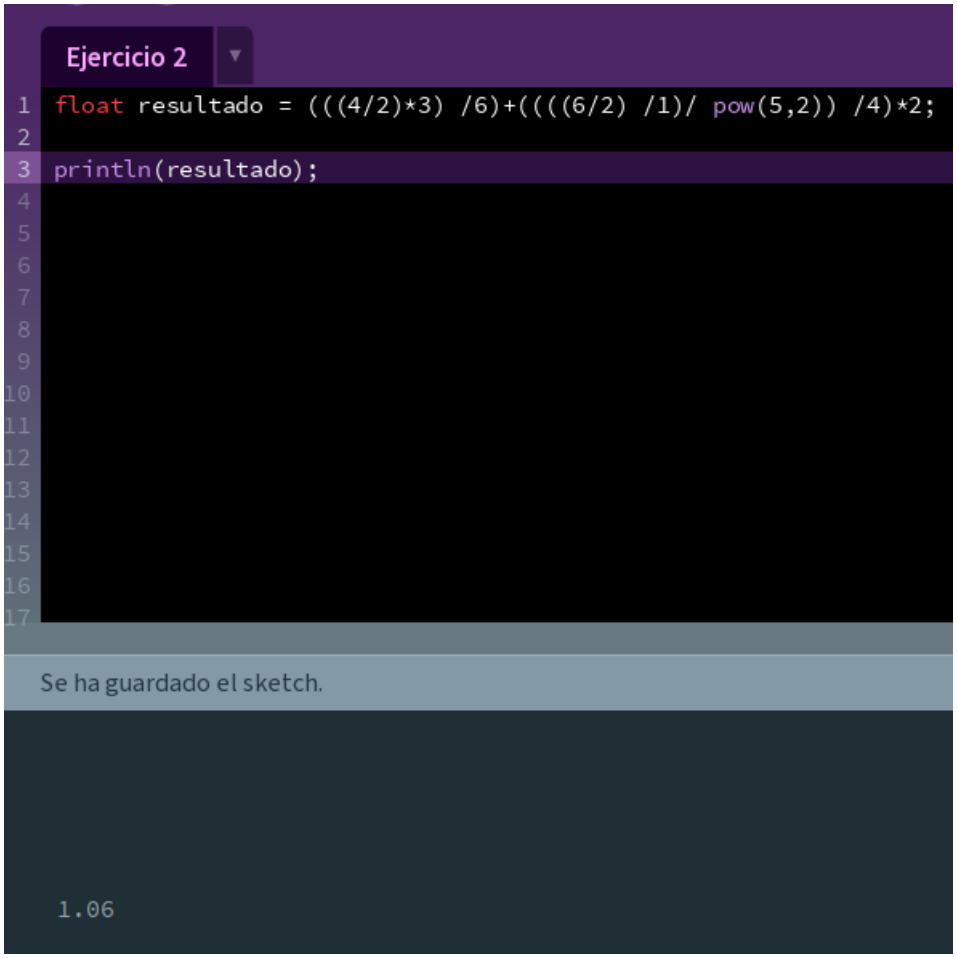
(6 / 6) + ((3/ 25) /4) * 2

1 + (0,12/ 4) * 2

1 + 0,03 * 2

1 + 0,06

1,06



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:

a) B * A – B ^ 2 / 4 * C

5 * 4 – ((5 ^ 2) / 4) * 1

20 - (25/4) * 1

20 - 6,25

13,75

b) (A * B) / 3 ^ 2

(4 * 5) / 3 ^ 2

20 / 9

2,222...

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

(((5 + 1) / 2 * 4 + 10) * 3 * 5) - 6

((6 / 2 * 4 + 10) * 3 * 5) - 6

((3 * 4 + 10) * 3 * 5) - 6

(22 * 3 * 5) - 6

330 – 6

324

Ejercicio 5 a

```
1  int A = 4, B = 5, C = 1;
2
3  float resultado = B*A - (pow(B, 2) /4)*C;
4
5  println(resultado);
6
```

```
13.75
```

Ejercicio 5 b

```
1 int A = 4, B = 5;
2
3 float resultado = (A * B) / pow(3, 2);
4
5 println(resultado);
```

```
2.2222223
```

Ejercicio 5 c

```
1 int A=4, B=5, C=1;
2
3 float resultado = (((B+C)/2*A+10)*3*B)-6;
4
5 println(resultado);
6
```

```
324.0
```

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

R1 = y+z

R2 = x >= R1

R1= 4+1 = 5

R2= 3 >= R1

Falso

Ejercicio 6

```
1 int x = 3, y = 4, z = 1;
2
3 int R1 = y + z;
4
5 boolean R2 = 3 >= R1;
6
7 println(R2);
```

```
false
```

