|  |
| --- |
| **Herramienta ECATP\_50v3** |
| Post-Proceso de archivos |
|  |
| En este documento se detalla información importante del funcionamiento del programa, así como explicación breve del código utilizado en dicha herramienta, y los pasos a seguir en caso de realizar una actualización. |
|  |
| **Daniel Rodriguez** |
| **18/10/2010** |
|  |



# Índice.

Contenido

[Índice. 2](#_Toc275194910)

[Introducción. 4](#_Toc275194911)

[Herramienta JCreator. 5](#_Toc275194912)

[Modificación de software ECATP\_50. 9](#_Toc275194913)

[Descripción de las clases existentes en el programa. 12](#_Toc275194914)

[Clase ECATP\_50. 13](#_Toc275194915)

[1. Método principal (main()). 13](#_Toc275194916)

[2. Método blockedfilelesen (). 18](#_Toc275194917)

[3. Método randomselection (). 23](#_Toc275194918)

[4. Método hypothesislesen (). 25](#_Toc275194919)

[Clase rawsamplenode. 27](#_Toc275194920)

[1. Constructores Rawsamplenode. 27](#_Toc275194921)

[2. Métodos Gets (). 30](#_Toc275194922)

[3. Métodos Sets (). 32](#_Toc275194923)

[Clase verketteteliste. 37](#_Toc275194924)

[1. Método insertelement (). 37](#_Toc275194925)

[2. Método deletefirst (). 37](#_Toc275194926)

[3. Método insertend (). 38](#_Toc275194927)

[4. Método nodedistancecheck (). 41](#_Toc275194928)

[5. Método summarizenodes (). 43](#_Toc275194929)

[6. Método deletenode (). 46](#_Toc275194930)

[7. Método ausgabedateiAVG (). 48](#_Toc275194931)

[8. Método CalculateAVG (). 50](#_Toc275194932)

[9. Método maxaverage (). 51](#_Toc275194933)

[10. Método numberofsegments (). 52](#_Toc275194934)

[11. Método deleterandom (). 54](#_Toc275194935)

[12. Método ausgabedateiRNDM (). 55](#_Toc275194936)

[13. Método passedsegments (). 57](#_Toc275194937)

[14. Método failedsegments (). 59](#_Toc275194938)

[15. Método insertendadapted (). 61](#_Toc275194939)

[Clase hptest. 62](#_Toc275194940)

[1. Constructor hptest (). 63](#_Toc275194941)

[2. Métodos Gets (). 63](#_Toc275194942)

[3. Métodos Sets (). 65](#_Toc275194943)

[Clase conf. 66](#_Toc275194944)

[1. Constructor conf (). 66](#_Toc275194945)

[2. Método insertelement (). 66](#_Toc275194946)

[3. Método insertend (). 67](#_Toc275194947)

[4. Método output (). 69](#_Toc275194948)

[5. Método errormarginadaption (). 71](#_Toc275194949)

[6. Método margintouse (). 73](#_Toc275194950)

[7. Método deletefirst (). 74](#_Toc275194951)

[Tipo de archivos que soporta la Herramienta. 75](#_Toc275194952)

Herramienta ECATP

# Introducción.

Este documento es creado para la versión 2.0 creada para el post-proceso de archivos .txt que contengan información de 50 frecuencias o menor a ellos.

**Limitantes de la herramienta:**

* Solo puede realizar post-procesos de archivos que contengan mediciones de 50 frecuencias.
* Dicha herramienta solo puede procesar archivos en formato .txt
* En caso de archivos corruptos no se podrá realizar el proceso del archivo (Por ejemplo: que contengan símbolos en columnas de datos).

**Descripción:**

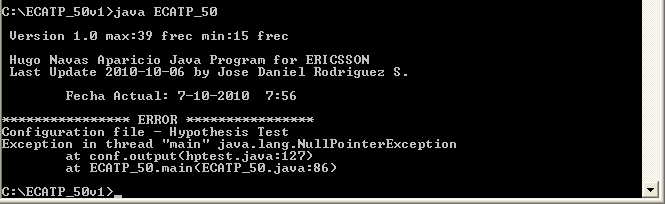
Existen 5 archivos .java en donde se contienen todas las clases de la programada. Los siguientes son (figura 1):

1. ECATP\_50.java (este es el archivo que contiene la clase principal).
2. ECATP\_adapt.java (este es otro archivo que contiene la clase principal).
3. Hptest.java
4. Rawsample.java
5. Keyin.java



**Figura 1.** Archivos .java de la herramienta ECATP\_50.

Esta herramienta también necesita del archivo de configuración inicial HPTC.txt. Sin este archivo a la hora de correr el programa sale el siguiente error:



**Figura 2.** Pantallazo del error en caso de que HPTC.txt no se encuentre.

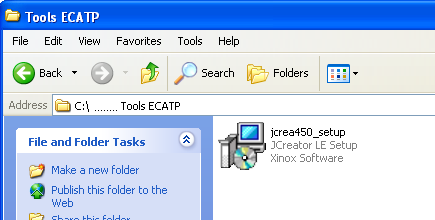
Esto indica que existe un valor nulo en un puntero, en la clase conf(), en el método output(). Por lo tanto este archivo es necesario.

# Herramienta JCreator.

El software utilizado para la compilación de este software es el JCreator versión 4.50.010. El archivo setup de instalación se encuentra en la carpeta tools ECATP.

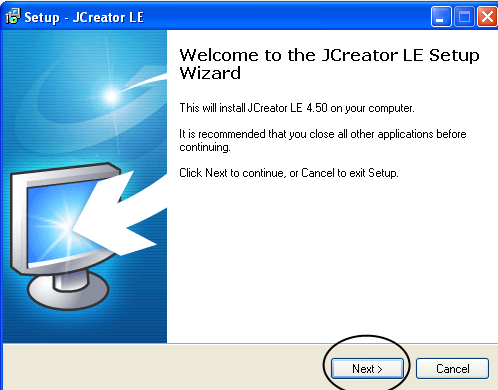
*Instalación del software:*

Este programa es sencillo de utilizar, simplemente se corre el archivo jcrea450\_setup.exe, se procede a su instalación:



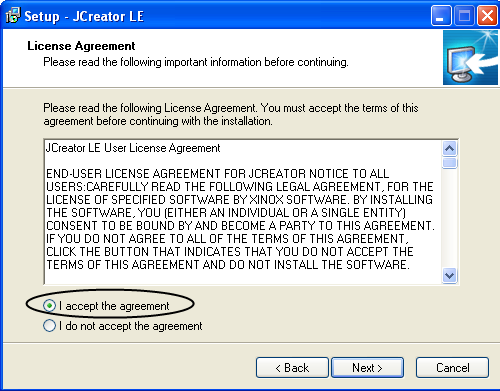
**Figura 3.** Archivo de instalación.

Se da doble click sobre el ejecutable (Figura 3) y aparece en pantalla como se ve en la figura 4:



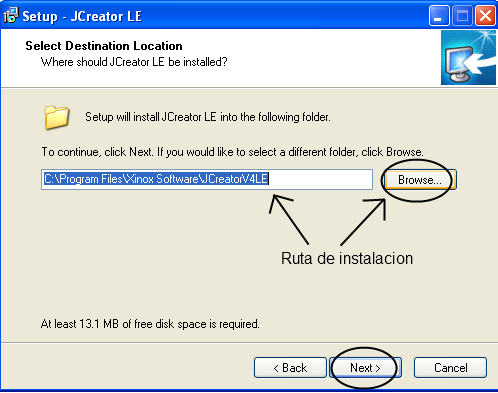
**Figura 4.** Instalación de JCreator.

