1.Elaborar un algoritmo para calcular e imprimir el precio de un terreno del cual se tienen los siguientes datos: largo, ancho y precio por metro cuadrado. Si el terreno tiene más de 400 metros cuadrados se hace un descuento de 10 %.

1. DIAGRAMA DE CLASES

Modelo

|  |
| --- |
| Ejercicio 1 |
| -ancho-  -largo  -costo  -costoT |
| +establecerancho ()  +establecerlargo ()  +establecergetcosto()  +costoTotal ()  +obtenercostoT () |

|  |
| --- |
| Ejecuta1 |

Controlador

Vista?

1. ANALISIS DEL PROBLEMA

Detectar objetos -> Ejercicio1

Entradas -> acho

largo

costo

Procesos -> costoTotal = (largo\*ancho) \*costo

costoTotal = ((largo\*ancho) \*costo) -(largo\*ancho) \*costo\*0,1)

Salidas -> Presentar: costoTotal

1. DISEÑO DEL PROGRAMA
2. Diagrama de clase
3. Pseudocódigo o miniespecificación (Algoritmo)

Algoritmo Calcular le precio Total del terreno

Clase Ejercicio1

1. Declarar datos

ancho: double

costo: double

tamaño: double

largo: double

precio: double

1. Método establecerAncho (ancho: double)
2. ancho: ancho
3. Fin Método establecerAncho
4. Método establecerLargo (largo: double)
5. largo: largo
6. Fin Método establecerLargo
7. Método establecerCosto (costo: double)
8. costo: costo
9. Fin Método establecerCosto
10. Método establecerTamaño(tamaño: double)
11. tamaño: tamaño
12. Fin Método establecerTamaño
13. Método establecerPrecio (precio: double)
14. precio: precio
15. Fin Método establecerPrecio
16. Método calcularCostoTotal ()
17. If tamaño <= 40 then
18. precio = tamaño \* costo;
19. Else
20. precio = (tamaño \* costo) - ((tamaño \* costo) \* 0.1);
21. endif
22. Fin del Método calcularCostoTotal

Fin Clase Ejeercicio1

Clase Ejecuta1

1. Método Principal ()
2. Declarar, crear e iniciar objeto

Ejercicio1 Ejercicio\_1 = new Ejercicio1 ()

1. Solicitar ancho del terreno
2. Leer ancho
3. Solicitar largo del terreno
4. Leer largo
5. Solicitar costo del metro del terreno
6. Leer costo
7. Establecer Ejercicio\_1. calcularCostoTotal ()
8. Imprimir Ejercicio\_1. obtenercostoT ()
9. Fin Método principal

Fin Clase Ejecuta1

1. DIAGRAMA DE CLASES

Modelo

|  |
| --- |
| Ejercicio 4 |
| -cantidad  -precio  -costoT |
| +establecercantidad ()  +establecerprecio ()  +establecergetcostoT ()  +costoTotal ()  +obtenercostoT () |

|  |
| --- |
| Ejecuta4 |

Controlador

Vista?

1. ANALISIS DEL PROBLEMA

Detectar objetos -> Ejercicio4

Entradas -> cantidad

precio

Procesos -> costoT=precio\*cantidad

costoT=((precio\*cantidad) -((precio\*cantidad) \*0.15));

Salidas -> Presentar: costoTotal

1. DISEÑO DEL PROGRAMA
2. Diagrama de clase
3. Pseudocódigo o miniespecificación (Algoritmo)

Algoritmo Calcular le precio Total por los productos

Clase Ejercicio4

1. Declarar datos

cantidad: entero

precio: double

costoT: Double

1. Método establecerCantidad (cantidad: entero)
2. cantidad: ancho
3. Fin Método establecerCantidad
4. Método establecerPrecio (precio: double)
5. precio: precio
6. Fin Método establecerPrecio
7. Método establecerCostoT (costoT: double)
8. costo: costoT
9. Fin Método establecerCostoT
10. If tamaño <= 50 then
11. costoT=precio\*cantidad
12. Else
13. costoT=((precio\*cantidad) -((precio\*cantidad) \*0.15));
14. endif
15. Fin del Método calcularCostoTotal

Fin Clase Ejeercicio4

Clase Ejecuta4

1. Método Principal ()
2. Declarar, crear e iniciar objeto

Ejercicio4 Ejercicio\_4 = new Ejercicio4 ()

