FACULTAD DE INGENIERA - UNIVERSIDAD NACIONAL DE JUJUY

ABRIL DE 2024

TRABAJO PRACTICO N°1

FUNDAMENTOS DE PROGRAMACION ORIENTADA A OBJETOS

TEMA: OPERADORES - METODOLOGIA DE LA PROGRAMACION

PREPARADO Y PRESENTADO POR

BONIFACIO PAOLA DEL MILAGRO
DNI: 46525675
LIBRETA: TUV000649

ejercicio 1

(3*A)-(4*B/(A^2))
6-(4*B/4)
6-5
1

ejercicio 1

```
1  int A=2, B=5;
2
3  float resultado = 3* A - 4 * B / pow(A,2);
4
5  println(resultado);
6
7
8
```

ejercicio 2

4 / 2 * 3 / 6 + 6 / 2 / 1 / 5 ^ 2 / 4 * 2
4 / 2 * 3 / 6 + 6 / 2 / 1 / 25 / 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 2

```
1  float resultado = 4.0/2*3/6+6/2/1/pow(5,2)/4*2 ;
2
3  println(resultado);
4
5
6
```

ejercicio 4

a) $b^2 - 4.a.c$

b) $3.(X^4) - 5.X^3 + X12 - 17$

c) $\frac{b+d}{c+4}$

d) $\sqrt{x^2 + y^2}$

ejercicio 4

```
int a=1, b=2, c=3, d=4, X=5, x=6, y=8;

float resultadoA = pow(b, 2) - 4 * a * c;

float resultadoB = 3 * pow(X, 4) - 5 * pow(X, 3)+ X * 12 - 17;

float resultadoC = (b + d) / (c + 4);

float resultadoD = pow(pow(x, 2)+ pow(y, 2), 0.5);

println(resultadoA);
println(resultadoB);
println(resultadoC);
println(resultadoD);
```

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
   20-B^0.5*C
   20-B^0.5
   20-2.23
   17,76

b) A*B/3^2
   20/9
   2.22

c) B+C/2*A+10*3*B–6
   6/2*A+10*3*B-6
   3*A+10)*3*B-6
   12+10*3*B-6
   22*3*B-6
   66*B-6
   330-6
   324

ejercicio 5

```
int A=4, B=5, C=1;

float resultadoA= B * A - pow(B, 2) / A * C;

float resultadoB= (A * B) / pow(3, 2);

float resultadoC= (((B + C) / 2 * A + 10) * 3 * B) - 6 ;

println(resultadoA);

println(resultadoB);

println(resultadoC);
```

ejercicio 6

Para x=3, y=4; z=1, evaluar el resultado de
R1 = y+z
R1 = 5
R2 = x >= R1
R2 = (3 >= 5)
R2= false

ejercicio 6

```
int x=3, y=4, z=1;

int R1= y+z;
boolean R2 = x >= R1;

println("resultado de R1: " + R1);
println("resultado de R2: " + R2);
```

ejercicio 7

Para contador1=3, contador2=4, evaluar el resultado de
R1 = ++ contador1
R1= 4
R2 = contador1 < contador2
R2= 4<4
R2=false

ejercicio 7

```
int contador1= 3, contador3= 4;

int R1 = ++contador1;
boolean R2 = contador1<contador3;

println("resultado de R1: "+R1);
println("resultado de R2: "+R2);
```

ejercicio 8

Para a=31, b=-1; x=3, y=2, evaluar el resultado de
a+b-1 < x*y
31-1<6
30<6
false

ejercicio 8

```
int a=31, b=-1, x=3, y=2;

boolean resultado = a + b - 1 < x * y ;

println("El resultado es " +resultado);
```

ejercicio 9

Para x=6, y=8, evaluar el resultado de
!(x<5)&& !(y>=7)
!(false) and !(true)
(true) and (false)
False

ejercicio 9

```
int x=6, y=8;

boolean resultado= !(x<5)&& !(y>=7);

println(resultado);
```

ejercicio 10

Para i=22,j=3, evaluar el resultado de
!((i>4) || !(j<=6))
!(true or !(true))
!(true or false)
!(true)
false