Se debe seleccionar next y luego se selecciona I accept the agreement como se ve en la figura 5.



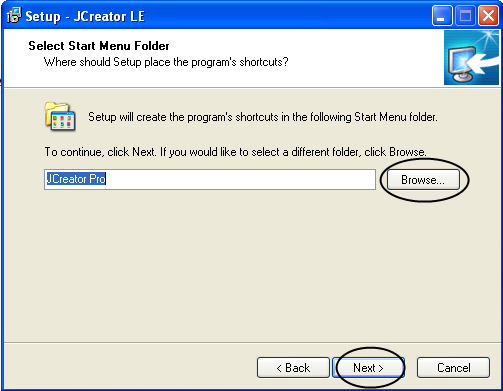
**Figura 5.** Instalación de JCreator.

Una vez aceptado el acuerdo aparece en pantalla como se observa en la figura 6. Aca se selecciona la ruta de instalación del programa. Mediante Browse se modifica dicha ruta. Luego se selecciona next.



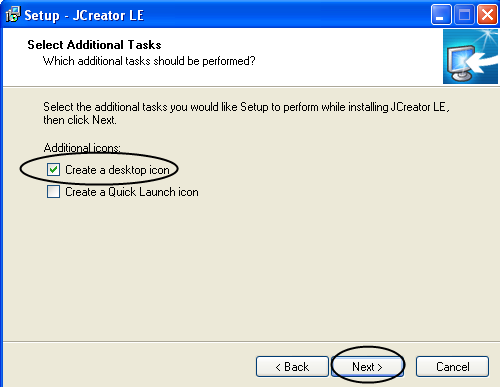
**Figura 6.** Instalación de JCreator.

En la siguiente pantalla se selecciona la carpeta de instalación como se observa en la figura 7.



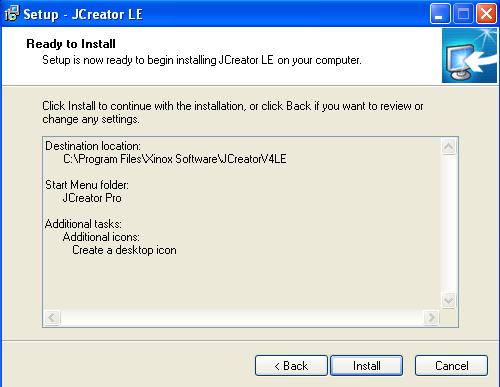
**Figura 7.** Instalación de JCreator.

Luego de seleccionar next en la figura 7, en la siguiente pantalla aparece las opciones para crear iconos en el escritorio o de quick launch como se observa en la figura 5n.



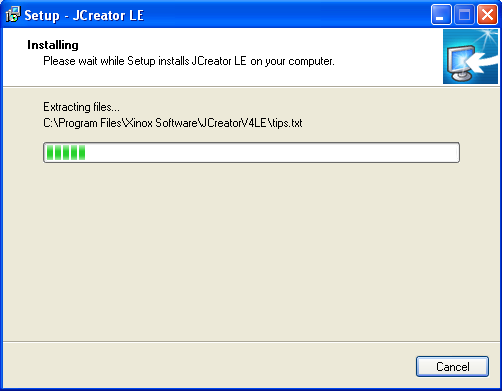
**Figura 8.** Instalación de JCreator.

Luego de dar next en la en la figura 8, aparece en pantalla el resumen de la instalación en caso de tener que cambiar algún parámetro, como se observa en la figura 9.



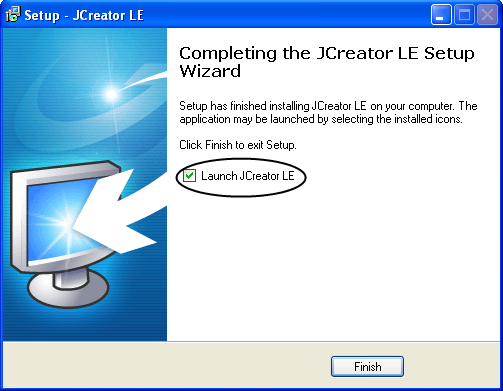
**Figura 9.** Instalación de JCreator.

En caso de que toda la información este bien, simplemente le damos next. En este punto inicia el proceso de instalación (figura 10).



**Figura 10.** Instalación de JCreator.

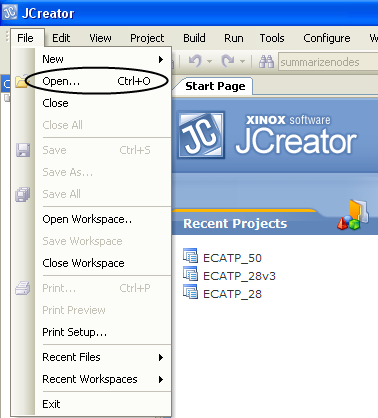
Cuando la instalación aparece la opción si se desea abrir el JCreator, como aparece en la figura 11. En caso de no querer esto se quita la opción y se presiona finish.



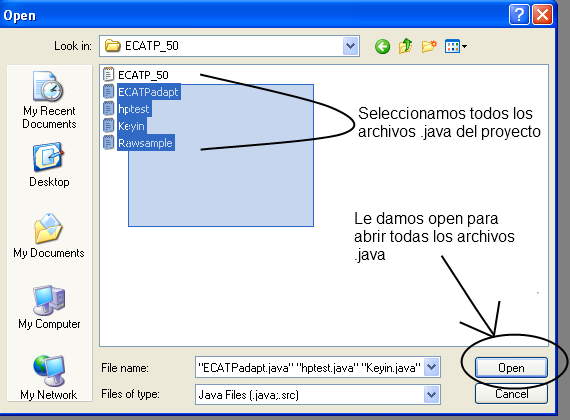
**Figura 11.** Instalación de JCreator.

# Modificación de software ECATP\_50.

En caso de querer realizar modificaciones del software, una vez instalado el JCreator, se procede a abrir el programa (JCreator LE). En la esquina superior izquierda, aparece en la barra la opción File, se selecciona open, y buscamos el proyecto que se desea abrir. (Figura 12 y 13)

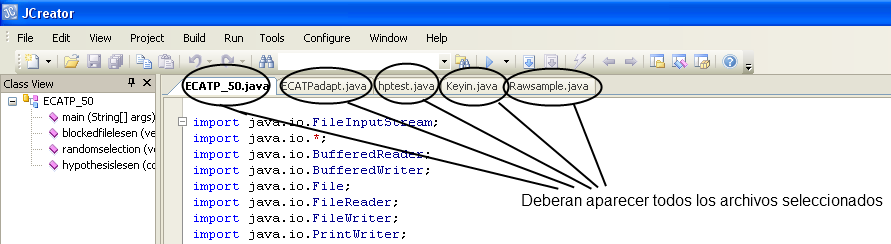


**Figura 12.** Uso del JCreator.



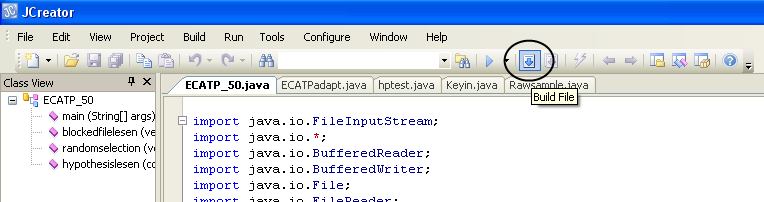
**Figura 13.** Uso del JCreator.

