Comenzado el	miércoles, 22 de marzo de 2023, 16:45
Estado	Finalizado
Finalizado en	miércoles, 22 de marzo de 2023, 18:53
Tiempo empleado	2 horas 7 minutos
Puntos	4,00/5,00
Calificación	**8,00** de 10,00 (**80**%)

Pregunta **1**

Correcta

Se puntúa 1,00 sobre 1,00

Siete picos en Kotlin

En 1969 se inauguró el Parque de Atracciones de Madrid; su atracción estrella era la montaña rusa *"Siete picos"*, que, tras 36 años de servicio y unos 77 millones de usuarios, fue desmontada en 2005 para, como ella mismo "dijo" en su carta de despedida, dejar paso a las nuevas generaciones.

Curiosamente, pese a su nombre, aquella montaña rusa no tenía siete picos. Si llamamos "pico" a un punto del recorrido que está más alto que el inmediatamente anterior y el inmediatamente siguiente, entonces tenía como mucho 6 y ni siquiera las crónicas se ponen de acuerdo en esto.

Dado el recorrido de varias montañas rusas, ¿puedes contar el número de picos? Ten en cuenta que las montañas rusas son circulares, y el punto de inicio de la entrada ¡podría ser un pico!

Entrada

El programa leerá de la entrada estándar múltiples casos de prueba, cada uno con la descripción de una montaña rusa.

Una montaña rusa queda descrita por un primer número $2 \leq n \leq 1.000$ indicando cuántas veces se ha anotado la altura del recorrido. A continuación vienen, en otra línea, n números positivos (menores que 1.000) con todas esas alturas.

La entrada termina con una montaña rusa sin alturas que no deberá procesarse.

Salida

Para cada caso de prueba el programa escribirá el número de picos de la montaña rusa que representa. Recuerda que las montañas rusas son circuitos cerrados, y tras el final vuelven a comenzar.

Por ejemplo:

Entrada	Resultado
4 4 10 3 2 4 10 3 2 4 5 4 10 10 3 2 0	1 1 0

Respuesta: (sistema de penalización: 0 %)

El editor Ace no está listo. Tal vez funcione si vuelve a cargar la página.
Falling back to raw text area.

```
fun main() {

    // Capturamos cant picos primera vez
    var numPicos = readln().toInt()
    while (numPicos > 0) {
        // variable tamaños picos
        val tamPicos = IntArray(numPicos)

        // Capturamos tamaños picos
        val input = readln().split(" ")
        for (i in input.indices){
            tamPicos[i] = input[i].toInt()
        }



        // contador picos
```

	Entrada	Esperado	Se obtuvo	
✔	4 4 10 3 2 4 10 3 2 4 5 4 10 10 3 2 0	1 1 0	1 1 0	✔
✔	6 1 4 2 2 3 5 0	2	2	✔

Todas las pruebas superadas. ✔

Correcta

Puntos para este envío: 1,00/1,00.

Pregunta **2**

Correcta

Se puntúa 1,00 sobre 1,00

Mejorar el cumplimiento del principio abierto - cerrado

Observa el siguiente código

```
abstract class Figura {
    abstract double area();
}

class Rectangulo extends Figura {
    double largo;
    double ancho;

    public Rectangulo(double largo, double ancho) {
        this.largo = largo;
        this.ancho = ancho;
    }

    @Override
    double area() {
        return largo * ancho;
    }

}

class Circulo extends Figura {
    double radio;

    public Circulo(double radio) {
        this.radio = radio;
    }

    @Override
    double area() {
        return radio * radio * Math.PI;
    }
}

public class App {
    public static void main(String[] args) {
        Figura[] figuras = { new Circulo(2), new Rectangulo(5, 2) };
        for (Figura f : figuras) {
            if (f.getClass().getName().equals("Circulo")) {
                System.out.println("Esta figura es un Círculo y sus área es: " + f.area());
            }
            if (f.getClass().getName().equals("Rectangulo")) {
                System.out.println("Esta figura es un Rectángulo y sus área es: " + f.area());
            }
        }
    }
}
```

Cada vez que queremos incorporar una nueva figura en nuestra App tenemos que añadir un nuevo if. Probablemente haremos copiar/pegar de un if existente y a continuación lo retocamos para la nueva figura.
El diseño tiene alguna carencia ya que si extendemos los tipos de figura hay que modificar la App. Se pide que hagas los cambios oportunos en la jerarquía de forma que funcionen los siguientes tests con código de App más extensible a nuevos tipos.

Recuerda que el área de un triángulo es base*altura/2

Envía a code runner: La jerarquía (completa, con todas sus clases)retocada para que funcionen los tests. No se envía App, sólo clases de jerarquía. Recuerda que ninguna clase/interface puede ser public en este formato de entrega.

