Entornos de Desarrollo

Dada la clase **Prueba**, realizar refactorizaciones con el objetivo final de modificar (optimizar) dicha clase convirtiéndola en dos clases mejoradas con los nombres **CambiaLongitud** y **NoCambiaLongitud** y la estructura siguiente:

- Clase1 -> **CambiaLongitud** (incluye métodos que cambian longitud array)
  Variables públicas -> Valor
  Métodos públicos -> Inserción, borrado y visualización
- Clase2 -> **NoCambiaLongitud** (incluye métodos que NO cambian longitud array)
  Variables públicas -> Valor
  Métodos públicos -> Modificación y visualización.

Se ha incluído un caso de prueba diseñado con JUnit para comprobar el correcto funcionamiento de la aplicación después de cada refactorización que realicemos (PruebaTest). Para ejecutarlo simplemente se ejecutará.
Por tanto, cada vez que se realize una optimización no olvides ejecutar la prueba para verificar que la/s clase/s hacen lo que se pide. RECUERDA QUE ESTE TIPO DE CAMBIOS NO DEBE AFECTAR A LA FUNCIONALIDAD DEL PROGRAMA. Para ello aplica las siguientes opciones de refactorización:

1º.- Cambiar nombre:

Cambia el nombre de la variable pública **va** por **valor**
Cambia el nombre de la variable **p** por **posicion**
Cambia el nombre del array **v** por **vector**
Cambia el nombre del método **mo** por **modificar**
Cambia el nombre del método **bo** por **borrar**
Cambia el nombre del método **in** por **insertar**

2º.- Inline:

Los métodos **BorraElemento**, **InsertaElemento** y **ModificaElemento** solo se llaman una vez y su cuerpo es bastante explícito por lo que no son necesarios. Reemplazad la llamada al método por su cuerpo

3º.- Cambiar parámetros del método

De cara a una futura mejora del programa añadir el parámetro **longitudActual** de tipo entero con un valor inicial de 0 a la lista de parámetros de los métodos **modificar**, **borrar** e **insertar**.

4º.- Introducir método

Como véis, hay un trozo de código que se corresponde con la visualización del vector (array).

```
System.out.print("v={");
    for (i=0;i<v.length;i++){
      System.out.print(v[i]+",");
      }
    System.out.println("}");
```

Podríamos crear un método llamado **visualizar** de tipo PÚBLICO. Recuerda cambiar todas las ocurrencias. Por otro lado, netbeans te crea un método llamado **visualizar** con dos parámetro (i y vector). El parámetro **i** no es necesario por lo que lo podéis eliminar (opción **cambiar parámetros** del método de refactorización y eliminarlo).
Nota: Al hacer el cambio no olvidéis comprobar que el programa hace lo que tiene que hacer.

5º.- Introducir parámetro

Podemos hacer que la variable local "**posicion**" de los métodos se pase como parámetro a través de los métodos. Basta con seleccionarla y elegir esta opción de la refactorización.

6º.- Encapsular campos

Encapsulamos en los métodos correspondientes la variable pública de la clase Prueba (la variable **valor**). Con esto conseguimos el encapsulamiento de las variables públicas de forma que éstas puedan ser modificados o accedidas a través de los métodos correspondientes.

7º.- Introducir variable (también podemos usar como alternativa introducir parámetro)

Si os fijáis en el método **visualizar** se usa la función *vector.length*. Podríamos mejorar la legibilidad del programa inicializando una variable llamada **longitud** al valor vector.length.

8º.- Copiar

Como nos hace falta una segunda clase que aproveche parte de la de Prueba, lo que hacemos es copiar la clase Prueba en una nueva clase dandole de nombre ***NoCambiaLongitud***.

9º.- Cambiar nombre

Cambia el nombre de la clase **Prueba** a **CambiaLongitud**

10º.- Eliminar métodos

Elimina de la clase **NoCambiaLongitud** los métodos **modificar**, **insertar** y **borrar**. Como véis no deberíamos borrar **modificar** pero lo hacemos simulando un error.

11º.- Mover

Movemos el método que habíamos borrado de forma fortuita (**modificar**) de la clase **CambiaLongitud** a la clase **NoCambiaLongitud**.

Hay que fijarse que la llamada a los métodos visualizar y getValor de la clase CambiaLongitud. Quitar el parámetro CambiaLongitud y modificar la llamada a los métodos.

12º.- Extraer superclase

Si os fijáis, las dos clases poseen un método público común (**visualizar**) y una variable pública común (**valor**). Se podría crear una clase padre de nombre **MiArray** con el método visualizar y la variable valor de forma que las clases **NoCambiaLongitud y CambiaLongitud** herede el método **visualizar** y la variable **valor**.