Comenzado el	viernes, 4 de noviembre de 2022, 21:14
Estado	Finalizado
Finalizado en	sábado, 5 de noviembre de 2022, 02:21
Tiempo empleado	5 horas 6 minutos
Puntos	2,00/2,00
Calificación	**10,00** de 10,00 (**100**%)

Pregunta **1**

Correcta

Se puntúa 1,00 sobre 1,00

Crea una clase denominada **Punto2D** que nos permita definir localizaciones en una espacio bidimensional, es decir, con dos coordenadas (x, y)

Los requerimientos de diseño de dicha clase son los siguientes:

- Definirá dos atributos **privados** denominados **cx** y **cy** para almacenar el valor numérico de las coordenadas del punto. Dichos valores serán números reales
- Proporcionará un **constructor** por defecto que inicialice las coordenadas a la posición (0.0, 0.0)
- Proporcionará un **constructor** parametrizado que permita inicializar las coordenadas a la posición indicada (x, y)
- Proporcionará un **constructor** que permitirá inicializar las coordenadas a la posición indicada por otro objeto Punto2D
- Proporcionará los métodos getter/setter **getX**, **getY**, **setX**, **setY** que permitan acceder y modificar el valor de las coordenadas de un objeto Punto2D
- Proporcionará los siguientes métodos públicos:
  - **flip()**, que intercambiará el valor de las coordenadas X e Y del objeto
  - **dist(Punto2D p)**, que devolverá la distancia al punto pasado como parámetro, calculada como: raíz cuadrada de $((x_2-x_1)^2 + (y_2-y_1)^2)$
  - **manhattanDist(Punto2D p)**, que devuelve la distancia "Manhattan" al punto pasado como parámetro. Esta distancia se calcula como la longitud del recorrido entre ambos puntos teniendo en cuenta que **sólo** nos podemos desplazar de forma horizontal y vertical
  - **slope(Punto2D p)**, que devolverá la pendiente existente entre el punto actual y el punto pasado como argumento. Dicha pendiente la calcularemos como el cociente de la división de la diferencia de las coordenadas Y entre la diferencia de las coordenadas en X
- La impresión de objetos Punto2D se ajustará al siguiente formato:
  - **Punto2D[***<valor_de_la_coordenada_X>*,*<valor_de_la_coordenada_Y>***]**

Por ejemplo:

Test	Resultado
`Punto2D p = new Punto2D();` `System.out.println(p);`	`Punto2D[0.0,0.0]`
`Punto2D p = new Punto2D(3.2, -4.8);` `System.out.println(p);`	`Punto2D[3.2,-4.8]`

`Punto2D p1 = new Punto2D(4.5, 5.5);`
`Punto2D p2 = new Punto2D(p1);`
`System.out.println(p2);`
	`Punto2D[4.5,5.5]`

`Punto2D p = new Punto2D();`
`p.setX(5.5);`
`p.setY(-2.0);`
`System.out.println(p);`
	`Punto2D[5.5,-2.0]`

`Punto2D p = new Punto2D(5.5, -2.0);` `System.out.println(p.getX() + " " + p.getY());`	`5.5 -2.0`

`Punto2D p = new Punto2D(3.2, -4.8);`
`p.flip();`
`System.out.println(p);`
	`Punto2D[-4.8,3.2]`

`Punto2D p = new Punto2D(-3.0, 4.0);`
`System.out.println(p.dist(new Punto2D()));`
`System.out.println(p.dist(new Punto2D(2.5, 1.5)));`
	`5.0`
	`6.041522986797286`

`Punto2D p1 = new Punto2D(-2.0, 3.0);`
`Punto2D p2 = new Punto2D(2.0, 1.0);`
`System.out.println(p1.manhattanDist(p2));`
`System.out.println(p2.manhattanDist(p1));`
	`6.0`
	`6.0`

Test	Resultado

`Punto2D p1 = new Punto2D(-2.0, 3.0);`
`Punto2D p2 = new Punto2D(2.0, 1.0);`
`System.out.println(p1.slope(p2));`
`p1.setY(-1);`
`System.out.println(p1.slope(p2));`
	`-0.5`
	`0.5`