Por ejemplo:

Test	Resultado

Test	Resultado
`Figura[] figuras = { new Circulo(2.0), new Rectangulo(5.0, 2.0), new Triangulo(2.0, 2.0) }; for (Figura f : figuras) { System.out.println(f); }`	`Este es un Círculo maravilloso y su área es 12.566370614359172 Esta figura es un Rectángulo y su área es 10.0 Este bonito triángulo tiene un área de 2.0`

Respuesta: (sistema de penalización: 0 %)

El editor Ace no está listo. Tal vez funcione si vuelve a cargar la página.
Falling back to raw text area.

```
abstract class Figura {
    abstract double area();
    public abstract String toString();
}

/*
 * Este es un Círculo maravilloso  y su área es 12.566370614359172
Esta figura es un Rectángulo y  su área es 10.0
Este bonito triángulo tiene un área de 2.0

 */
class Rectangulo extends Figura {
    double largo;
    double ancho;

    public Rectangulo(double largo, double ancho) {
        this.largo = largo;
        this.ancho = ancho;
```

	Test	Esperado	Se obtuvo	
✔	`Figura[] figuras = { new Circulo(2.0), new Rectangulo(5.0, 2.0), new Triangulo(2.0, 2.0) }; for (Figura f : figuras) { System.out.println(f); }`	`Este es un Círculo maravilloso y su área es 12.566370614359172 Esta figura es un Rectángulo y su área es 10.0 Este bonito triángulo tiene un área de 2.0`	`Este es un Círculo maravilloso y su área es 12.566370614359172 Esta figura es un Rectángulo y su área es 10.0 Este bonito triángulo tiene un área de 2.0`	✔

Todas las pruebas superadas. ✔

Correcta

Puntos para este envío: 1,00/1,00.

Pregunta **3**

Correcta

Se puntúa 1,00 sobre 1,00

Guardar poemas en ficheros

Tengo objetos Poema que encapsulan la información básica de mis poemas: el título y el texto del poema. Por otro lado, a menudo deseo grabar mis poemas en formato pdf para lo que escribo una clase PDFSave. En el ejemplo, la grabación se simula con un println(). Usando estas clases escribo una App que me permite grabar mis Poemas.

```
class PDFSave {
    public void guardar(String poema) {
       System.out.println("poema salvado en formato PDF");
    }
}


class Poema{
    String titulo;
    String texto;
    PDFSave grabador;
    Poema(String titulo, String texto){
        this.titulo=titulo;
        this.texto=texto;
        grabador= new PDFSave();
    }
    void guardar(){
        grabador.guardar(titulo+texto);
    }

}
public class App{
    public static void main(String[] args) {
       Poema p1= new Poema("El viento", "¿Qué es el viento ...?");
       p1.guardar();
    }
}
```

Ahora, resulta que queremos grabar los poemas en diferentes formatos como doc y txt de tal forma que la App debería poder indicar en que formato se quiere grabar el poema. El diseño anterior se nos queda corto ya que la clase Poema depende del formato pdf, es decir, para grabar depende de una clase concreta cuando debería hacerlo de algo más abstracto. Mejora el diseño anterior de forma que funcionen los siguientes tests, pero además **para puntuar es obligaorio cumplir el principio de inversion de dependencias.**

SE ENTREGA: nuevo diseño pero sin la clase App del ejemplo

Por ejemplo:

Test	**Resultado**

`Poema p1 = new Poema("El viento. ", "¿Qué es el viento ...?");`
`p1.guardar(new DOCSave());`
`p1.guardar(new TXTSave());`
`p1.guardar(new PDFSave());`
	`poema salvado en formato DOC: El viento. ¿Qué es el viento ...?`
	`poema salvado en formato txt: El viento. ¿Qué es el viento ...?`
	`poema salvado en formato PDF: El viento. ¿Qué es el viento ...?`

Respuesta: (sistema de penalización: 0 %)

El editor Ace no está listo. Tal vez funcione si vuelve a cargar la página.
Falling back to raw text area.

```
interface Guardar {
    public void guardar(String titulo, String texto);
}
// poema salvado en formato DOC: El viento. ¿Qué es el viento ...?
class DOCSave implements Guardar {

    @Override
    public void guardar(String titulo, String texto) {
        System.out.println("poema salvado en formato DOC: " + titulo+ texto );
    }

}

class TXTSave implements Guardar {
    @Override
    public void guardar(String titulo, String texto) {
        System.out.println("poema salvado en formato txt: " + titulo + texto );
    }
```