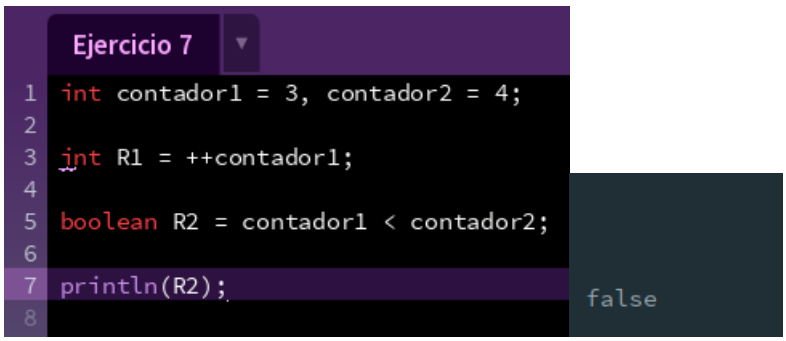
Ejercicio 7: Para contador1=3, contador3=4, evaluar elresultado de

R1 = ++contador1

R2 = contador1 < contador2

R2= 4 < 4

R2= falso



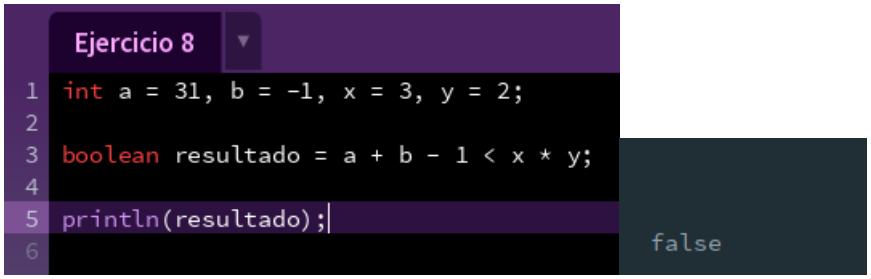
Ejercicio 8: Para a=31, b=-1; x=3, y=2, evaluar el resultado de

a+b-1 < x*y

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

29 < 12

falso



Ejercicio 9 Para x=6, y=8, evaluar el resultado de ! (x<5) &&!(y>=7)

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

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

falso && falso

falso

Ejercicio 9

```
1  int x = 6, y = 8;
2
3  boolean resultado = !(x < 5) && !(y >= 7);
4
5  println(resultado);
6
```

```
false
```

Ejercicio 10Para i=22, j=3, evaluar el resultado de !((i>4) || !(j<=6))

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

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

!(verdadero || falso)

!(verdadero)

Falso

Ejercicio 10

```
1  int i = 22, j = 3;
2
3  boolean resultado = !((i > 4) || !(j <= 6));
4
5  println(resultado);
6
```

```
false
```

Ejercicio 11 Para a=34, b=12, c=8, evaluar el resultado de
!(a+b==c) || (c!=0) && (b-c>=19)

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

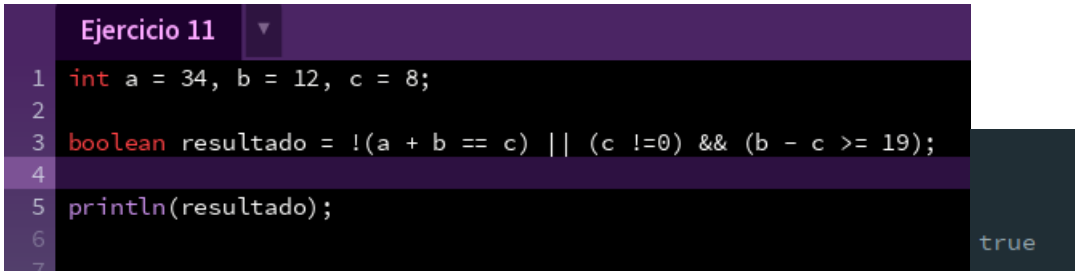
!(34+12==8) || (8!=0)&&(12-8>=19)

!(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.

Análisis:

Datos de Entrada: nombre // cadena de texto

Datos de Salida: saludo // cadena de texto

Proceso:

¿Quien debe realizar el proceso?: El algoritmo o computadora

¿Cual es el proceso que resuelve?: Ingresar un nombre en la consola y mostraste un saludo con tu nombre en pantalla

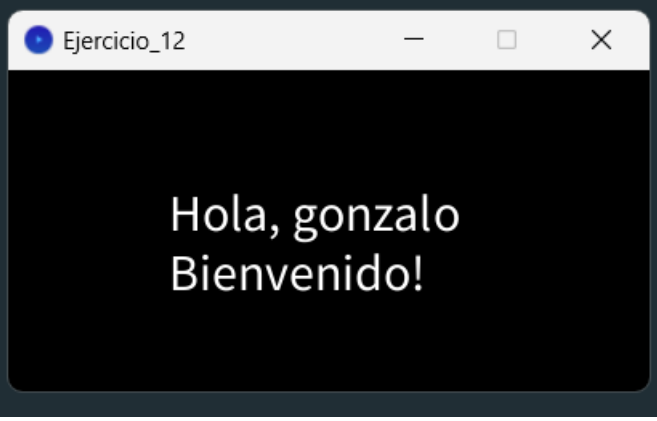
Diseño:

Entidad que resuelve el problema: Algoritmo
Variables: **nombre: string //** almacena el nombre **saludo: string //** almacenara una cadena de caracteres
Nombre del Algoritmo: saludar
Proceso del algoritmo: *inicio* *Leer nombre* *saludo ← "Hola, " + nombre + " ¡Bienvenido!"* *Mostrar saludo* *fin*

Ejercicio 12

```
1  String texto = "escribe tu nombre:";
2  String nombre = "";
3  String saludo = "";
4
5  void setup() {
6    size(400, 200);
7    println(texto);
8  }
9
10 void draw() {
11   background(0);
12   text(saludo, 100, 100);
13   textSize(32);
14
15 }
16
17 void keyPressed() {
18   nombre += key;
19   println(nombre);
20
21   if (key == '\n') {
22     saludo = "Hola, " + nombre + "Bienvenido!";
23     println(saludo);
24   }
25 }
```

```
escribe tu nombre:
g
go
gon
gonz
gonza
gonzal
gonzalo
gonzalo

Hola, gonzalo
Bienvenido!
```



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

Análisis:

Datos de Entrada: base, altura //almacena valores decimales

Datos de Salida: perimetro, area // almacena valores decimales

Proceso:

¿Quién debe realizar el proceso?: El algoritmo

¿Cuál es el proceso que resuelve?: Calcula el perímetro y el área de un rectángulo

Diseño:

Entidad que resuelve el problema: algoritmo
Variables: • **base: float //** almacena un valor decimal • **area: float** // almacena un valor decimal • **perimetro: float // almacena un valor de calculos** • **area: float //almacena un valor de calculos**
Nombre del Algoritmo: calcular
Proceso del algoritmo: *inicio* *Leer base* *Leer area* *perimetro ← 2*(base + altura)* *area ← base * altura* *Mostrar ← "Perímetro del rectángulo: " + perimetro* *Mostrar ← "Área del rectángulo: " + area* *fin*

Ejercicio 13

```
1  float base = 6, altura = 4;
2
3  float perimetro = 2*(base + altura);
4
5  float area = base*altura;
6
7  println("Perímetro del rectángulo: " + perimetro);
8
9  println("Área del rectángulo: " + area);
10
```

```
Perímetro del rectángulo: 20.0
Área del rectángulo: 24.0
```

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

Análisis:

Datos de Entrada: catetoA, catetoB

Datos de Salida: hipotenusa

Proceso:

¿Quien debe realizar el proceso?: la persona

¿Cual es el proceso que resuelve?: Para calcular la hipotenusa de un triángulo rectángulo se usan los catetos y con su formula obtenemos el resultado

Diseño:

Entidad que resuelve el problema: algoritmo
Variables: • **catetoA: int** // almacena un valor decimal • **catetoB: int** // almacena un valor decimal • **hipotenusa: int** // almacena el resultado
Nombre del Algoritmo: calcular
Proceso del algoritmo: *Inicio* *Leer catetoA* *Leer catetoB* *hipotenusa ← (a^2 + b^2) ^(0.5)* *mostrar hipotenusa* *Fin*

Ejercicio 14

```
1  int catetoA = 5, catetoB = 7;
2
3  float hipotenusa = (pow(pow(catetoA, 2) + pow(catetoB, 2), 0.5));
4
5  println("La hipotenusa es: " + hipotenusa);
6
```

```
La hipotenusa es: 8.602325
```

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

Análisis:

Datos de Entrada: n1, n2

Datos de Salida: suma, resta, multiplicacion, division

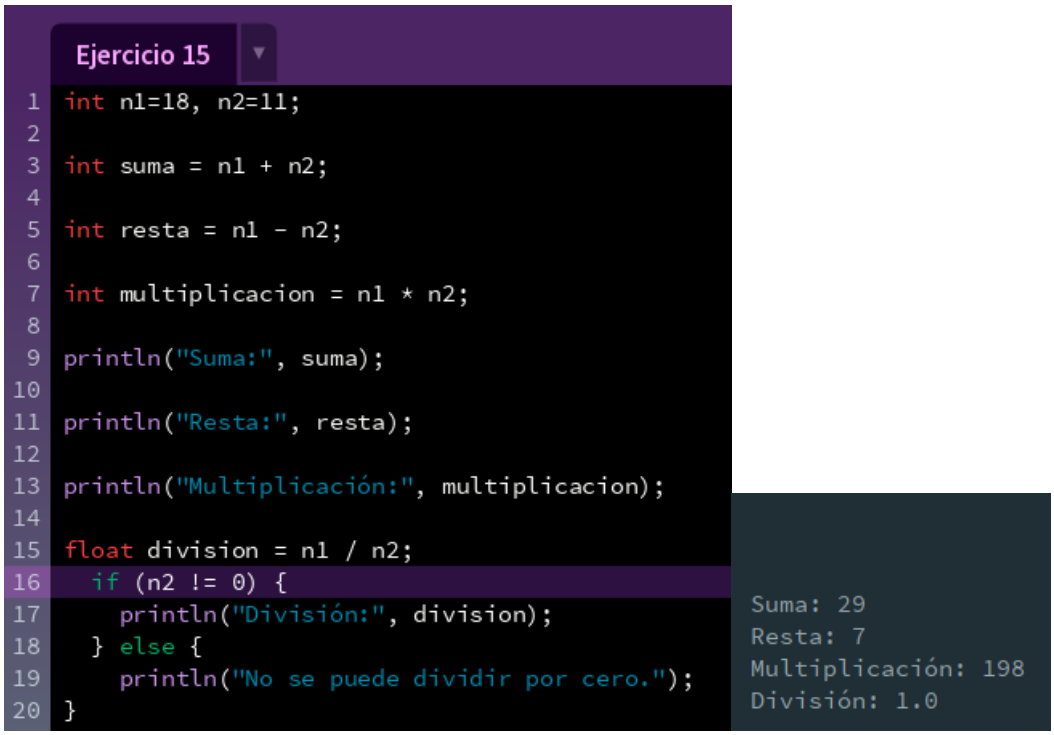
Proceso:

¿Quien debe realizar el proceso?: La persona

¿Cual es el proceso que resuelve?: Resuelve suma, resta, multiplicacion y division

Diseño:

Entidad que resuelve el problema: algoritmo
Variables: • n1: int // almacena un valor entero • n2: int // almacena un valor entero • suma: int // almacena el resultado de la suma • resta: int // almacena el resultado de la resta • multiplicacion: int // almacena el resultado de la multiplicacion • division: float //almacena el resultado de la division
Nombre del Algoritmo: calculadora
Proceso del algoritmo: *inicio* *Leer n1* *Leer n2* *suma ← n1 + n2* *mostrar ← "el resultado de la suma es: " + suma* *resta ← n1 – n2* *mostrar ← "el resultado de la resta es: " + resta* *multiplicacion ← n1 * n2* *mostrar ← "el resultado de la multiplicación es: " + multiplicacion* *division ← n1 / n2* *Si (num2 !=0) entonces* *mostrar ← "el resultado de la división es: " + division* *si_no* *mostrar ← "la division por cero no está definida."* *Fin*



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

Análisis:

Datos de Entrada: Temperatura en grados Fahrenheit

Datos de Salida: Temperatura en grados Celsius

Proceso:

¿Quién debe realizar el proceso?: Algoritmo

¿Cuál es el proceso que resuelve?: Convertir una temperatura dada en grados Fahrenheit a grados Celsius

$$°F = °C \times \frac{9}{5} + 32$$

Diseño:

Entidad que resuelve el problema: algoritmo
Variables: • **tempF: float //** almacena un valor decimal • **tempC: float // almacena un valor decimal**

Nombre del Algoritmo: convertor

Proceso del algoritmo:

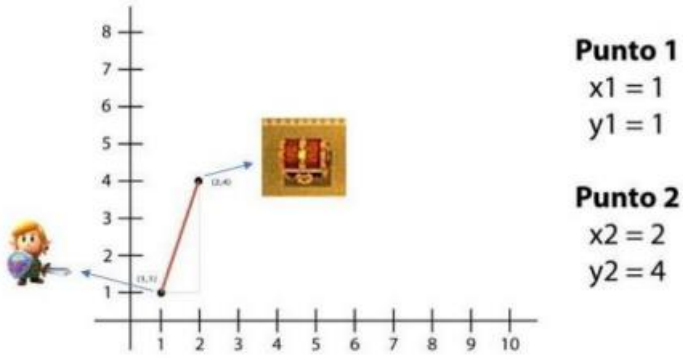
inicio
Leer tempF
*tempC ← (5.0 / 9.0) * (tempF – 32)*
mostrar tempC
fin

Ejercicio 16

```
1  float tempF = 75;
2
3  float tempC = (5.0 / 9.0) * (tempF - 32);
4
5  println("Temperatura en Celsius: " + tempC);
6
```

```
Temperatura en Celsius: 23.88889
```

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 Link con un Circulo, y al tesoro con un cuadrado. Además, mueva a Link mediante el mouse.



Punto 1
x1 = 1
y1 = 1

Punto 2
x2 = 2
y2 = 4

Análisis:

Datos de Entrada: Coordenadas de Link, Coordenadas del tesoro

Datos de Salida: Distancia entre Link y tesoro.

Proceso:

¿Quién debe realizar el proceso?: El programa informático o una calculadora que pueda realizar cálculos matemáticos.