En este caso se busca última versión ECATP (ECATP\_50). Una vez abierto todos los archivos deberán aparecer como pestañas por debajo del menú principal (figura 14)



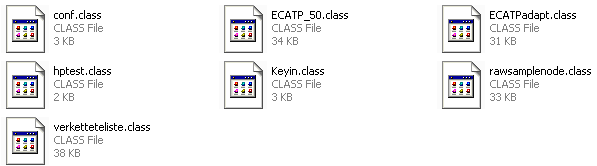
**Figura 14.** Uso del JCreator.

Ahora ya podemos modificar el código de programación de la herramienta ECATP\_50. Si se realiza algún cambio en la programada, se debe compilar mediante el botón con la flecha hacia abajo (figura 15).



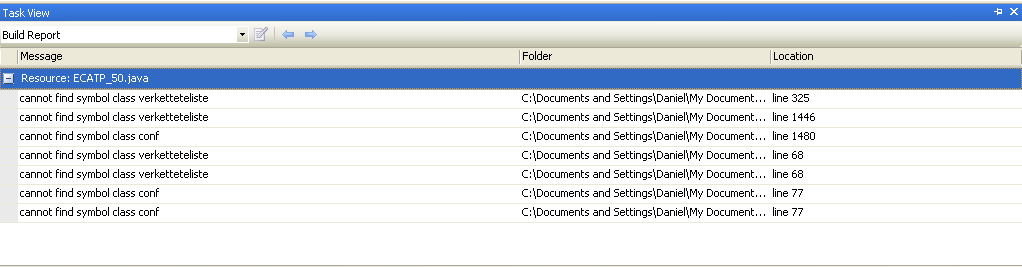
**Figura 15.** Uso del JCreator.

Para cada clase existente en los archivos .java, a la hora de presionar el botón Build File, se crean los archivos .class (Figura 16). Estos archivos son los requeridos para correr el programa, desde el command prompt.

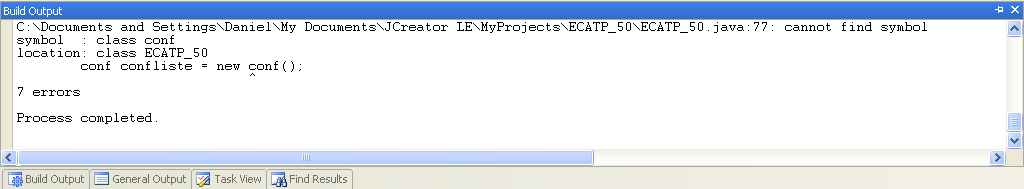


**Figura 16.** Uso del JCreator.

Si en la carpeta donde se abrió el proyecto no existen los archivos .class y se da un Build File al ECATP\_50, se generara un error debido a que este es un archivo el cual contiene el programa principal (main), por lo que en el Task view (figura 17) y en el Build output (figura 18) aparece un error de cannot find symbol class y el nombre de la clase que no posee el archivo .class. Esto también sucede con el ECATPadapt.

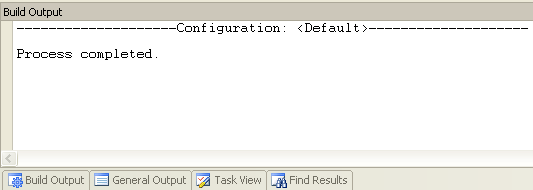


**Figura 17.** Mensajes en el Task View.



**Figura 18.** Mensajes en el Build Output.

Esto se soluciona construyendo los archivos .class con el botón Build File, de la clase que hace falta en el proyecto. Una vez realizado esto, ya se puede presionar el botón de Build File del main, en ese caso debe aparecer en el Build Output process completed, como se observa en la figura 19.



**Figura 19.** Mensajes en el Build Output.

En caso de querer realizar una modificación del software para aumentar a mas de 50 frecuencias se deben de seguir los pasos internos que se encuentran dentro de la programación como Notas. Los archivos que se deben modificar son: ECATP\_50.java, ECATPadapt y Rawsample.java

# Descripción de las clases existentes en el programa.

## Clase ECATP\_50.

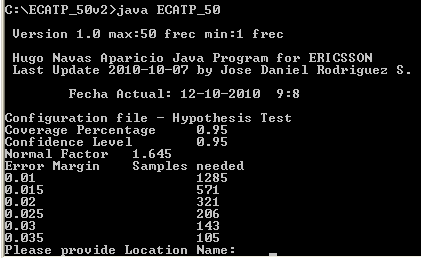
### Método principal (main()).

#### Diagrama de bloques.



**Figura 20**. Diagrama de Bloques del método main del programa.

El diagrama de bloques que se observa en la figura 20 corresponde al diagrama general del programa, en el primer bloque aparece la impresión del inicio del programa como se observa en la figura 21.



**Figura 21.** Inicio del programa, esperando variables.

Una vez introducido las variables iniciales, comienza el proceso de ejecución del programa y una vez finalizado este, generara 3 archivos finales: reporte final en formato HTML, otro archivo con el nombre xxxx\_Internal\_Ericsson y un último archivo con el nombre xxxx\_Random\_Segments el cual como su nombre lo indica contiene los segmentos aleatorios tomados del archivo original.

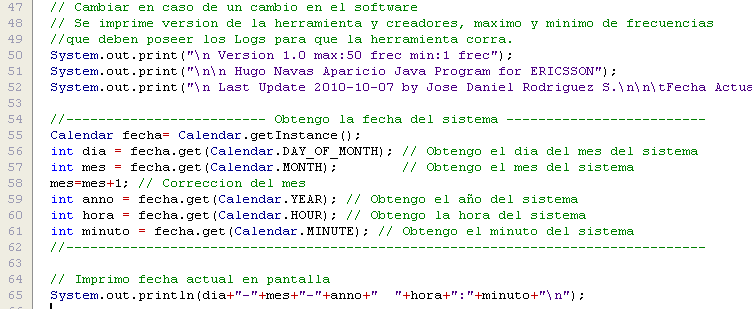
#### Diagrama de Flujo.



**Figura 22**. Diagrama de flujo del método main del programa.

#### Código.

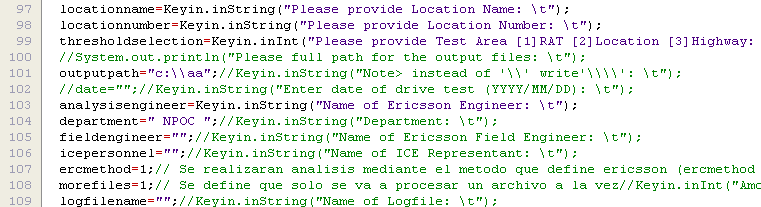
Si se observa el código de la clase principal (ECATP\_50.java), Se observa en la línea 50 la impresión en pantalla de los creadores de la herramienta (ver figura 17n), también se puede observar la fecha actual (fecha del sistema). Todo esto mediante un System.out.print(). Como se puede observar en la figura 23.



