TDA. Se desea mantener información de competidores de salto en alto. Sobre cada competidor se conoce dni, nombre, apellido, edad, ciudad de origen, peso y altura (en cm)) y la longitud máx llegada por ese competidor. Para determinar los tiempos y designar al ganador."

Especificar el diagrama UML para la clase *Competidor, explicitar categorías de los métodos.*

*I*mplementar la clase en un archivo java, los métodos:

1. *equals(...)* -dos competidores
2. *toString()*
3. *Dos métodos constructores*
4. *Dos métodos visualizadores*
5. *Dos métodos modificadores*
6. *Definir un método que dados dos valores (centímetros y peso) calcule el Índice de Masa corporal IMC = (peso / estatura^2) en kg y metros*

Puede ser un método estático? si/no por qué?

<Competidor>
<-> ENTERO dni, edad
<-> TEXTO nombre, apellido, ciudadOrigen
<-> REAL peso, altura, longitudMax

Contructores
<+> <competidor>(ENTERO dni, TEXTO nombre, TEXTO apellido, ENTERO edad, TEXTO ciudadOrigen, REAL peso, REAL altura, REAL longitudMax)
<+> <competidor>(ENTERO dni)

Observadoras
<+> getDni():ENTERO
<+> getNombre():TEXTO

Modificadoras
<+> setNombre(TEXTO nombre): VACIO
<+> setEdad(ENTERO edad): VACIO

Propias del tipo
<+> equals(Competidor objCompetidor): LOGICO
<+> toString(): TEXTO
<+> imcActual(): REAL //no estático
<+> imcEstimado(REAL peso, REAL altura): REAL // estático

2. Realice un ***pseudocódigo y código*** que, dado un arreglo de competidores.

   a. Realice un método que recorra el arreglo de competidores retorne otro arreglos con los competidores que tienen un IMC de bajo peso (< 18.5)
   (grupo 1)

   b. Realice un método que dado un IMC específico retorne si existe al menos un competidor con dicho IMC en el arreglo.
   (grupo 2)

2a.

```
MODULO ImcFiltro(REAL [] arregloCompetidores) RETORNA REAL[]
(*Dado un arreglo de competidores no nulo, retorna un arreglo nuevo con los competidores con imc menores a 18.5*)
        ENTERO longitudOriginal ← longitud(arreglo)
        ENTERO i←0, j←0
        REAL [] arregloNuevo ← CREAR REAL [longitudOriginal]

        PARA i←0 HASTA longitudOriginal-1 PASO 1 HACER
                SI (arreglo[i].imcActual < 18.5) ENTONCES
                        arregloNuevo[j] ← arreglo[i]
                        j++
                FIN SI
        FIN PARA
        RETORNA arregloNuevo
FIN MODULO
```

2b.

```
MODULO imcEspecifico (REAL [] arreglo, REAL imc) RETORNA LOGICO
(*Dado un arreglo de competidores no nulo y un IMC a buscar dentro de este, nos dice si existe al menos 1 dentro del arreglo*)
        LOGICO existencia ← falso
        ENTERO i ← 0
        ENTERO longitud ← LONGITUD(arreglo)-1

        REPETIR
                SI (arrelgo[i] = imc) ENTONCES
                        existencia ← verdadero
                FIN SI
                i++
        HASTA (i<longitud AND existencia = falso)
        RETORNA existencia
FIN MODULO
```

3. Dada un texto,

a. determine la cantidad de las palabras con más de 3 vocales diferentes.
Realice una traza para: “ Modularizar no es problematico.”, (debe retornar 2)

b. Determine todos las palabras que tienen al menos una letra igual pegada.
Realice una traza para el siguiente texto: “corriendo bajo la lluvia “, (debe retornar 2)

3a)

```
MODULO tresVocalesDiferentes (TEXTO cadena) RETORNA ENTERO
(*Dado una cadena de texto con palabras, evalúa si la palabra tiene al menos 3 vocales diferentes. Devuelve el número de palabras diferentes*)
        ENTERO i, j, contadorVocales, cantidadPalabras ← 0
        TEXTO[] arreglo ← Dividir(cadena, “ ”) // split
        TEXTO vocales ← “aeiou”
        PARA i←0 HASTA longitud(arreglo)-1 PASO 1 HACER
                PARA j ← 0 HASTA longitud(vocales) PASO 1 HACER
                        SI indiceDe(arreglo[i], posicion(vocales, j)) > -1 ENTONCES
                                contadorVocales ← contadorVocales +1
                        FIN SI
                FIN PARA
                SI contadorVocales >= 3 ENTONCES
                        cantidadPalabras ← cantidadPalabras +1
                FIN SI
                contadorVocales ← 0
        FIN PARA
        RETORNA cantidadPalabras
FIN MODULO
```

3b)

```
MÓDULO letraIgualPegada (TEXTO cadena) RETORNA ENTERO
        ENTERO letraPegada←0
        LOGICO banderin ← falso
        ENTERO i←0, j←0
        TEXTO [] arregloPalabras ← DIVIDIR(cadena, “ ”)

        PARA i←0 HASTA LONGITUD(arregloPalabras)-1 PASO 1 HACER
                REPETIR
                        SI (j != LONGITUD(arregloPalabras[i])-1)
                                SI(POSICION(arregloPalabras[i], j) = POSICION(arrelgoPalabras[i], j+1))
                                        letraPegada ← letraPegada+1
                                        banderin ← verdadero
                                FIN SI
                        FIN SI
                        j ← j+1
                HASTA (j<LONGITUD(arregloPalabras[i]) AND banderin = falso)
                j←0
        FIN PARA
        RETORNA letraPegada
FIN MODULO
```

letraPegada	banderin	i	j	arregloPalabras	RETORNA