¿Cuál es el proceso que resuelve?: Calculamos las diferencias en las coordenadas **x;y** entre los dos puntos que nos darán los catetos formados por los puntos

Diseño:

Entidad que resuelve el problema: persona

Variables:

- **x1: float //** almacena un valor decimal
- **y1: float // almacena un valor decimal**
- **x2: float //** almacena un valor decimal
- **y2: float //** almacena un valor decimal
- **coordenadaX:** float // almacena el resultado de un calculo
- **coordenadaY:** float //almacena el resultado de un calculo
- **distancia: float // almacena el resultado de un calculo**
- **distanciaTesoro: float // almacena un valor**

Nombre del Algoritmo: calcular_distancia

Proceso del algoritmo:

```
inicio
Leer x1
Leer y1
Leer x2
Leer y2
distanciaTesoro ← 50
coordenadaX ← x2 - x1
coordenadaY ← y2 – y1
distancia ← ((coordenadaX)^2 + (coordenadaY)^2)^2
mostrar “la distancia es de: ” + distancia
si (distancia = distanciaTesoro) entonces
mostrar “¡PowerUp activado!”
fin_si
fin
```

Ejercicio 17

```
float x1 = 100, y1 = 100;
float x2, y2;
float distanciaTesoro = 50;
boolean powerUp = false;

void setup() {
  size(800, 600);
  x2 = random(width - 40) + 20;
  y2 = random(height - 40) + 20;
}

void draw() {
  background(0);
  float dx = x2 - x1;
  float dy = y2 - y1;
  float distancia = sqrt(dx*dx + dy*dy);
  String textoDistancia = "La distancia es de: " + distancia;
  println(textoDistancia);
  if (distancia <= distanciaTesoro && !powerUp) {
    println("¡Power-Up activado!");
    powerUp = true;
    x2 = random(width - 40) + 20;
    y2 = random(height - 40) + 20;
  }
  if (!powerUp) {
    fill(255, 255, 0);
    rect(x2, y2, 40, 40);
  }
  fill(255, 0, 0);
  ellipse(x1, y1, 40, 40);
}

void mouseMoved() {
  x1 = mouseX;
  y1 = mouseY;
}
```

Distancia inicial que marca en consola y mensaje al agarrar el power up:

```
La distancia es de: 143.52269
La distancia es de: 143.52269
La distancia es de: 143.52269
La distancia es de: 143.52269
La distancia es de: 143.52269
La distancia es de: 143.52269
La distancia es de: 143.52269
La distancia es de: 143.52269
La distancia es de: 143.52269

La distancia es de: 41.573708
¡Power-Up activado!
```

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

Análisis:

Datos de Entrada: Coeficientes de la ecuación cuadrática: a, b y c.

Datos de Salida: Raíces de la ecuación cuadrática.

Proceso:

¿Quién debe realizar el proceso?: El programa informático o una calculadora que pueda realizar cálculos matemáticos.

¿Cuál es el proceso que resuelve?: Calcular el discriminante de la ecuación cuadrática utilizando la fórmula

Diseño:

Entidad que resuelve el problema: algoritmo

Variables:

- **a, b, c : float //** almacena un valor decimal
- **discriminante: float** //almacena el valor de calculos

Nombre del Algoritmo: calculadora

Proceso del algoritmo:

inicio
Leer a
Leer b
Leer c
*discriminante ← b^2 – 4*a*c*
si (discriminante > 0) entonces
*raiz1 ← (-b + (discriminante))^0.5 /(2*a)*
*raiz2 ← (-b - (discriminante))^0.5 /(2*a)*
mostrar "las raíces son: " + raiz1 + " y " + raiz2
si_no si (discriminante == 0) entonces
raiz ← *-b / (2*a)*
mostrar "la raíz doble es: " + raiz
si_no
mostrar "no hay raíces reales"
fin

Ejercicio 18

```
float a = 1, b = -3, c = 2;

void setup() {
  size(400, 200);
  float discriminante = b*b - 4*a*c;
  if (discriminante > 0) {
    float raiz1 = (-b + sqrt(discriminante)) / (2 * a);
    float raiz2 = (-b - sqrt(discriminante)) / (2 * a);
    println("Las raíces son: " + raiz1 + " y " + raiz2);
  } else if (discriminante == 0) {
    float raiz = -b / (2 * a);
    println("La raíz doble es: " + raiz);
  } else {
    println("No hay raices reales");
  }
}
```

```
Las raíces son: 2.0 y 1.0
```

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

Análisis:

Datos de Entrada: **línea, dir**

Datos de Salida: bucle de la línea y circulo

Proceso:

¿Quien debe realizar el proceso?: la computadora

¿Cual es el proceso que resuelve?:

Diseño:

Entidad que resuelve el problema: lienzo

Variables:

- **linea: entero** // almacena un valor entero
- **dir : entero // almacena un valor enter**