**Figura 23.** Líneas de Código del main.

Después de esto se crea la lista concatenada, y se inicializa su primer valor. Luego se procede a realizar la impresión del menú principal para realizar el post-proceso.

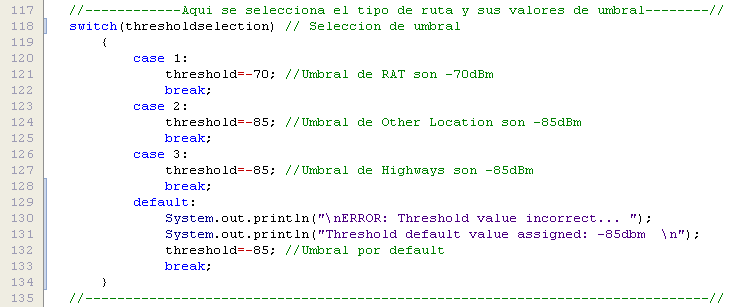
En esta sección se solicita información al usuario tal como: Nombre del lugar, numero del lugar, selección de umbral (RAT, location o Highway) y nombre del ingeniero que realiza el post-proceso.



**Figura 24.** Líneas de Código de solicitud de información inicial (main).

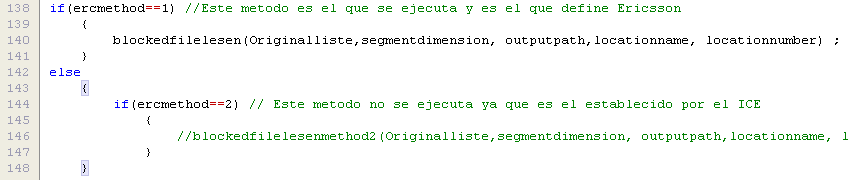
En este caso como se observa en las líneas de código de la 104 a la 109, se establece por defecto el departamento (NPOC), el ingeniero de campo y personal del ice se deja en espacio vacio. Se deja por defecto el método ercisson 1, y solamente la posibilidad de correr un archivo.

El nivel de umbral se elige mediante las opciones RAT, Location y Highway previamente solicitadas, en la línea 118 de la figura 25, se realiza la conversión a valores de potencia de la opción escogida. Esto selección se manipula mediante la variable declarada en el main con el nombre thresholdselection. En caso de haber ingresado un valor inválido de umbral, se le asigna por defecto un umbral de -85dBm.



**Figura 25.** Nivel de umbral.

Continuando con el código se ejecuta el método blockedfilelesen() (Lectura del archivo)

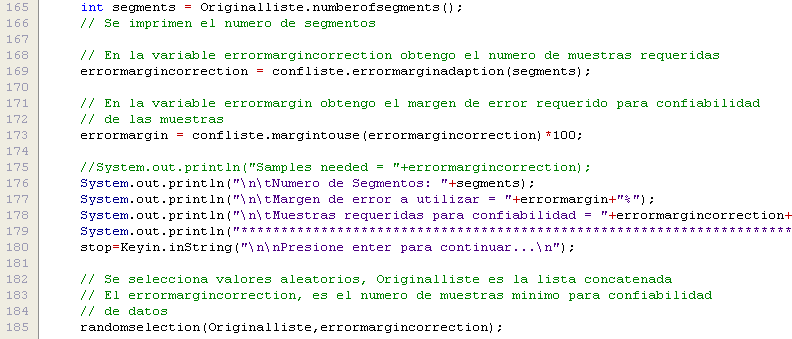


**Figura 26.** Escogencia de Método.

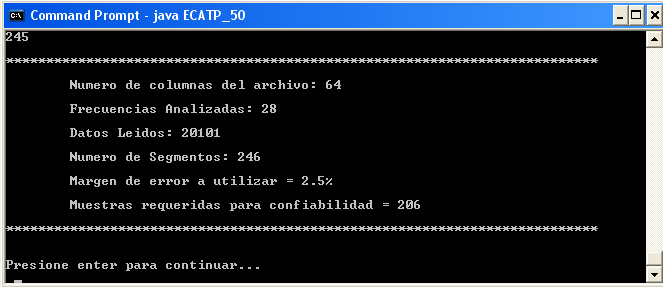
Dicho método recibe como argumentos de entrada Originalliste (que es una instancia de verketteliste, lista concatenada), segmentdimension (esta variable es la dimensión de segmento, en este caso su valor fijo es de 10 metros), outputpath (ruta de salida), locantionname (nombre del lugar), y por ultimo locationnumber (estas dos ultimas variables introducidas por el usuario en el menú principal).

En este método (blockedfilelesen()), se procesa la información del archivo original, la cual es almacenada en un lista concatenada. Aquí se realiza la creación del archivo xxxx\_Internal\_Ericsson.txt

En el código de la figura 27, en la línea 165 se obtiene la cantidad de numero de segmentos totales de la lista concatenada mediante el método numberofsegments() de la clase verketteliste, se obtiene además se obtiene el margen de error en porcentaje (línea 173). A continuación se da la impresión de resultados intermedios del proceso y se espera por tecla enter (línea 180, ver figura 28). Luego se escogen los segmentos aleatoriamente mediante la función randomselection, que recibe como argumentos de entrada la lista concatenada original y errormargincorrection que se elige conforme a la cantidad de mediciones realizadas por TEMS (versión 8 o 10).

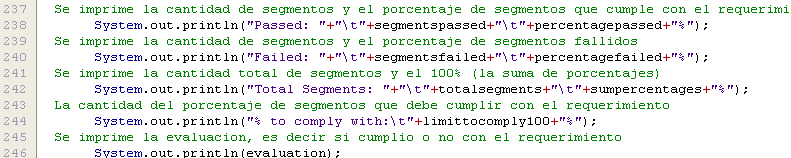


**Figura 27.** Código del main.



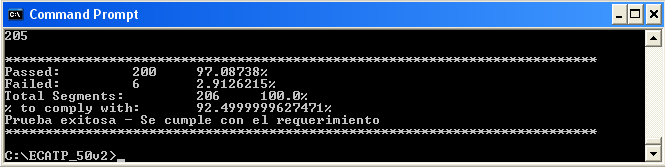
**Figura 28.** Ejecución del programa.

Luego se obtiene los valores de la cantidad de segmentos que pasan la prueba y los que la fallan y se genera un reporte en el sistema final como se puede observar en el código de la figura 29.



**Figura 29.** Código del main.

Este código (figura 29) genera el reporte final en el sistema como se observa en la figura 30.



**Figura 30.** Ejecución del programa.

Por dentro del código main se realiza la generación del reporte HTML final, que se imprimirá para presentar ante el ICE.

