SIETE PICOS EN KOTLIN

```
fun main(){
    var casos=readln()
    while(casos!="0"){
        var listaMedidas= mutableListOf<Int>()
        //la línea de medidas se almacena en lista de enteros
        for(medida in readln().split(" ")){
            listaMedidas.add(medida.toInt())
        }
        //procesamos la lista
        var picos=0
        for(i in listaMedidas.indices){
            val ant= if(i==0) listaMedidas.lastIndex else i-1
            val sgte=if(i==listaMedidas.lastIndex) 0 else i+1
            if(listaMedidas[ant]<listaMedidas[i]&& listaMedidas[i]>listaMedidas[sgte])
                picos++

        }
        println(picos)
        casos=readln()
    }


}
```

Mejorar el cumplimiento del principio abierto - cerrado

Se trata de sobreescribir en cada clase el método toString(). Aunque resolver el ejercicio consiste simplemente en esta pequeña acción, lo realmente importante es observar cómo este pequeño cambio basado en polimorfismo tiene impacto en el grado de cumplimiento del principio de abierto cerrado

```
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;
    }

    @Override
    public String toString() {
        return "Esta figura es un Rectángulo y  su área es " + this.area();
    }
}

class Circulo extends Figura {
    double radio;

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

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

    @Override
    public String toString() {
        return "Este es un Círculo maravilloso  y su área es " + this.area();
    }
}
class Triangulo extends Figura{
    double base;
    double altura;
    public Triangulo(double base, double altura) {
        this.base = base;
        this.altura = altura;
    }

    @Override
    double area() {
        return base*altura/2;
    }
    @Override
    public String toString() {
        return "Este bonito triángulo tiene un área de " + this.area();
    }
}
```

Guardar poemas en ficheros

A menudo un diseño pobre no cumple o cumple sólo parcialmente los principios SOLID. En este caso los principios más vulnerados son el de abierto-cerrado y sobre todo de forma muy directa incumple el principio de inversión de dependencias: la clase Poema debería depender de una clase grabador abstracta, además a partir de aquí también se mejoraría el cumplimiento del principio abierto - cerrado.

```
interface Grabador {
   void guardar(String texto);
}

class DOCSave implements Grabador {
   @Override
   public void guardar(String poema) {
      System.out.print("poema salvado en formato DOC: ");
      System.out.println(poema);
   }
 }

class PDFSave implements Grabador {
   @Override
   public void guardar(String poema) {
      System.out.print("poema salvado en formato PDF: ");
      System.out.println(poema);
   }
}

class TXTSave implements Grabador {
   @Override
   public void guardar(String poema) {
      System.out.print("poema salvado en formato txt: ");
      System.out.println(poema);
   }
}

class Poema {
   String titulo;
   String texto;
   Grabador grabador;

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

   void guardar(Grabador grabador) {
      this.grabador=grabador;
      this.grabador.guardar(titulo + texto);
   }

}
```

Singleton para guardar configuración de App

```
class AppConfig {
  private static AppConfig instance = null;
  private String appTitle;

  private AppConfig() {
      appTitle = "Mi Aplicación";
  }

  public  synchronized static AppConfig getInstance() {
      if (instance == null) {
          instance = new AppConfig();
      }
      return instance;
  }

  public String getAppTitle() {
      return appTitle;
  }
  public void setAppTitle(String appTitle) {
    this.appTitle = appTitle;
  }

}
```

¡Miré que me hizo su hijo!

```
import java.util.Scanner;

public class App{
    static Scanner sc = new Scanner(System.in);
    static char[][] leerMatriz(){
        int fil=sc.nextInt();
        int col=sc.nextInt();
        char[][] m= new char[fil][col];
        for (int i = 0; i < m.length; i++) {
            m[i]=sc.next().toCharArray();
        }
        return m;
    }
    static void imprimirMatriz(char[][] m){
        for (int i = 0; i < m.length; i++) {
            for (int j = 0; j < m[i].length; j++) {
                System.out.print(m[i][j]);
            }
            System.out.println();
        }
    }
    static void contaminar(char[][] m, int fil, int col){
        if(fil<0 || fil >=m.length || col<0 || col>=m[fil].length) return;
        if(m[fil][col]=='T') return;
        //es A o R, si es R no se hace nada
        if(m[fil][col]=='A'){
            m[fil][col]='R';
            contaminar(m, fil-1,col);
            contaminar(m, fil,col-1);
            contaminar(m, fil,col+1);
            contaminar(m, fil+1,col);
        }

    }
    public static void main(String[] args) {

        char[][] m=leerMatriz();

        //coordenadas ataque tirachinas
        int fil=sc.nextInt();
        int col=sc.nextInt();
        contaminar(m,fil,col);

        imprimirMatriz(m);

    }

}
```