Respuesta: (sistema de penalización: 0 %)

```
1  class Punto2D {
2      private double cx;
3      private double cy;
4
5      // GETTER / SETTER
6      public double getX(){
7          return this.cx;
8      }
9
10     public void setX(double cx){
11         this.cx = cx;
12     }
13
14     public double getY(){
15         return this.cy;
16     }
17
18     public void setY(double cy){
19         this.cy = cy;
20     }
21     // CONSTRUCTORS
22     Punto2D(){
```

	Test	Esperado	Se obtuvo	
✔	`Punto2D p = new Punto2D();` `System.out.println(p);`	`Punto2D[0.0,0.0]`	`Punto2D[0.0,0.0]`	✔
✔	`Punto2D p = new Punto2D(3.2, -4.8);` `System.out.println(p);`	`Punto2D[3.2,-4.8]`	`Punto2D[3.2,-4.8]`	✔

✔	`Punto2D p1 = new Punto2D(4.5, 5.5);`
	`Punto2D p2 = new Punto2D(p1);`
	`System.out.println(p2);`
		`Punto2D[4.5,5.5]`	`Punto2D[4.5,5.5]`	✔

✔	`try {`
	`  Field f = Punto2D.class.getDeclaredField("cx");`
	`  System.out.println("Punto2D.cx es privado = " + Modifier.isPrivate(f.getModifiers()));`
	`}`
	`catch(NoSuchFieldException e) {`
	`  System.out.println(e);`
	`}`
		`Punto2D.cx es privado = true`	`Punto2D.cx es privado = true`	✔

✔	`try {`
	`  Field f = Punto2D.class.getDeclaredField("cy");`
	`  System.out.println("Punto2D.cy es privado = " + Modifier.isPrivate(f.getModifiers()));`
	`}`
	`catch(NoSuchFieldException e) {`
	`  System.out.println(e);`
	`}`
		`Punto2D.cy es privado = true`	`Punto2D.cy es privado = true`	✔

✔	`Punto2D p = new Punto2D();`
	`p.setX(5.5);`
	`p.setY(-2.0);`
	`System.out.println(p);`
		`Punto2D[5.5,-2.0]`	`Punto2D[5.5,-2.0]`	✔

✔	`Punto2D p = new Punto2D(5.5, -2.0);` `System.out.println(p.getX() + " " + p.getY());`	`5.5 -2.0`	`5.5 -2.0`	✔

	Test	Esperado	Se obtuvo	

✔	`Punto2D p = new Punto2D(3.2, -4.8);`
	`p.flip();`
	`System.out.println(p);`
		`Punto2D[-4.8,3.2]`	`Punto2D[-4.8,3.2]`	✔

✔	`Punto2D p = new Punto2D(-3.0, 4.0);`
	`System.out.println(p.dist(new Punto2D()));`
	`System.out.println(p.dist(new Punto2D(2.5, 1.5)));`
		`5.0`
		`6.041522986797286`
			`5.0`
			`6.041522986797286`
				✔

✔	`Punto2D p1 = new Punto2D(-2.0, 3.0);`
	`Punto2D p2 = new Punto2D(2.0, 1.0);`
	`System.out.println(p1.manhattanDist(p2));`
	`System.out.println(p2.manhattanDist(p1));`
		`6.0`
		`6.0`
			`6.0`
			`6.0`
				✔

✔	`Punto2D p1 = new Punto2D(-2.0, 3.0);`
	`Punto2D p2 = new Punto2D(2.0, 1.0);`
	`System.out.println(p1.slope(p2));`
	`p1.setY(-1);`
	`System.out.println(p1.slope(p2));`
		`-0.5`
		`0.5`
			`-0.5`
			`0.5`
				✔

Todas las pruebas superadas. ✔

Correcta

Puntos para este envío: 1,00/1,00.