### Método blockedfilelesen ().

#### Diagrama de bloques.



**Figura 31**. Diagrama de Bloques del método blockedfilelesen ().

#### Diagramas de Flujo.



**Figura 32**. Diagrama de Flujo general del método blockedfilelesen del programa.



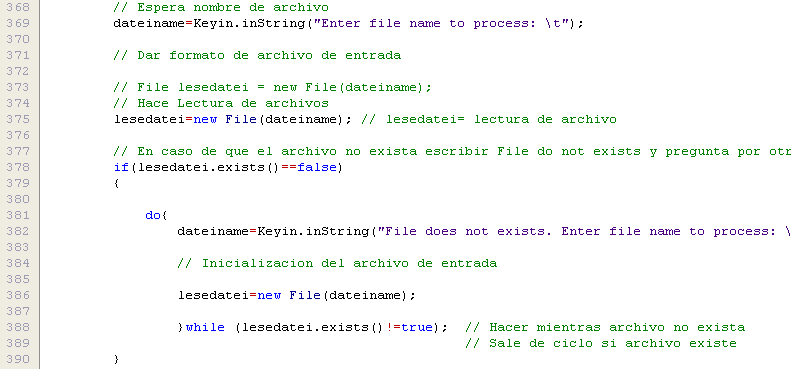
**Figura 33**. Diagrama de Flujo Lectura interna del archivo txt, del método blockedfilelesen del programa.



Figura 34. Diagrama de Flujo de la copia de valores de leser del método blockedfilelesen del programa.

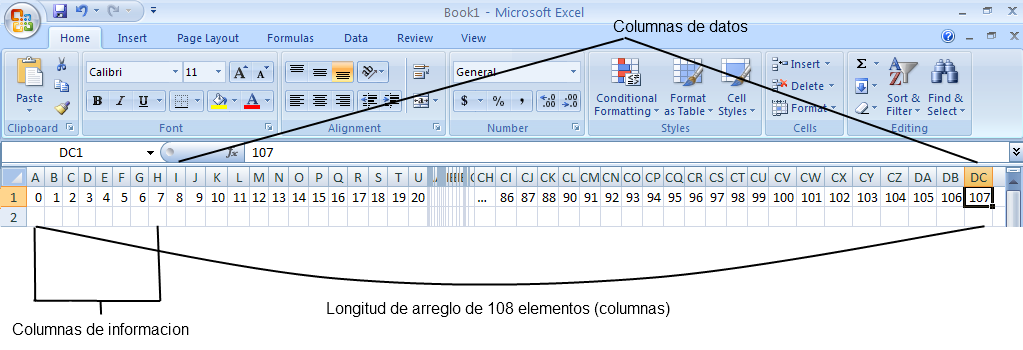
#### Código.

Dicho metodo se queda esperando por nombre del archivo a procesar, verifica si este existe si no es asi vuelve a preguntar por nombre del archivo a procesar. Esto se observar en el codigo la figura 35 (ver tambien diagrama de la figura 32).



**Figura 35.** Código del método blockedfilelesen.

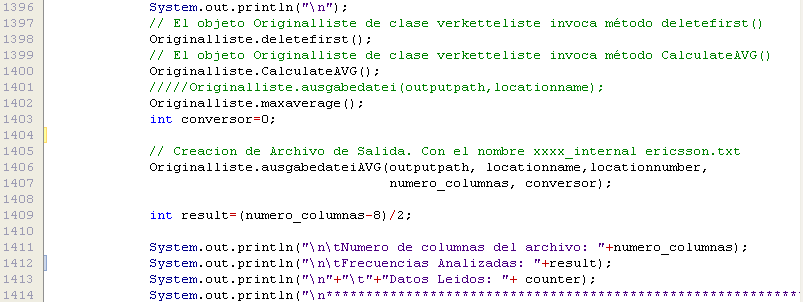
Luego se procede a realizar la lectura del archivo mediante el BufferReader, el archivo txt, en donde las columnas se dividen en tabulaciones como se observa en la figura 36. En donde el archivo podra contener un maximo de 108 columnas, las cuales las primeras 8 correspondran a columnas de informacion tal como latitud y longitud de ubicación y las siguientes 100 correponde a los valores de frecuencias analizadas y el nivel de potencia medido mediante TEMS.



**Figura 36**. Valores del arreglo Leser (lector de filas).

El archivo es leído por líneas independientes (filas) y esta información es almacenada en un arreglo denominado leser (lector), el cual contiene el numero de columnas, dependiendo del numero de frecuencias analizadas (igual o menor a 50 frecuencias).

En este método se realiza la creación de la lista concatenada utilizando el método insertend () para ingresar elementos en la lista y llenar la lista. Al final de la lectura y el ingreso de las filas dentro de la lista se calcula el promedio de cada nodo mediante el método CalculateAVG (), además de encontrar el valor máximo de potencia de cada nodo mediante maxarcfn (). También es aquí donde se genera el archivo xxxxx\_Internal\_Ericsson mediante el método ausgabedateiAVG y por ultimo realizar la impresión de información en pantalla. Todo esto se muestra en la figura 37 y el diagrama de flujo de la figura 32.



**Figura 37.** Código del método blockedfilelesen.

### Método randomselection ().

#### Diagrama de Bloques.



**Figura 38**. Diagrama de bloques del método randomselection.

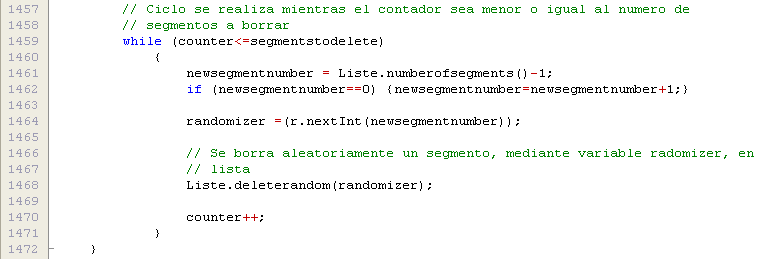
#### Diagrama de Flujo.



**Figura 39**. Diagrama de flujo del método randomselection.

#### Código.

Dicho método realiza la operación de eliminar el exceso de elementos de la lista para solo quedarse con la cantidad de segmentos requerida para confiabilidad de la medición según valores de estadística (dichos valores se encuentran dentro del archivo HPTC.txt). El ciclo para eliminar segmentos es que aparece en la figura 40. Dicho ciclo utiliza un while y la función deleterandom ().



**Figura 40.** Código del método randomselection.

### Método hypothesislesen ().

#### Diagrama de Bloques.



**Figura 41**. Diagrama de bloques del método hypothesislesen.

En dicho método se realiza la lectura del archivo HPTC.txt y se crea otra lista concatenada con los valores de estadística contenidos en este archivo. Cada fila será un elemento de la lista. Inicialmente dicha lista contendrá 7 elementos, a la hora de realizar un deletefirst (), tendrá los 6 elementos de la lista (elementos validos de la lista).

#### Diagrama de Flujo.



**Figura 42**. Diagrama de bloques del método hypothesislesen.

## Clase rawsamplenode.

En esta clase se definen los constructores de la muestra del nodo que es manejado como una lista concatenada. El primer constructor definido es el que se muestra en la figura 43.