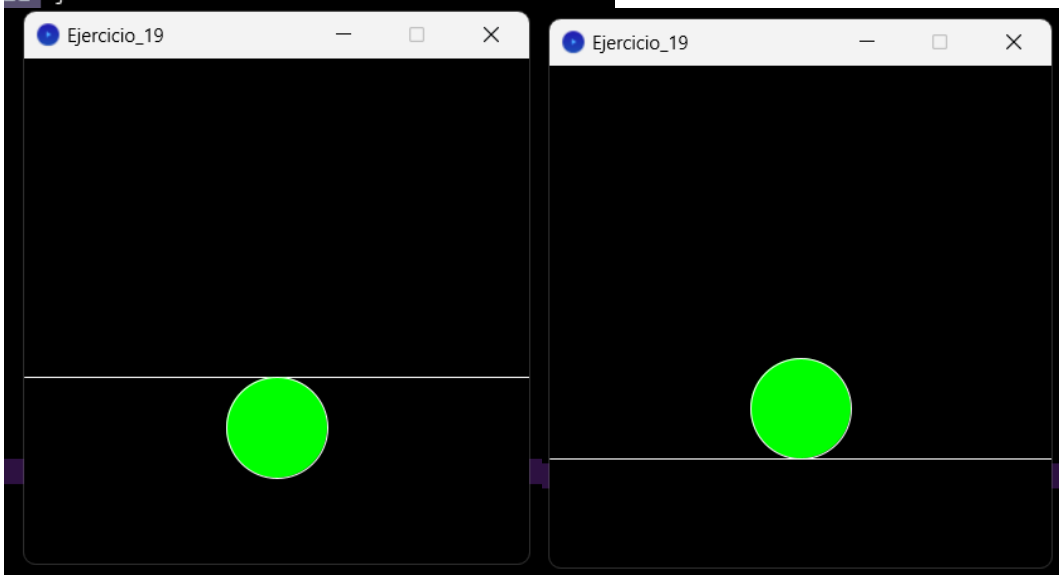
Nombre del Algoritmo: inversion_del_circulo

Proceso del algoritmo:

```
inicio
Leer linea
Leer dir
anchoLienzo ← 400
altoLienzo ← 400
y← linea + dir * 40
si ((linea >= anchoLienzo) O (linea <= 0)) entonces
dir ← dir *= -1
fin_si
mostrar linea
dibujar linea en (dir, linea, altoLienzo, linea)
dibujar circulo en (altoLienzo/2, y, 80, 80)
fin
```

Ejercicio 19

```
1  int linea;
2  int dir = 1;
3
4  void setup() {
5    size(400, 400);
6    linea = height / 2;
7  }
8
9  void draw() {
10   background(0);
11   linea += dir;
12   int ellipseY = linea + (dir * 40);
13   if (linea >= height || linea <= 0) {
14     dir *= -1;
15   }
16
17   stroke(255);
18   line(0, linea, width, linea);
19   fill(0, 255, 0);
20   ellipse(width / 2, ellipseY, 80, 80);
21 }
```



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. El lienzo debería verse así:

Análisis:

Datos de Entrada: distancia, ancho, alto

Datos de Salida: Rectángulos dibujados en el lienzo según las especificaciones dadas.

Proceso:

¿Quien debe realizar el proceso?: la computadora

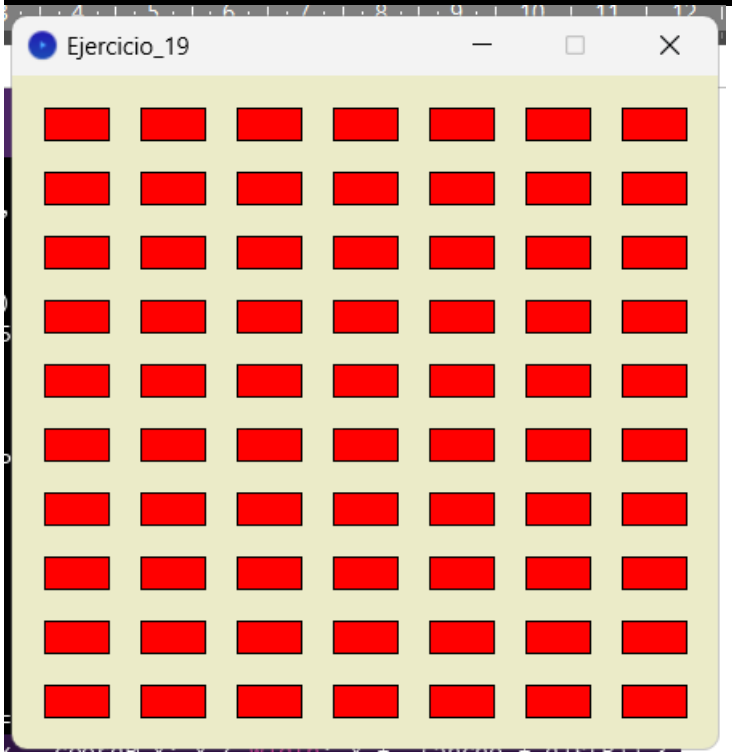
¿Cual es el proceso que resuelve?: dibujar un lienzo (440, 420) y dentro de el rectangulos separados por una distancia de 20 pixeles ya sea vertical como horizontalmente.

Diseño:

Entidad que resuelve el problema: lienzo
Variables: • **coordenadasRect: float //**almacena un valor de coordenadas • **ancho, alto, distanciaEntreRect : int //almacena un valor entero** • **anchoLienzo, altoLienzo: int //almacenan valores enteros**
Nombre del Algoritmo: rectangulos_repetidos
Proceso del algoritmo: *inicio* *anchoLienzo ← 440* *altoLienzo ← 420* *ancho ← 40* *alto ← 20* *distanciaRect ← 20* *para x ← coordenadasRect.x hasta anchoLienzo con paso (ancho+distanciaEntreRect)* *hacer* *para y ← coordenadasRect.y hasta altoLienzo con paso (alto+distanciaEntreRect)* *hacer* *dibujar rectangulo en (x,y,ancho,alto)* *fin_para* *fin_para* *fin*

Ejercicio 20

```
PVector coordR;
int alto, ancho, distR;

void setup() {
  size(440, 420);
  background(235,235,200);
  distR = 20;
  ancho = 40;
  alto = 20;
  coordR = new PVector(distR, distR);
}

void draw() {
  dibujarR();
}

void dibujarR() {
  for (float y = coordR.y; y < height; y += (alto + distR)) {
    for (float x = coordR.x; x < width; x += (ancho + distR)) {
     rect(x, y, ancho, alto);
     fill(255,0,0);
    }
  }
}
```



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.