Pregunta **2**

Correcta

Se puntúa 1,00 sobre 1,00

Crea una clase denominada **TimeLapse** que nos permita almacenar instantes temporales. Los requerimientos de diseño de dicha clase son los siguientes:

- Definirá tres atributos **privados** denominados **h**, **m** y **s** donde almacenará un valor numérico de tipo entero que representan horas, minutos y segundos. Tanto los minutos como los segundos tendrán que estar en el rango: 0-59.
- Está garantizado que los casos de prueba que recibirá tu clase nunca provocaran valores mayores de Integer.MAX_VALUE.
- Proporcionará un **constructor** por defecto que inicialice los atributos a 0
- Proporcionará un **constructor** parametrizado que permita inicializar los atributos a partir de tres parámetros que representaran las horas, los minutos y los segundos (en ese orden)
- Proporcionará un **constructor** parametrizado que permita inicializar los atributos a partir de otro objeto TimeLapse
- Proporcionará un **constructor** que permitirá inicializar los atributos a partir de un único valor numérico expresado en segundos. Ten en cuenta que tanto minutos como segundos deben estar en el rango 0-59. Por tanto, este método podría dar lugar a la modificación de los tres atributos (no sólo de los segundos)
- Proporcionará los métodos getter/setter **getH**, **getM**, **getS**, **setH**, **setM**, **setS** que permitan acceder y modificar el valor de los atributos del objeto.
- Proporcionará los siguientes métodos públicos:
  - **totalSec()**, devuelve el número total de segundos
  - **reset()**, inicializa a 0 los tres atributos
  - **addSec(int sec)**, añade el número indicado de segundos (recuerda la limitación de rangos)
  - **addTime(TimeLapse t)**, añade al tiempo atual el tiempo pasado como parámetro (recuerda la limitación de rangos)
- La impresión de objetos TimeLapse se ajustará al siguiente formato:

- **TimeLapse[***<Horas>***h:***<Minutos>***m:***<Segundos>***s]**

(Para resolver este problema sólo puedes emplear estructuras del lenguaje vistas hasta ahora. Es decir, no puedes usar bucles, condicionales,..)

A continuación se muestran una serie de casos de uso de la clase junto con los resultados esperados de la ejecución de dichos tests:

Por ejemplo:

Test	Resultado
`TimeLapse t = new TimeLapse();` `System.out.println(t);`	`TimeLapse[0h:0m:0s]`
`TimeLapse t = new TimeLapse(3, 25, 42);` `System.out.println(t);`	`TimeLapse[3h:25m:42s]`

`TimeLapse t1 = new TimeLapse(3, 25, 42);`
`TimeLapse t2 = new TimeLapse(t1);`
`System.out.println(t2);`
	`TimeLapse[3h:25m:42s]`

`TimeLapse t = new TimeLapse(12930);` `System.out.println(t);`	`TimeLapse[3h:35m:30s]`

`TimeLapse t = new TimeLapse();`
`t.setH(5);`
`t.setM(20);`
`t.setS(50);`
`System.out.println(t);`
	`TimeLapse[5h:20m:50s]`

`TimeLapse t = new TimeLapse(5, 20, 50);` `System.out.println(t.getH() + " " + t.getM() + " " + t.getS());`	`5 20 50`
`TimeLapse t = new TimeLapse(5, 20, 50);` `System.out.println(t.totalSec());`	`19250`

Test	Resultado

`TimeLapse t = new TimeLapse(5, 20, 50);`
`t.reset();`
`System.out.println(t);`
	`TimeLapse[0h:0m:0s]`

`TimeLapse t = new TimeLapse(5, 50, 50);`
`t.addSec(10000);`
`System.out.println(t);`
	`TimeLapse[8h:37m:30s]`

`TimeLapse t = new TimeLapse(5, 50, 50);`
`t.addTime(new TimeLapse(2, 46, 40));`
`System.out.println(t);`
	`TimeLapse[8h:37m:30s]`

Respuesta: (sistema de penalización: 0 %)

```
1  /*
2      El control de los segundos y minutos no está bién. Si añades mas de 60, no se hace nada
3   */
4  class TimeLapse {
5
6      private int h;
7      private int m;
8      private int s;
9
10     // Solo como variables de la clase. No se ven de forma externa
11     private int segMin = 60;
12     private int segHora = 60 * segMin;
13
14     // GETTER SETTER
15     public int getH(){
16         return this.h;
17     }
18     public void setH(int h){
19             this.h = h;
20     }
21
22     public int getM(){
```