	Test	Esperado	Se obtuvo	

✔	`Poema p1 = new Poema("El viento. ", "¿Qué es el viento ...?");`
	`p1.guardar(new DOCSave());`
	`p1.guardar(new TXTSave());`
	`p1.guardar(new PDFSave());`
		`poema salvado en formato DOC: El viento. ¿Qué es el viento ...?`
		`poema salvado en formato txt: El viento. ¿Qué es el viento ...?`
		`poema salvado en formato PDF: El viento. ¿Qué es el viento ...?`
			`poema salvado en formato DOC: El viento. ¿Qué es el viento ...?`
			`poema salvado en formato txt: El viento. ¿Qué es el viento ...?`
			`poema salvado en formato PDF: El viento. ¿Qué es el viento ...?`
				✔

Todas las pruebas superadas. ✔

Correcta

Puntos para este envío: 1,00/1,00.

Pregunta **4**

Correcta

Se puntúa 1,00 sobre 1,00

Singleton para guardar configuración de App

A menudo, en aplicaciones grandes se requiere tener una clase para guardar la configuración de la aplicación: título de la App, autores, contacto, licencias, ... Esta configuración, para que sea congruente, se debe guardar de forma centralizada en un único objeto , y por lo tanto, la mejor forma de conseguir este efecto es que la clase se escriba acorde al patrón Singleton
Para simplificar, usamos como configuración de nuestra aplicación simplemente el título. El nombre por defecto de la aplicación como se aprecia en el test es "Mi Aplicación"

```
class AppConfig {
  private String appTitle;
   .... resto de clase ....
}
```

Se pide escribir la clase AppConfig siguiendo el patrón Singleton y de forma que pase el test ejemplo. Ten en cuenta que pasar el test no es suficiente para puntuar, se revisará que cumple el patrón Singleton.

ENTREGA: la clase AppConfig
Por ejemplo:

Test	Resultado

`AppConfig appcfg= AppConfig.getInstance();`
`System.out.println(appcfg.getAppTitle());`
`appcfg.setAppTitle("Tú aplicación");`
`System.out.println(appcfg.getAppTitle());`
`appcfg= AppConfig.getInstance();`
`System.out.println(appcfg.getAppTitle());`
	`Mi Aplicación`
	`Tú aplicación`
	`Tú aplicación`

Respuesta: (sistema de penalización: 0 %)

El editor Ace no está listo. Tal vez funcione si vuelve a cargar la página.
Falling back to raw text area.

```
class AppConfig {
    private static AppConfig instanciaUnica;
    private String title;


    public void setAppTitle(String title) {
        this.title = title;
    }

    public String getAppTitle() {
        return title;
    }

    private AppConfig() {
        this.title = "Mi Aplicación";
    }

    public static synchronized AppConfig getInstance() {
```

	Test	Esperado	Se obtuvo	

✔	`AppConfig appcfg= AppConfig.getInstance();`
	`System.out.println(appcfg.getAppTitle());`
	`appcfg.setAppTitle("Tú aplicación");`
	`System.out.println(appcfg.getAppTitle());`
	`appcfg= AppConfig.getInstance();`
	`System.out.println(appcfg.getAppTitle());`
		`Mi Aplicación`
		`Tú aplicación`
		`Tú aplicación`
			`Mi Aplicación`
			`Tú aplicación`
			`Tú aplicación`
				✔

Todas las pruebas superadas. ✔

Correcta

Puntos para este envío: 1,00/1,00.

Pregunta **5**

Incorrecta

Se puntúa 0,00 sobre 1,00

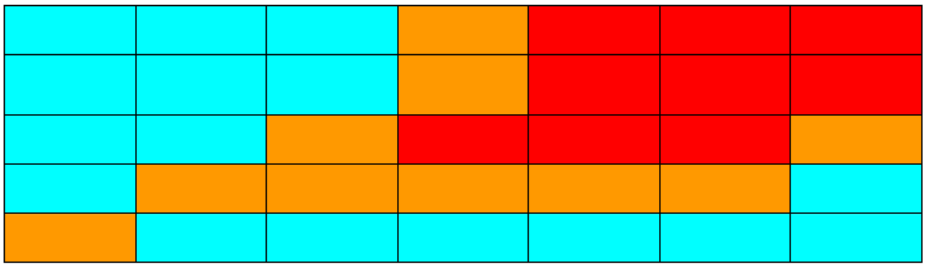
¡Mire lo que me hizo su hijo!