Análisis:

Datos de Entrada: puntoA, puntoB, puntoC, puntoD, distancia

Datos de Salida: una imagen que consiste en escalones con puntos de color rojo en los bordes.

Proceso:

¿Quién debe realizar el proceso?: El programa, mediante el código en Processing.

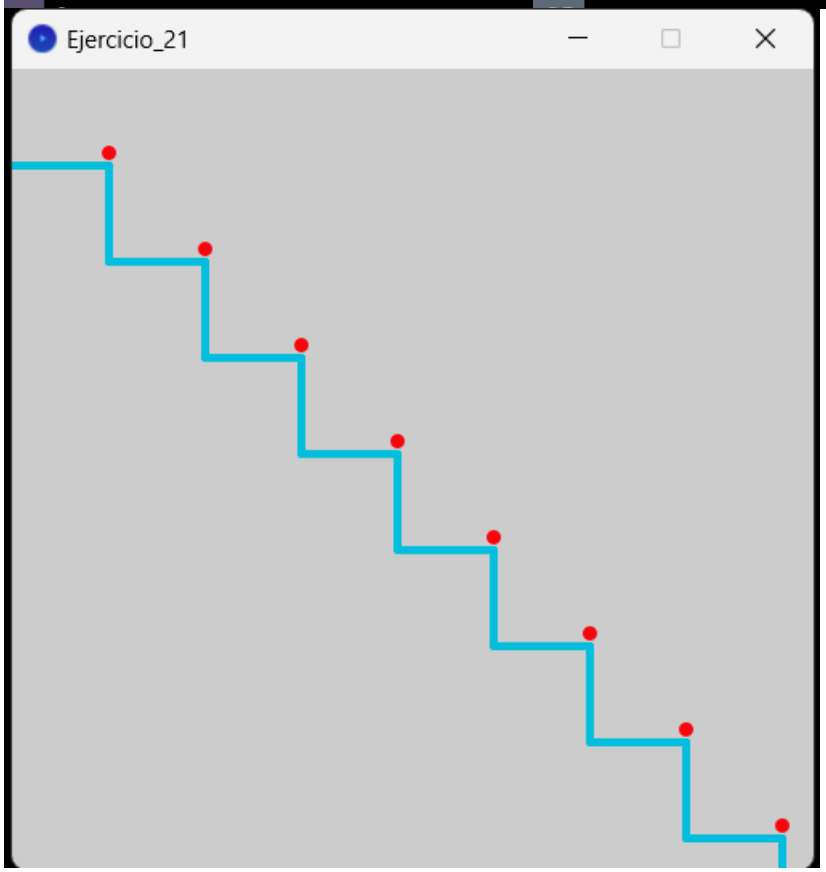
¿Cuál es el proceso que resuelve?: El proceso consiste en iterar mediante while() para dibujar escalones y puntos rojos en los bordes

Diseño:

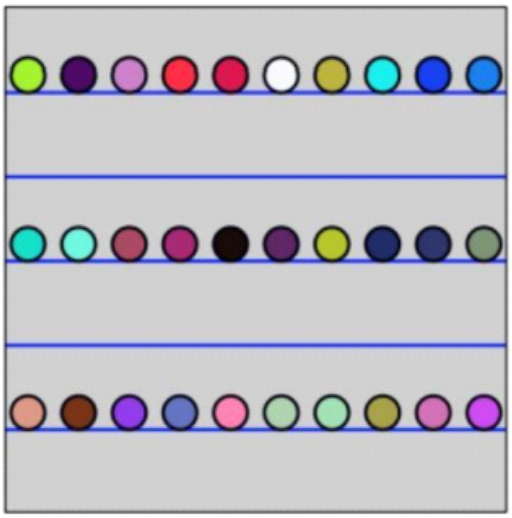
Entidad que resuelve el problema: programa
Variables: • **p1,p2,p3,p4: int** //almacena un vector • **d : int //almacena un valor entero**
Nombre del Algoritmo: escalones_puntos
Proceso del algoritmo: *Inicio* *anchoLienzo ← 500* *altoLienzo ← 500* *d ← 60* *mientras (p1.y sea menor o igual que anchoLienzo) Hacer* *dibujar línea horizontal en (p1.x, p1.y, p2.x, p2.y)* *dibujar línea vertical en (p2.x, p2.y, p3.x, p3.y)* *dibujar circulo en (p4.x, p4.y)* *p1.x ← p3.x* *p1.y ← p3.y* *fin_mientras* *Fin*