	Test	Esperado	Se obtuvo	
✔	`TimeLapse t = new TimeLapse();` `System.out.println(t);`	`TimeLapse[0h:0m:0s]`	`TimeLapse[0h:0m:0s]`	✔
✔	`TimeLapse t = new TimeLapse(3, 25, 42);` `System.out.println(t);`	`TimeLapse[3h:25m:42s]`	`TimeLapse[3h:25m:42s]`	✔

✔	`TimeLapse t1 = new TimeLapse(3, 25, 42);`
	`TimeLapse t2 = new TimeLapse(t1);`
	`System.out.println(t2);`
		`TimeLapse[3h:25m:42s]`	`TimeLapse[3h:25m:42s]`	✔

✔	`TimeLapse t = new TimeLapse(12930);` `System.out.println(t);`	`TimeLapse[3h:35m:30s]`	`TimeLapse[3h:35m:30s]`	✔

✔	`try {`
	`  Field f =`
	`TimeLapse.class.getDeclaredField("h");`
	`  System.out.println("TimeLapse.h es privado = "`
	`+ Modifier.isPrivate(f.getModifiers()));`
	`}`
	`catch(NoSuchFieldException e) {`
	`  System.out.println(e);`
	`}`
		`TimeLapse.h es privado = true`	`TimeLapse.h es privado = true`	✔

	Test	Esperado	Se obtuvo	

✔	`try {`
	`  Field f =`
	`TimeLapse.class.getDeclaredField("m");`
	`  System.out.println("TimeLapse.m es privado = "`
	`+ Modifier.isPrivate(f.getModifiers()));`
	`}`
	`catch(NoSuchFieldException e) {`
	`  System.out.println(e);`
	`}`
		`TimeLapse.m es privado = true`	`TimeLapse.m es privado = true`	✔

✔	`try {`
	`  Field f =`
	`TimeLapse.class.getDeclaredField("s");`
	`  System.out.println("TimeLapse.s es privado = "`
	`+ Modifier.isPrivate(f.getModifiers()));`
	`}`
	`catch(NoSuchFieldException e) {`
	`  System.out.println(e);`
	`}`
		`TimeLapse.s es privado = true`	`TimeLapse.s es privado = true`	✔

✔	`TimeLapse t = new TimeLapse();`
	`t.setH(5);`
	`t.setM(20);`
	`t.setS(50);`
	`System.out.println(t);`
		`TimeLapse[5h:20m:50s]`	`TimeLapse[5h:20m:50s]`	✔

✔	`TimeLapse t = new TimeLapse(5, 20, 50);`
	`System.out.println(t.getH() + " " + t.getM() + " "`
	`+ t.getS());`
		`5 20 50`	`5 20 50`	✔

✔	`TimeLapse t = new TimeLapse(5, 20, 50);` `System.out.println(t.totalSec());`	`19250`	`19250`	✔

	Test	Esperado	Se obtuvo	

✔	`TimeLapse t = new TimeLapse(5, 20, 50);`
	`t.reset();`
	`System.out.println(t);`
		`TimeLapse[0h:0m:0s]`	`TimeLapse[0h:0m:0s]`	✔

✔	`TimeLapse t = new TimeLapse(5, 50, 50);`
	`t.addSec(10000);`
	`System.out.println(t);`
		`TimeLapse[8h:37m:30s]`	`TimeLapse[8h:37m:30s]`	✔

✔	`TimeLapse t = new TimeLapse(5, 50, 50);`
	`t.addTime(new TimeLapse(2, 46, 40));`
	`System.out.println(t);`
		`TimeLapse[8h:37m:30s]`	`TimeLapse[8h:37m:30s]`	✔

Todas las pruebas superadas. ✔

Correcta

Puntos para este envío: 1,00/1,00.

◄ Preguntas CodeRunner para probar clases java

Ir a...

Foro para Estructuras de control en detalle ►