1. Solicitar cantidad del producto
2. Leer cantidad
3. Solicitar precio del producto
4. Leer precio
5. Establecer Ejercicio\_4. calcularCostoTotal ()
6. Imprimir Ejercicio\_1. obtenercostoT ()
7. Fin Método principal

Fin Clase Ejecuta4

1. DIAGRAMA DE CLASES

Modelo

|  |
| --- |
| Ejercicio 7 |
| - nombre  -nota1  -nota2  -nota3  -notaT  -mensaje |
| +establecernombre ()  +establecernota1()  +establecernota2()  +establecernota3()  +NotaTotal ()  +obtenernotaT () |

|  |
| --- |
| Ejecuta7 |

Controlador

Vista?

1. ANALISIS DEL PROBLEMA

Detectar objetos -> Ejercicio7

Entradas -> nombre

nota1

nota2

nota3

Procesos -> notaT=((nota1+nota2+nota3) /3);

notaT=((nota1+nota2+nota3)/3);

Salidas -> Presentar: mensaje

notaT

1. DISEÑO DEL PROGRAMA
2. Diagrama de clase
3. Pseudocódigo o miniespecificación (Algoritmo)

Algoritmo Calcular la Nota del estudiante

Clase Ejercicio7

1. Declarar datos

nombre: Cadena

nota1: double

nota2: double

nota3: double

notaT: double

mensaje: String

1. Método establecerNombre (nombre: cadena)
2. nombre: nombre
3. Fin Método establecerNombre
4. Método establecerNota1 (nota1: double)
5. nota1: nota1
6. Fin Método establecerNota1
7. Método establecerNota2 (nota2: double)
8. nota2: nota2
9. Fin Método establecerNota2
10. Método establecerNota3(nota3: double)
11. nota3: nota3
12. Fin Método establecerNota3
13. Método NotaTotal ()
14. If nota1>70, nota>70, nota>nota3>70 then
15. mensaje=("Acreditado");
16. notaT=((nota1+nota2+nota3) /3);
17. Else
18. mensaje= (" No Acreditado");
19. notaT=((nota1+nota2+nota3) /3);
20. endif
21. Fin del Método NotaTotal

Fin Clase Ejeercicio4

Clase Ejecuta4

1. Método Principal ()
2. Declarar, crear e iniciar objeto

Ejercicio7 ejrcicio7= new Ejercicio 7 ()

1. Solicitar el nombre del estudiante
2. Leer nombre
3. Solicitar nota1
4. Leer nota1
5. Solicitar nota2
6. Leer nota2
7. Solicitar nota3
8. Leer nota3
9. Establecer ejercicio7. calcularnotaTotal ()
10. Imprimir ejercicio7. nombre ()
11. Imprimir ejercicio7.notaT
12. Fin Método principal

Fin Clase Ejecuta7

1. DIAGRAMA DE CLASES

Modelo

|  |
| --- |
| Ejercicio 8 |
| - triangulo  -nTriangulo |
| +establecertriangulo ()    +NombreTriangulos ()  +nTriangulo () |

|  |
| --- |
| Ejecuta8 |

Controlador

Vista?

1. ANALISIS DEL PROBLEMA

Detectar objetos -> Ejercicio8

Entradas -> triangulo

Procesos -> triangulo==90

triangulo>90&triangulo<180

triangulo>180

Salidas -> Presentar: nTriangulo

1. DISEÑO DEL PROGRAMA
2. Diagrama de clase
3. Pseudocódigo o miniespecificación (Algoritmo)

Algoritmo Calcular el nombre del triángulo según el ángulo ingresado

Clase Ejercicio8

1. Declarar datos

nTriangulo: Cadena

triangulo: int

1. Método establecerTriangulo (triangulo: int)
2. triangulo: triangulo
3. Fin Método establecerTriangulo
4. Método establecerNTriangulo (nTriangulo: double)
5. nTriangulo: nTriangulo
6. Fin Método establecerNTriangulo
7. Método establecerNota3(nota3: double)
8. nota3: nota3
9. Fin Método establecerNota3
10. Método NombreTriangulos ()
11. If (triangulo==90)
12. nTriangulo= ("Triangulo rectángulo");

1. Else
2. If (triangulo>90&triangulo<180)
3. nTriangulo= (" Triangulo Obtusángulo")
4. nTriangulo= (" Triangulo Acutángulo");