### Constructores Rawsamplenode.



**Figura 43**. Diagrama de Constructor 1 rawsamplenode.



**Figura 44**. Diagrama de Constructor 2 rawsamplenode.



**Figura 45**. Diagrama de Constructor 3 rawsamplenode.

### Métodos Gets ().

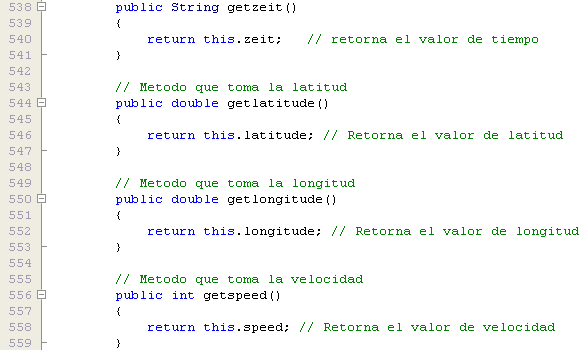
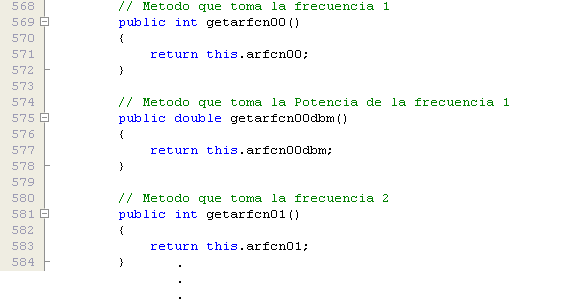
#### Diagrama.



**Figura 46**. Diagrama de métodos gets () de la clase rawsamplenode.

#### Código.

La función de dichos métodos es la de tomar el valor de la variable requerida (ya sea zeit, latitud, longitud), que se encuentre en ese momento como la variable de la clase con el valor necesario. Este código se encuentra en el código de la figura 47.

**Figura 47.** Código de los métodos gets ().

### Métodos Sets ().

#### Diagramas.



**Figura 48**. Diagrama de métodos sets () de la clase rawsamplenode.

En las figuras 49, 50 y 51 aparecen los Diagramas internos de los setnode ().



**Figura 49**. Diagrama de setnode () de la clase rawsamplenode.



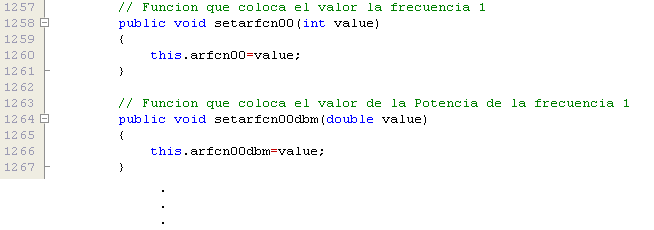
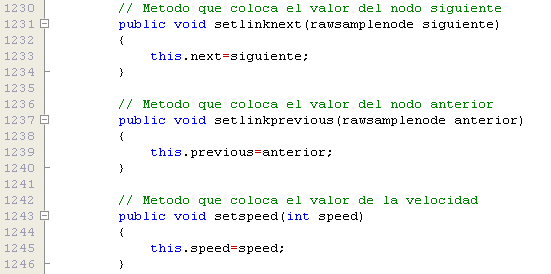
**Figura 50**. Diagrama de setnode1 () de la clase rawsamplenode.



**Figura 51**. Diagrama de setnode2 () de la clase rawsamplenode.

#### Código.

Los métodos set sirven para ingresar valores dentro de las variables de la clase (utilizando this). Estos métodos deben de poseer como argumentos de entrada la variable la cual se quiere modificar dentro de la clase.



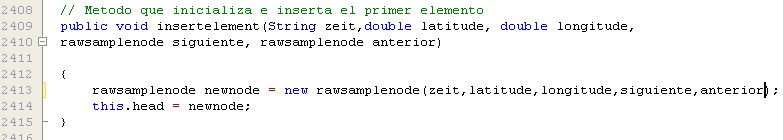
**Figura 52**. Diagrama de métodos sets () de la clase rawsamplenode.

## Clase verketteteliste.

### Método insertelement ().

#### Código.

Para insertar elementos dentro de una lista primeramente se debe de crear nodo y declarar de que tipo es el objeto, por ello en el código se deberá establecer mediante un new en este caso al requerir uno de tipo nodo, new va seguido del nombre rawsamplenode. Ver código de la figura 53.

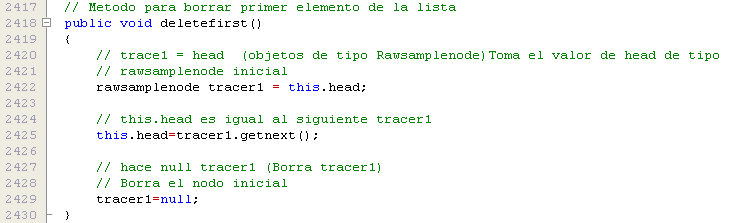


**Figura 53**. Código de insertelement () de la clase verketteliste.

### Método deletefirst ().

#### Código.

Este método solo se utilizara al inicio del programa, ya que su función es la de borrar el primer elemento de la lista. Esto se realiza debido a que para inicializar valores de la lista se inserta un nodo con inicialización de variables, el cual no posee información relevante dentro del proceso, por lo que, una vez creada la lista se elimina este nodo inicial.



**Figura 54**. Código de deletefirst () de la clase verketteliste.

### Método insertend ().

#### Diagrama de Bloques.



**Figura 55**. Diagrama de insertend () de la clase verketteliste.

Para insertar nuevos nodos dentro de la lista, se debe hacer al final de esta misma, cada vez que se realiza esto se debe hacer una revisión constante de la distancia recorrida, ya que se insertara un nuevo nodo en caso de que, se hayan recorrido igual o mas de 10 metros. Esto debido a que cada segmento mide 10 metros, todo esto es según especificación ICE. Como aparece en el diagrama de la figura 55.

Si un nuevo nodo esta dentro del mismo segmento, los valores contenidos en dicho nodo se suman al nodo que se encuentra dentro de este segmento. Se sigue así hasta pasar al siguiente nodo. Luego mediante el método CalculateAVG (), se realizara el calculo real los valores promedio.

En caso de que un nuevo nodo corresponda a un nuevo segmento, se deben establecer los enlaces de dicho nodo.

#### Diagrama de Flujo.



**Figura 56**. Diagrama de insertend () de la clase verketteliste.

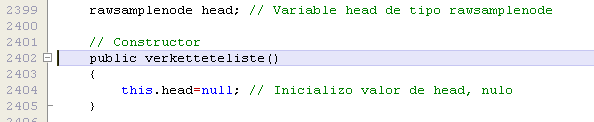
#### Código.

Este método es donde se insertan valores dentro de la lista concatenada, dicho proceso se puede observar en la figura 57. En donde se puede observar el proceso de inserción de los nodos nuevos.



**Figura 57**. Diagrama de formación de la lista concatenada.

Para poder iniciar este proceso es necesario declarar el constructor dentro de la clase verketteliste, en donde el valor inicial de head que es de tipo rawsampenode debe inicializarse con un valor nulo (null, ver figura58).



**Figura 58.** Código para inicialización de la lista concatenada (verketteliste).

En insertend () es en donde se toma la decisión de si crear o no un nuevo nodo, esto dependiendo de si en el segmento interno la distancia recorrida es mayor a diez metros (figura 59).



