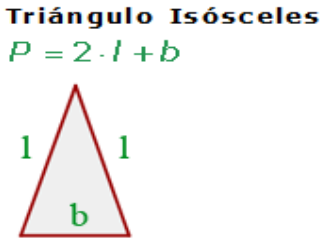
1.- Haz una clase para trabajar con triángulos isósceles (2 lados iguales). Dicha clase tendrá como atributos la base y la altura, que serán privados. También tendrá como mínimo un constructor y dos métodos para calcular el área y el perímetro de un triángulo, y todos aquellos métodos que sean necesarios para el buen funcionamiento del programa.

Fórmula a aplicar:



perimetro = 2 * lado + base, siendo lado la medida del lado repetido
lado^2 = (base/2)^2 + altura^2
lado = Raiz Cuadrada((base/2)^2 + altura^2)

Crea una clase Principal que cree un Array/ArrayList de triángulos en el que se realicen las siguientes operaciones:

1. Añadir un triángulo.
2. Calcular e imprimir el triángulo con el área más grande.
3. Calcular e imprimir el triángulo con el perímetro más pequeño.
4. Imprimir el área y el perímetro de un triángulo en concreto.
5. Imprimir la altura, la base, el área y el perímetro de todos los triángulos.

En las opciones 2 y 3 puedes hacerlo con un método que devuelva el objeto Triangulo que cumpla la condición, o bien con un método que devuelva la posición en el ArrayList del triángulo que cumpla la condición.

En la opción 4, pedimos al usuario la base y la altura como identificadores del triángulo que queremos mostrar.
Otra opción sería mostrar todos los triángulos (Triángulo n: base y altura) y pedir al usuario la posición en la lista del triángulo que se quiere visualizar.

2.- Dada la clase Viaje siguiente:

```
class Viaje {
        private String origen;
        private String destino;
        private double distancia;
}
```

Escribe un método denominado uneViaje que reciba como parámetros dos objetos tipo Viaje y devuelva un nuevo objeto de esa misma clase con:

- El origen del primer viaje y el destino del segundo viaje.

- La distancia será la suma de las distancias de los dos viajes originales, si el destino del primer viaje coincide con el origen del segundo, si no se cumple dicha condición se asignará -1 a la distancia.

Nota: La función para comparar cadenas es: cadena1.compareTo(cadena2) devuelve 0 si las dos cadenas son iguales, < 0 si la cadena1 < cadena2 y > 0 si cadena1 > cadena2.

Crea una clase Principal, en la que se defina un array/ArrayList de N objetos Viaje.

1. Imprime por pantalla el resultado de usar el método uneViaje, con una componente y la siguiente (1 y 2, 2 y 3, 3 y 4, …, 8 y 9). Visualiza cada viaje resultante de emparejar de dos en dos los viajes.

2. Realiza también la unión de todos los viajes para mostrar el origen y el destino final, así como los kilómetros totales.

Ejemplo para 3 viajes:

Entrada:

```
Viaje 1
Introduce origen del viaje
a
Introduce destino del viaje
b
Introduce kilómetros
40

Viaje 2
Introduce origen del viaje
b
Introduce destino del viaje
c
Introduce kilómetros
50

Viaje 3
Introduce origen del viaje
c
Introduce destino del viaje
d
Introduce kilómetros
10
```

Salida:

```
1.- Trayectos parciales dos a dos

Nuevo Viaje 1 y 2
Origen a
Destino c
Distancia 90.0

Nuevo Viaje 2 y 3
Origen b
Destino d
Distancia 60.0


2.- Trayecto total

Trayecto total
Origen a
Destino d
Distancia 100.0
```