ejercicio 10

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

ejercicio 11

Para a=34, b=12,c=8, evaluar el resultado de
!(a+b==c) || (c!=0)&&(b-c>=19)
!(46==c) or (true) and (4>=19)
!(false) or (true) and (false)
true or (true and false)
true OR false
true

ejercicio 11

```
1  int a=34, b=12, c=8;
2
3  boolean resultado = !(a+b==c) || (c!=0)&&(b-c>=19);
4
5  println(resultado);
```

ejercicio 12

Definición del problema: mostrar un saludo con el nombre del usuario

Análisis:

- Datos de Entrada:
  -nombre del usuario: string
- Datos de Salida:
  -saludo con nombre del usuario: string

- Proceso:
  ¿Quién debe realizar el proceso?: el usuario con el programa

  ¿Cuál es el proceso que realiza?:
  Recibir el nombre del usuario para realizar un saludo con su nombre

Diseño:

Entidad: usuario
Variables: - nombreUsuario: string - saludo: string
Nombre de Algoritmo: saludar_usuario Proceso del Algoritmo: Inicio Importar JOptionPane del paquete Java *Leer* nombre *Mostrar* saludo en (“Hola ”+ nombre) fin

ejercicio 12

```
1 import javax.swing.JOptionPane;
2
3 String nombre = JOptionPane.showInputDialog ("EScriba su nombre");
4
5 print("Hola " + nombre);
6
```

ejercicio 13

Definición del problema: calcular el perímetro y área de un rectángulo

Analisis:

- Datos de Entrada:
  -base: int
  -altura: int
- Datos de Salida:
  -perimetro: int
  -area: int

- Proceso:
  ¿Quién debe realizar el proceso?: el programa

  ¿Cuál es el proceso que realiza?:
  Calcular el perímetro y área de un rectángulo con la base y altura dadas

Diseño:

Entidad: programa
Variables: - base: int - altura: int - perímetro: int - area: int

Nombre de Algoritmo: calcular_rectangulo Proceso del Algoritmo: Inicio importar JOptionPane del paquete Java *Leer* base *Leer* altura *Mostrar* perímetro del rectángulo igual a (base*2+altura*2) *Mostrar* área del rectángulo igual a (base*altura) fin

ejercicio 13

```
import javax.swing.JOptionPane;
int ladoA, ladoB;

ladoA = int(JOptionPane.showInputDialog ("Ingresar lado A"));
ladoB = int(JOptionPane.showInputDialog ("Ingresar lado B"));

int perimetro = 2*(ladoA +ladoB);

print("El perimetro del rectangulo es " + perimetro);
```

ejercicio 14

Definición del problema: obtener hipotenusa de un triángulo

Análisis:

- Datos de Entrada:
  -cateto1: int
  -cateto2: int
- Datos de Salida:
  -hipotenusa: float

- Proceso:
  ¿Quién debe realizar el proceso?: el programa

  ¿Cuál es el proceso que realiza?:
  Calcular la hipotenusa de un triangulo con sus dos catetos dados

Diseño:

Entidad: programa
Variables: - cateto1: int - cateto2: int - hipotenusa: float
Nombre de Algoritmo: calcular_hipotenusa Proceso del Algoritmo: Inicio

importar JOptionPane del paquete Java
Leer cateto1
Leer cateto2
hipotenusa raiz de (cateto1^2 + cateto2^2)
Mostrar hipotenusa del triangulo
fin

ejercicio 14

```
1  import javax.swing.JOptionPane;
2  int catetoA, catetoO;
3
4  catetoA = int (JOptionPane.showInputDialog ("ingrese el cateto adyacente "));
5  catetoO = int (JOptionPane.showInputDialog ("ingrese el cateto opuesto "));
6
7  float hipotenusa = sqrt (pow(catetoA, 2) + pow(catetoO, 2));
8
9  print("la hipotenusa del triangulo rectangulo es " + hipotenusa);
10
```

ejercicio 15

Definicion del problema: calcular la suma, resta, multiplicacion y división de dos números dados