**Figura 59**. Diagrama lista concatenada avanzada.

### Método nodedistancecheck ().

#### Diagrama de Bloques.



**Figura 60**. Diagrama de bloques del método nodedistancecheck () de la clase verketteliste.

Este método es que se requiere para encontrar la distancia recorrida entre un punto con sus coordenadas geográficas y otro punto. Dicho método se utiliza dentro del método insertend ().

#### Diagrama de Flujo.



**Figura 61**. Diagrama de flujo del método nodedistancecheck () de la clase verketteliste.

#### Código.

En este código se revisa la distancia recorrida, en caso de pertenecer a un segmento nuevo se inserta el nuevo nodo encontrado (en este caso el puntero del nodo tracer1 se convierte en el puntero de nuevo nodo, y el nuevo nodo es insertado dentro de la lista concatenada, ver figura 62).



**Figura 62**. Diagrama del proceso de inserción de nuevos nodos en la lista concatenada.

### Método summarizenodes ().

#### Diagrama de Bloques.



**Figura 63**. Diagrama de bloques del método summarizenode () de la clase verketteliste.

#### Diagrama de Flujo.



**Figura 64**. Diagrama de bloques del método summarizenode de la clase verketteliste.

#### Código.

En dicho método se establece el resumen de un nodo, en el ejemplo de la figura 65, el puntero nodo tracer1, se encuentra apuntando al último valor de la lista diferente de nulo, la distancia recorrida en dicho segmento es menor a 10 metros, por lo tanto debe de ejecutarse el método summarizenodes (), el cual toma los valores del segmento apuntado por tracer1 y los valores de el nodo nuevo, y realiza una sumatoria de valores de los ambos nodos. Lo cual dejara como resultado un solo nodo, existente. En caso de que se nuevo sea parte de otro segmento, dichos valores se insertan al final de la lista.

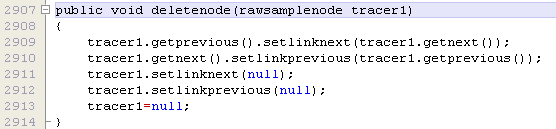


**Figura 65**. Ejemplo de una lista concatenada.

### Método deletenode ().

#### Código.

El código de este método se encuentra en la figura 66.



**Figura 66.** Código del método deletenode.

Dicho código realiza la función de eliminar cierto nodo (el cual se elige mediante el valor que insertemos en el argumento del método) como se observa en la figura 67 (Diagrama 2 de la figura). El cual para eliminar un nodo se debe realizar los siguientes pasos:

1. Eliminar el enlace de nodo siguiente del nodo previo (que actualmente se encuentra apuntando al nodo que vamos a eliminar) y enlazar al nodo siguiente al que vamos a eliminar.
2. Eliminar el enlace de nodo previo del nodo siguiente (que actualmente se encuentra apuntando al nodo que vamos a eliminar) y enlazar al nodo previo al que vamos a eliminar.
3. Hacer el enlace de nodo previo del nodo que vamos a eliminar como nulo.
4. Hacer el enlace de nodo siguiente del nodo que vamos a eliminar como nulo.
5. Y asignamos un valor de nulo al nodo que se esta eliminando.



**Figura 67.** Diagramas del proceso del deletenode.

### Método ausgabedateiAVG ().

#### Diagrama de Flujo.

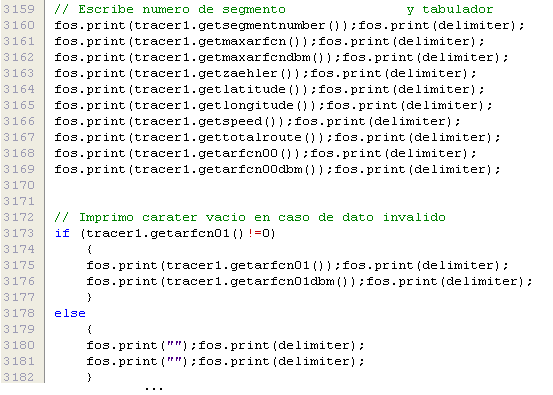


**Figura 68**. Diagramas de flujo del método ausgabedatei () de la clase verketteliste.

#### Código.

En este método se realiza la creación del archivo xxxx\_Internal\_Ericsson.txt, el cual es un archivo que se utiliza en caso de que el recorrido no haya pasado la prueba para modificar ruta mediante TEMS y luego realizar proceso con ECATPadapt.

Para realizar una escritura dentro del archivo xxxx\_Internal\_Ericsson.txt, se realiza mediante el escritor PrintWriter, el cual en este método esta definido con el nombre de fos, como se observa en el código de la figura 69.



**Figura 69**. Código interno del método ausgabedateiAVG ().

### Método CalculateAVG ().

#### Diagrama de Flujo.



**Figura 70**. Diagramas de flujo del método CalculateAVG de la clase verketteliste.

#### Código.

En dicho método como su nombre lo indica se calcula el promedio de las variables de los nodos. En este caso la sumatoria total de los valores dentro del nodo se divide entre cantidad de nodos resumidos en un segmento (figura 71).



**Figura 71**. Código interno del método CalculateAVG ().

### Método maxaverage ().

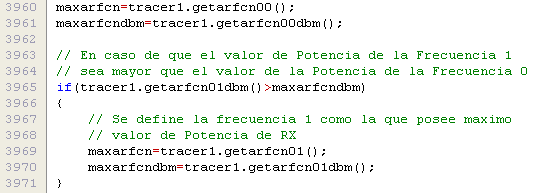
#### Diagrama de Flujo.



**Figura 72**. Diagramas de flujo del método maxaverage de la clase verketteliste.

#### Código.

Este método toma los valores de potencia de cada frecuencia dentro de un segmento y encuentra el valor máximo y la frecuencia de trabajo.



**Figura 73**. Código interno del método maxaverage ().

### Método numberofsegments ().

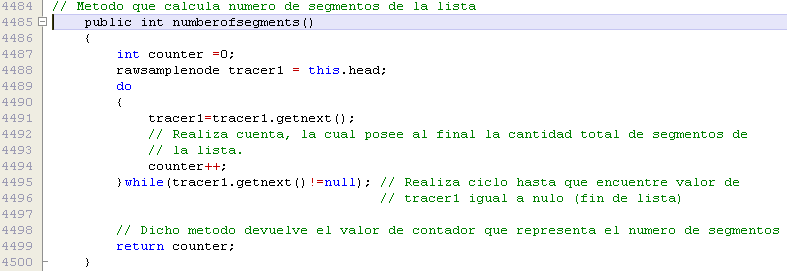
#### Diagrama de Flujo.



**Figura 74**. Diagramas de flujo del método numberofsegments de la clase verketteliste.

#### Código.

Este método encuentra la cantidad total de segmentos, haciendo un recorrido desde nodo inicial hasta nodo final de la lista, en cada corrimiento se suma un uno a contador ().

****

**Figura 75**. Código interno del método numberofsegments ().

Este método devuelve el valor de contador en el programa para saber cuantos segmentos posee la lista.

### Método deleterandom ().

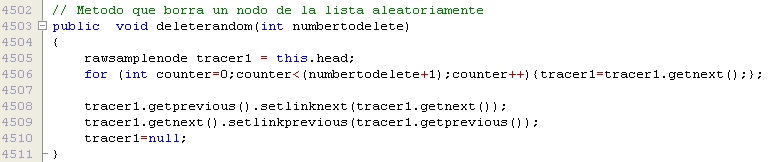
#### Diagrama de Flujo.