Ejercicio 21

```
int d;
PVector p1, p2, p3, p4;

public void setup (){
  size(500,500);
  d=60;
  p1 = new PVector(0,d);
  while(p1.y <= height){
  escalon();
  circulo();
  repeticion();
  }
}

public void escalon(){
  stroke(#00BEDE);
  strokeWeight(5);
  p2 = new PVector(p1.x+d, p1.y);
  line(p1.x, p1.y,p2.x,p2.y);
  p3 = new PVector(p2.x,p2.y+60);
  line(p2.x,p2.y,p3.x,p3.y);
}

public void circulo(){
  stroke(#FC030B);
  strokeWeight(9);
  p4 = new PVector(p2.x, p2.y-8);
  point(p4.x,p4.y);
}

public void repeticion(){
  p1.x = p3.x;
  p1.y = p3.y;
}
```



Ejercicio 21: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.

Análisis:

Datos de Entrada: lineas y circulos

Datos de Salida: círculos con colores randoms sobre líneas con un color con distanciamiento por medio

Proceso:

¿Quien debe realizar el proceso?: la computadora

¿Cual es el proceso que resuelve?: dibujar 5 lineas y en 3 de ellas dibujar sobre ellas circulos con colores aleatorios o randoms

Diseño:

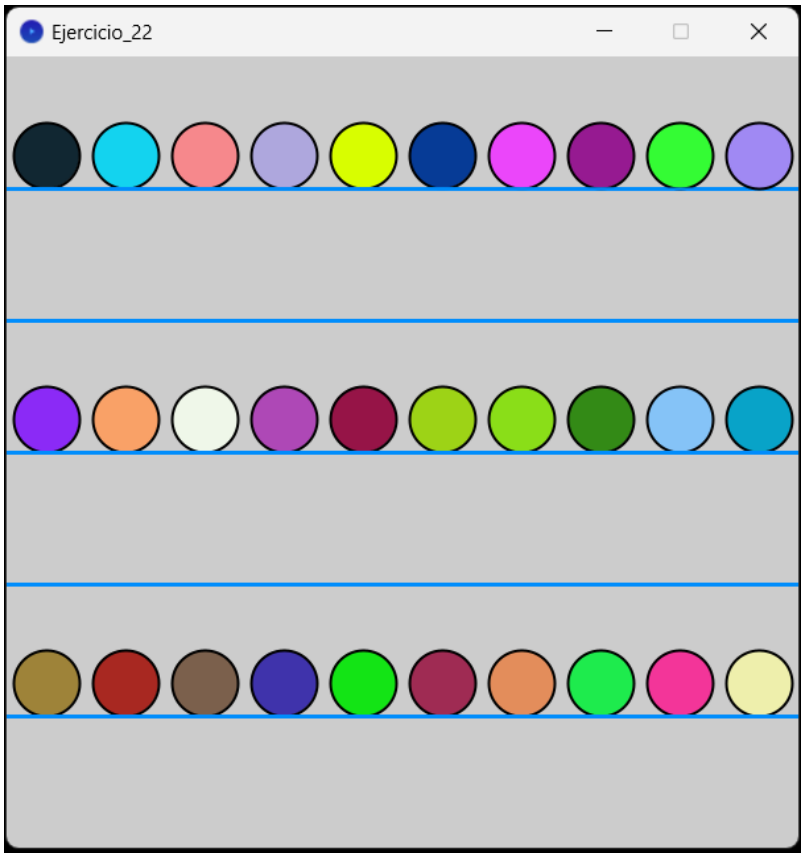
Entidad que resuelve el problema: processing
Variables: • **x, y, circuloX, circuloY, d : int //almacena un valor entero** • **anchoLienzo, altoLienzo: int //almacenan valores enteros**
Nombre del Algoritmo: rectangulos_repetidos

Proceso del algoritmo:

inicio
anchoLienzo ← 600
altoLienzo ← 600
x ← 0
y ← 100
dist ← 30;
circuloY ← 75
hacer
circuloX ← dist
hacer
dibujar linea en (x, y, anchoLienzo, y)
dibujar circulo en circuloX, circuloY, 50, 50)
*circuloX ← circuloX + dist*2*
fin_hacer
mientras*(circuloX sea menor que ancholienzo)*
y ← y + 100;
circuloY ← circuloY + 200;
fin_hacer
mientras *(y sea menor que altoLienzo)*
fin

Ejercicio 22

```
1  void setup(){
2    size(600,600);
3    int x = 0, y = 100, circuloY = 75, dist = 30;
4    do{
5      int circuloX = dist;
6    do{
7      stroke(#008DFC);
8      line(x,y,width,y);
9      fill(random(255), random(255), random(255));
10     stroke(0);
11     strokeWeight(2);
12     ellipse(circuloX,circuloY,50,50);
13     circuloX += dist*2;
14 }while(circuloX < width);
15     y += 100;
16     circuloY += 200;
17 }while(y < height);
18 }
```


Ejercicio_22