Análisis:

- Datos de Entrada:
  -numeroA: int
  -numeroB: int
- Datos de Salida:
  -resultadoSuma: float
  -resultafoResta: float
  -resultadoMulti: float
  -resultadoDiv: float

- Proceso:
  ¿Quién debe realizar el proceso?: el programa

  ¿Cuál es el proceso que realiza?:
  Calcular la suma, resta, multiplicación y división de dos números dados

Diseño:

Entidad: programa
Variables: - numeroA: int - numeroB: int - resultadoSuma: float - resultafoResta: float - resultadoMulti: float - resultadoDiv: float

Nombre de Algoritmo: calcular_resultado

Proceso del Algoritmo:
Inicio
importar JOptionPane del paquete Java
Leer numeroA
Leer numeroB
resultadoSuma (numeroA+numeroB)
resultafoResta (numeroA-numeroB)
resultadoMulti (numeroA*numeroB)
resultadoDiv (numeroA/numeroB)
Mostrar resultadoSuma
Mostrar resultadoResta
Mostrar resultadoMulti
Mostrar resultadoDiv
fin

ejercicio 15

```
1  import javax.swing.JOptionPane;
2  int numero1, numero2;
3
4  numero1= int(JOptionPane.showInputDialog("Escribir el primer numero"));
5  numero2= int(JOptionPane.showInputDialog("Escribir el segundo numero"));
6
7  float suma = numero1 + numero2;
8  float resta = numero1 - numero2;
9  float multiplicacion = numero1 * numero2;
10 float division = float(numero1) / numero2;
11
12 println ("La suma es igual a: "+suma);
13 println("La resta es igual a: " + resta);
14 println("La multiplicacion es igual a: "+ multiplicacion);
15 if (numero2 !=0)
16 {
17   println("La divison es igual a: "+ division);
18 } else
19 {
20   println("no se puede realizar division entre 0");
21 }
```

ejercicio 16

Definición del problema: convertir temperatura Fahrenheit en grados Celsius

Análisis:

- Datos de Entrada:
  -gradoFah: int
- Datos de Salida:
  -gradoCel: float

- Proceso:

¿Quién debe realizar el proceso?: el programa

¿Cuál es el proceso que realiza?:
Convertir los grados Fahrenheit dados en grados Celsius

Diseño:

Entidad: programa
Variables: - gradoFah: int - gradoCel: float
Nombre de Algoritmo: transformar_celsius Proceso del Algoritmo: inicio importar JOptionPane del paquete Java *Leer* gradoFah gradoCel (gradoFah-32) / 1.8 *Mostrar* gradoCel fin

```
ejercicio 16
1  import javax.swing.JOptionPane;
2
3  int temperaturaF;
4
5  temperaturaF= int (JOptionPane.showInputDialog("Ingrese la temperatura en Fahrenheit para convertirla en grados Celsius"));
6
7  float gradosC= (temperaturaF -32) /1.8;
8
9  println("La temperatura en grados es:"+ gradosC);
10
```

ejercicio 17

Definición del problema: calcular la distancia entre Link y la caja

Análisis:

- Datos de Entrada:
  -ancho, alto de Lienzo: int
  -coordenadasLink: coordenadas cartesianas
  -coordenadasCaja: coordenadas cartesianas
  -ancho, alto de Link: int
  -ancho, alto de Caja: int
  -catetoA, catetoO: coordenadas cartesianas
  -colorLink: color
  -colorCaja: color

- Datos de Salida:
  -distanciaLinkCaja: float

- Proceso:
  ¿Quién debe realizar el proceso?: el programa

  ¿Cuál es el proceso que realiza?:
  Colocar en un lienzo un cuadrado estatico y un circulo controlado por el mouse que dependiendo la posición de este se calcule la distancia entre ambos

Diseño:

Entidad: programa
Variables: - anchoLienzo, altoLienzo: int - coordenadasLink: coordenadas cartesianas - coordenadasCaja: coordenadas cartesianas - anchoLink, altoLink: int - anchoLink, altoCaja: int - catetoA, catetoO: coordenadas cartesianas - colorLink: color - colorCaja: color - distanciaLinkCaja: float
Nombre de Algoritmo: calcular_distanciaLinkCaja Proceso del Algoritmo: inicio anchoLienzo□ 400 altoLienzo□ 400 coordenadaXCaja□ 200 coordenadaYCaja□ 300 altoLink□20 anchoLink□20 altoCaja□20 anchoCaja□20 catetoA□ xCaja – xLink catetoB□ yCaja – yLink distanciaLinkCaja□ raiz de (catetoA^2 + catetoB^2) *Leer* coordenadasLink *Mostrar* distanciaLinkCaja *fin*

ejercicio 17

```
float ylink, xlink;
float ycaja, xcaja;

void setup()
{
  size(400, 400);

  xlink= 100;
  ylink= 100;

  xcaja= 200;
  ycaja= 300;
}

void draw()
{
  background(255);

  float catetoA = xcaja - xlink;
  float catetoO = ycaja - ylink;

  float distancia = sqrt(pow(catetoA, 2) + pow(catetoO, 2));

  fill(255, 0, 0);
  ellipse(xlink, ylink, 20, 20);

  fill(0, 0 , 255);
  rectMode(CENTER);
  rect(xcaja, ycaja, 20, 20);

  println("La distancia entre Link y la caja de tesoro es: " + distancia);
}

void mouseMoved()
{
  xlink = mouseX;
  ylink = mouseY;
}
```

ejercicio 18

Definición del problema: crear un algoritmo que resuelva raíces de ecuaciones de segundo grado y analize su discriminante

Análisis:

- Datos de Entrada:
  -numeroA: float
  -numeroB: float
  -numeroC:float
- Datos de Salida:
  -resultadoEcua:
  -discriminante:

- Proceso:

¿Quién debe realizar el proceso?: el programa

¿Cuál es el proceso que realiza?:
Calcular la ecuación de segundo grado dada y analizar su discriminante

Diseño:

Entidad: programa
Variables: - numeroA: float - numeroB: float - numeroC: float - resultafoEcua: float - discriminante: float
Nombre de Algoritmo: calcular_ecuacion Proceso del Algoritmo: inicio importar JOptionPane del paquete Java *Leer* numeroA *Leer* numeroB *Leer* numeroC discriminante□numeroB^2-4*numeroA*numeroC si la discriminante es mayor a cero se realizará x1□(-numeroB+raiz del(discriminante))/(2*a) x2□(-numeroB–raiz del(discriminante))/(2*a) *Mostrar* las raices reales y distintas x1, x2 si la discriminante es igual a cero se realizará x□-numeroB/(2*a) *Mostrar* la raíz real e igual x si no cumple ninguna de las condiciones anteriores se realizará *Mostrar* las raíces son complejas *fin*

ejercicio 18

```
import javax.swing.JOptionPane;
float a, b, c;

a= float(JOptionPane.showInputDialog("Escribir la variable a"));
b= float(JOptionPane.showInputDialog("Escribir la variable b"));
c= float(JOptionPane.showInputDialog("Escribir la variable c"));

float discriminante = pow(b, 2) - 4*a*c;
println("la discriminante es igual: " + discriminante);

if (discriminante > 0){
 float x1= (-b + sqrt(discriminante)) / (2 * a);
 float x2= (-b - sqrt(discriminante)) / (2 * a);
 print("Las raices son reales y distintas: " + "x1="+ x1 + " x2="+ x2);
}
else if (discriminante == 0) {
 float x= -b/(2 * a);
 print("las raices son reales e iguales: " + "x= "+ x);
}
else {
  print("las raices son complejas");
}
```

ejercicio 19

Definición del problema: dibujar una línea que toque la parte superior de un circulo y hacer que los dos se muevan juntos indefinidamente de arriba a abajo de los bordes del lienzo

Análisis:

- Datos de Entrada:
  - anchoLienzo, altoLienzo:
  -coordenadaLink:.
  -coordenadaObje:

- Datos de Salida:
  -movimiento de circulo con linea

- Proceso:
  ¿Quién debe realizar el proceso?: el programa

  ¿Cuál es el proceso que realiza?:
  Calcular la ecuación de segundo grado dada y analizar su discriminante

Diseño:

Entidad: programa
Variables: - numeroA: float - numeroB: float

- numeroC: float - resultadoEcua: float - discriminante: float
Nombre de Algoritmo: calcular_ecuacion Proceso del Algoritmo: inicio fin

ejercicio 19

```
1  float yLinea, yElipse, velocidad = 2 ;
2  int direccionLinea = 1, direccionElipse = 1;
3
4  void setup(){
5   size(400, 400);
6   yLinea = height / 2;
7   yElipse = 240;
8  }
9
10 void draw() {
11  background(0);
12  fill(0, 50, 255);
13  ellipse(width/2, yElipse, 80, 80);
14  stroke(255);
15  line(0, yLinea, width, yLinea);
16  yLinea += direccionLinea * velocidad;
17
18  if (yLinea >= height || yLinea <= 0) {
19    direccionLinea *= -1;
20  }
21
22  yElipse += direccionElipse * velocidad;
23
24  if (yElipse >= height || yElipse <= 0){
25    direccionElipse *= -1;
26  }
```

ejercicio 20

Definición del problema: dibujar una serie de rectángulos idénticos en un lienzo

Análisis:

- Datos de Entrada:
  - coordenadasRect: coordenadas cartesianas
  -ancho, alto, distanciaEntreRect: enteros
  -anchoLienzo, altoLienzo: enteros
  -rect_color: color

- Datos de Salida:
  -rectangulos dibujados

- Proceso:
  ¿Quién debe realizar el proceso?: el programa

  ¿Cuál es el proceso que realiza?:
  Dibujar una serie de rectángulos con determinado espaciado y color entre ellos en un lienzo con un determinado tamaño

Diseño:

Entidad: programa
Variables: - coordenadasRect: coordenadas - ancho, alto, distanciaEntreRect: enteros - anchoLienzo, altoLienzo: enteros - rect_color: color
Nombre de Algoritmo: dibujar_rectangulos Proceso del Algoritmo: danchoLienzo□440 altoLienzo□420 distanciaEntreRect□20 ancho□ 40 alto□20 color□color(255, 165, 0) para x□coordenadasRect.x hasta anchoLienzo con paso (ancho+distanciaEntreRect) hacer para y□coordenadasRect.y hasta altoLienzo con paso (alto+distanciaEntreRect) hacer rellenar con color los rectangulos dibujar un rectángulo en (x.coordenadasRect.y) con dimensiones ancho y alto fin para fin para

ejercicio 20

```
PVector coordenadasRect;
int ancho, alto, distEntreRect;
color Rect_color;

public void setup(){
   size(440,420);
   distEntreRect = 20;
   ancho=40;
   alto=20;
   coordenadasRect = new PVector(distEntreRect,distEntreRect);
   Rect_color = color(255, 165, 0);
}

public void draw(){
  dibujarRectangulos();
}

public void dibujarRectangulos(){
  for(float x=coordenadasRect.x;x<width;x+= (ancho+distEntreRect)){
    for(float y=coordenadasRect.y;y<height;y+=(alto+distEntreRect)){
      fill(Rect_color);
      rect(x,y,ancho,alto);
    }
    }
}
```

ejercicio 21

Definición del problema: dibujar escalones sobre el lienzo y colocar sobre cada escalón un punto rojo

Análisis:

- Datos de Entrada: coordenadas cartesianas en 2D
  -puntoA
  -puntoB
  -puntoC
  -puntoD

- Datos de Salida:
  -EL dibujo en la línea horizontal
  -El dibujo en la línea vertical
  -El dibujo del punto rojo