Mi vecino tiene una piscina increible ya que a través de un complicado sistema de tabiques moviles puede hacer dentro de la piscina subpiscinas sin comunicación de aguas entre sí. Un día la divide en dos para separar niños de mayores, otro día hace pocitas individuales etc. Como buen tiquismiquis le echa al agua de la piscina el famoso producto que detecta si alguien vierte de alguna forma líquido sucio a la piscina en cuyo caso tiñe de rojo el agua para avisar de que algo indeseable ocurre en la zona de baño. Mi vecino echa tal concetración de producto, que de detectarse suciedad sospechosa se tiñe de rojo todo el agua de la subpiscina afectada. Como las subpiscinas están aisladas por los tabiques entre sí la contaminación de una piscina no afecta a las otras. Cuando mi hijo se entero de esto, el muy gamberrete se compró un tirachinas y lanzaba bolas de papel impregnadas en *líquido sucio* hacia la piscina. Si conseguía que una bola cayera en una de las subpiscinas esta se teñia de rojo. Al poco tiempo de que mi hijo daba en el blanco, tenía al vecino en la puerta de mi casa enseñándome una foto a la pisicina diciéndomoe ¡Mire lo que me hizo su hijo!.

Queremos realizar un programa en java que simule todo este proceso para poder explicarselo al psicologo de mi hijo. La piscina la representamos como una matriz de caracteres. Cada celda de la matriz de entrada puede valer el caracter 'A' para indicar que tiene agua limpia o 'T' para indicar que es un tabique. Recuerda que con los tabiques generamos subpiscinas aisladas entre sí. Si a nuestro programas le indicamos la coordenada de la celda en la que cae una bola de papel contaminante, nos devolverá "la foto" en la que se aprecia la superfice de la piscina contaminada. Sí la bola cae fuera de la piscina o encima de un tabique no habrá ninguna subpiscina contaminada.

ENTRADA: la primera línea contiene dos enteros separados por un espacio. El primero indica la cantidad de filas y el segundo la cantidad de columnas. A continuación viene f filas que describen la matriz. La última fila contiene dos enteros separados por un espacio que indican las coordenadas de la celda donde cayó la bola contaminante.

SALIDA: Matriz con las celdas contaminadas descritas con el caracter R. Por ejemplo, el siguiente gráfico se corresponde con el último caso de prueba



naranja: tabique
azul: agua limpia
rojo: agua contaminada

SE ENTREGA: Programa java que resuelve el problema
Por ejemplo:

Entrada	Resultado
3 3 ATA TAT ATA 0 1	ATA TAT ATA
3 3 ATA TAT ATA 2 2	ATA TAT ATR

Entrada	Resultado
5 5 AAAAA AAAAA TTTTT AAAAA AAAAA 1 2	RRRRR RRRRR TTTTT AAAAA AAAAA
5 7 AAATAAA AAATAAA AATAAAT ATTTTTA TAAAAAA 1 6	AAATRRR AAATRRR AATRRRT ATTTTTA TAAAAAA

Respuesta: (sistema de penalización: 0 %)

El editor Ace no está listo. Tal vez funcione si vuelve a cargar la página.
Falling back to raw text area.

```
import java.util.Scanner;

public class App {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        // La piscina la representamos como una matriz de caracteres
        char[][] piscina;
        // la primera línea contiene dos enteros separados por un espacio. El primero
        // indica la cantidad de filas y el segundo la cantidad de columnas
        String[] input_1 = sc.nextLine().split(" ");
        int totalFilas = Integer.parseInt(input_1[0]);
        int totalColumnas = Integer.parseInt(input_1[1]);

        // Cada celda de la matriz de entrada puede valer el caracter 'A' para indicar
        // que tiene agua limpia o 'T' para indicar que es un tabique
        char celdaLimpia = 'A';
        char celdaTabique = 'T';
```

	Entrada	Esperado	Se obtuvo	
✖	3 3 ATA TAT ATA 0 1	ATA TAT ATA	A A A A A A A A A	✖
✖	3 3 ATA TAT ATA 2 2	ATA TAT ATR	A A A A A A A A A	✖

	Entrada	Esperado	Se obtuvo	
✖	5 5 AAAAA AAAAA TTTTT AAAAA AAAAA 1 2	RRRRR RRRRR TTTTT AAAAA AAAAA	A A A A A A A A A A A A A A A A A A A A A A A A A	✖
✖	5 7 AAATAAA AAATAAA AATAAAT ATTTTTA TAAAAAA 1 6	AAATRRR AAATRRR AATRRRT ATTTTTA TAAAAAA	A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A	✖

Your code must pass all tests to earn any marks. Try again.

Mostrar diferencias

Incorrecta

Puntos para este envío: 0,00/1,00.

◄ unas soluciones

Ir a...

ExamenEv2 2023-sol ►