1. endif
2. Fin del Método NombreTriangulos

Fin Clase Ejeercicio8

Clase Ejecuta8

1. Método Principal ()
2. Declarar, crear e iniciar objeto

Ejercicio8 ejrcicio8= new Ejercicio 8 ()

1. Solicitar el grado del triangulo
2. Leer triangulo
3. Establecer ejercicio8.NombreTriangulos()
4. Imprimir ejercicio8. nTriangulo ()
5. Fin Método principal

Fin Clase Ejecuta8

1. DIAGRAMA DE CLASES

Modelo

|  |
| --- |
| Ejercicio 16 |
| -tGrado  -grados |
| +establecertGrado ()  +establecergrados ()    +CalcularGrados ()  +obtenernGrados1  +obtenerGrados2  +obtenernGrados3  +obtenercGrados1  +obtenerGrados2  +obtenercGrados3 |

|  |
| --- |
| Ejecuta16 |

Controlador

Vista?

1. ANALISIS DEL PROBLEMA

Detectar objetos -> Ejercicio16

Entradas -> tGrado

grados

Procesos -> (tGrado.equals("Fahrenheit")) {

nGrados = ("Celsius: ");

cGrados = (grados - 32 \* (1 / 9));

nGrados = ("Kelvin: ");

cGrados = (cGrados1 + 273);

nGrado = ("Rankine: ");

cGrados = (cGrados1 + 460);

Salidas -> Presentar: nGrados

cGrados

1. DISEÑO DEL PROGRAMA
2. Diagrama de clase
3. Pseudocódigo o miniespecificación (Algoritmo)

Algoritmo Calcular le precio Total del terreno

Clase Ejercicio1

1. Declarar datos

ancho: double

costo: double

tamaño: double

largo: double

precio: double

1. Método establecerTGrado (tGrado: cadena)
2. tGrado: tGrado
3. Fin Método TGrado
4. Método establecerGrados (grados: double)
5. grados: grados
6. Fin Método establcerGrados
7. Método CalcularGrados ()
8. if (tGrado.igua ("Fahrenheit"))
9. nGrados1 = ("Celsius: ");
10. cGrados1 = (grados - 32 \* (1 / 9));
11. nGrados2 = ("Kelvin: ");
12. cGrados2 = (cGrados1 + 273);
13. nGrados3 = ("Rankine: ");
14. cGrados3 = (cGrados1 + 460);
15. else
16. if (tGrado.equals("Celsius"))
17. nGrados1 = ("Fahrenheit: ");
18. cGrados1 = (grados \* (9 / 5) + 32);
19. nGrados2 = ("Kelvin: ");
20. cGrados2 = (grados + 273.15);
21. nGrados3 = ("Rankine: ");
22. cGrados3 = (grados \* (9 / 5) + 491.67);
23. else
24. if (tGrado.equals("Kelvin"))
25. nGrados1 = ("Fahrenheit: ");
26. cGrados1 = ((grados - 273.15) \* (9 / 5) + 32);
27. nGrados2 = ("Celcius: ");
28. cGrados2 = (grados - 273.15);
29. nGrados3 = ("Rankine: ");
30. cGrados3 = (grados \* 1.8);
31. else
32. if (tGrado.equals("Rankine ")) {
33. nGrados1 = ("Fahrenheit: ");
34. cGrados1 = (grados - 459.7);
35. nGrados2 = ("Celsius: ");
36. cGrados2 = (cGrados1 - 491.67) \* (5 / 9);
37. nGrados3 = ("Kelvin: ");
38. cGrados3 = (cGrados1 \* 5 / 9);

Fin del Método CalcularGrados

Fin Clase Ejeercicio16

Clase Ejecuta16

1. Método Principal ()
2. Declarar, crear e iniciar objeto
3. Ejercicio16 ejercicio16= new Ejercicio16 ()
4. Solicitar el Nombre del tipo de grado
5. Leer tGrado
6. Solicitar el número de grados
7. Leer Grados
8. Establecer Ejercicio\_1. calcularCostoTotal ()
9. Imprimir ejercicio16. nGrados1() +ejercicio16.cGrados1()
10. Imprimir ejercicio16. nGrados2() +ejercicio16.cGrados2()
11. Imprimir ejercicio16. nGrados3() +ejercicio16.cGrados3()
12. Fin Método principal

Fin Clase Ejecuta16