- Proceso:
  Dibujar una linea horizontal entre los puntos A y B, con distancia igual a distLinea
  Dibujar una linea vertical entre los puntos B y C, con distancia igual a distLinea
  Dibujar un punto en la siguiente posicion: x=posicion en x de B, y =posición en y de B -5 unidades
  Actualizar las coordenadas de punta con las de puntoC
  Repetir desde el principio hasta que la coordenada en y de puntoA sea mayor que el alto del lienzo

Diseño:

Entidad: Escalon

Variables: coordenadas cartesianas en 2D

- puntoA
- puntoB
- puntoC
- puntoD
- distLinea: entero

Nombre de Algoritmo: dibujar_escalon
Proceso del Algoritmo:
inicio
dibujar una linea horizontal entre los puntos A y B, con distancia distLinea
dibujar una linea vertical entre los puntos B y C, con distancia distLinea
dibujar_circulo
fin

Nombre algoritmo: dibujar_circulo
Proceso del Algoritmo:
inicio
dibujar un punto en la siguiente posicion: x= posicion en x de B, y = posicion en y de B - 10
fin

Nombre algoritmo: actualizar_coordenadas_A
Proceso del Algoritmo:
inicio
puntoA.x puntoC.x
puntoA.y puntoC.y
fin

ejercicio 21

```
1  PVector puntoA, puntoB, puntoC, puntoD;
2  int distLinea;
3
4  public void setup(){
5    size(500,500);
6    distLinea = 60;
7    puntoA = new PVector(0,distLinea);
8
9    while(puntoA.y < height){
10      dibujarEscalon();
11      actualizarCoordenadasA();
12   }
13
14
15 }
16
17 public void dibujarEscalon(){
18
19   stroke(#1AEAFF);
20   strokeWeight(4);
21   puntoB = new PVector(puntoA.x+distLinea,puntoA.y);
22   line(puntoA.x,puntoA.y,puntoB.x,puntoB.y);
23   puntoC = new PVector(puntoB.x,puntoB.y+distLinea);
24   line(puntoB.x,puntoB.y,puntoC.x,puntoC.y);
25   dibujarPunto();
26 }
```

```
26 }
27
28 public void dibujarPunto(){
29  stroke(255,0,0);
30  strokeWeight(10);
31  puntoD = new PVector(puntoB.x,puntoB.y-10);
32  point(puntoD.x,puntoD.y);
33 }
34
35 public void actualizarCoordenadasA(){
36  puntoA.x = puntoC.x;
37  puntoA.y = puntoC.y;
38 }
```

ejercicio 22

Análisis:

Definición del problema: 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 aleatorio.

Datos de entrada: distanciaEntreRect, alto, ancho: enteros

Datos de salida:

Líneas horizontales separadas entre si

Puntos sobro líneas impares con colores aleatorios

Proceso:

Dibujar rectas en una posición 100 respecto a la ubicación Y del lienzo

Dibujar circulos de igual tamaño y separados entre ellos en mismas distancias con un color aleatorio en cada uno

Replicar las rectas y círculos hasta acabar el lienzo, evitando que los círculos aparezcan en las rectas pares

Diseño:

Entidad: lienzo
VARIABLES coordenadasRect: coordenadas ancho, alto, distanciaRect: enteros anchoLienzo, altoLienzo: enteros
Nombre de Algoritmo: dibujar_rectangulos Proceso de Algoritmo Inicio anchoLienzo=440 altoLienzo=420 distanciaRect=20 ancho=40

alto=20
hacer circulo=distanciaCirculo
hacer línea(lineaX, lineaY, ancho, lineaY)
circulo(circuloX, circuloY, 50, 50)
Mientras(circulo < ancho)
lineaX += 100
circuloY += 200;
Mientras(lineaY < alto)

Fin

ejercicio 22

```
 do{
   int circuloX = distanciaCirculo;

do{
  stroke(#008DFC);
  line(lineaX, lineaY, width, lineaY);
  fill(random(255), random(255), random(255));
  stroke(0);
  strokeWeight(2);
  ellipse(circuloX, circuloY, 50, 50);
  circuloX += distanciaCirculo*2;
}
while(circuloX < width);
lineaY += 100;
circuloY += 200;
 }
 while(lineaY < height);
}
```