**Figura 76**. Diagramas de flujo del método deleterandom de la clase verketteliste.

#### Código.

Este método recibe como variable de entrada numbertodelete, el cual como su nombre lo indica es será el numero de segmento a borrar dentro de la lista concatenada.



**Figura 77.** Código de deleterandom.

### Método ausgabedateiRNDM ().

#### Diagrama de Flujo.



**Figura 78**. Diagramas de flujo del método ausgabedateiRNDM () de la clase verketteliste.

#### Código.

Este método se crea el archivo xxxx\_Random\_Segments.txt. La primer fila a escribir dentro del archivo, es la que contiene los nombres de las columnas, esto se puede observar en el código de la figura 79.



**Figura 79**. Código de ausgabedateiRNDM ().

Es se realiza mediante un PrinterWriter que este caso es designado con el nombre de fos, como se observa en la línea de código 4542 de la figura 79.

### Método passedsegments ().

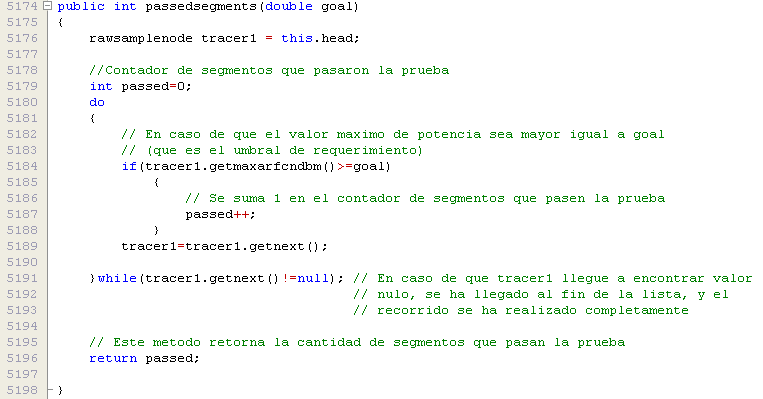
#### Diagrama de Flujo.



**Figura 80**. Diagramas de flujo del método passedsegments de la clase verketteliste.

#### Código.

En este método se cuenta la cantidad de segmentos que pasan la prueba, dicha cantidad se guarda en la variable con el nombre de passed como se observa en la línea de código 5179 de la figura 81.



**Figura 81**. Código de passedsegments.

### Método failedsegments ().

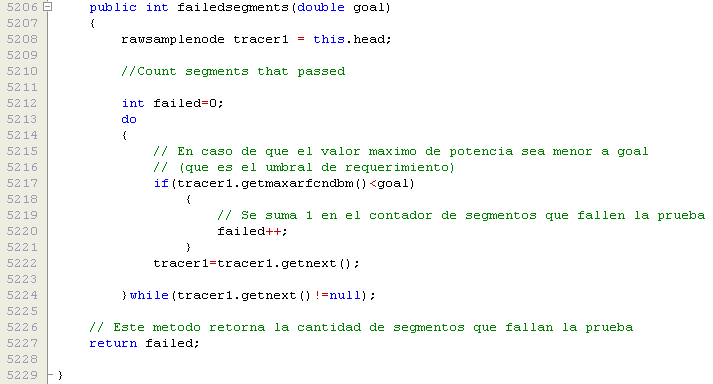
#### Diagrama de Flujo.



**Figura 82**. Diagramas de flujo del método passedsegments de la clase verketteliste.

#### Código.

En este método se cuenta la cantidad de segmentos que fallen la prueba, dicha cantidad se guarda en la variable con el nombre de failed como se observa en la línea de código 5212 de la figura 83.



**Figura 83**. Código de failedsegments.

### Método insertendadapted ().

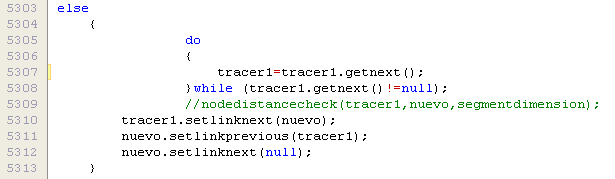
#### Diagrama de Flujo.



**Figura 84**. Diagramas de flujo del método insertendadapted de la clase verketteliste.

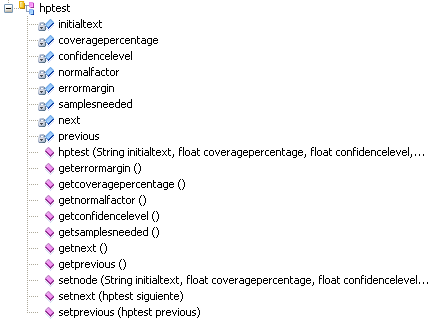
#### Código.

Dicho método realiza una función similar a insertend (). La diferencia es que este método es utilizado por ECATPadapted que utiliza el archivo TEMS modificado en caso que la ruta no pase la prueba. Este método utiliza además el setnode2 () de la clase rawsamplenode, mientras que el insertend () utiliza el setnode () de la clase rawsamplenode. Además en dicho método no es necesario realizar la revisión de distancia recorrida, solamente requiere realizar la inserción al final de la lista, ya que los nodos resumen y los segmentos ya han sido previamente calculados.



**Figura 85**. Código de insertendadapted.

## Clase hptest.

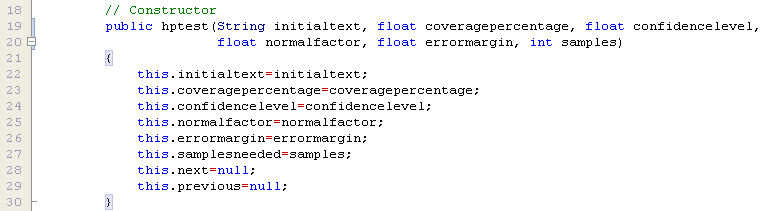


**Figura 86**. Métodos y variables de la clase hptest.

### Constructor hptest ().

#### Código.

Este método crea el constructor de nodo de la lista concatenada de los valores de configuración inicial. Como se puede observar en la figura 87.



**Figura 87**. Código de constructor hptest.

### Métodos Gets ().

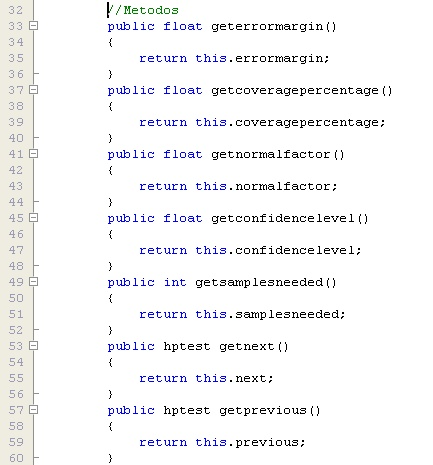
#### Diagramas.



**Figura 88**. Métodos gets () de la clase hptest.

#### Código.

Los métodos gets () como se nombre lo indican sirven para tomar los valores contenidos en un nodo especifico y poder manipular la información, por ello los métodos deben de contener un return dentro de sus llaves como se observa en el código de la figura 8.



**Figura 89.** Código de los Métodos gets () de la clase hptest.

### Métodos Sets ().

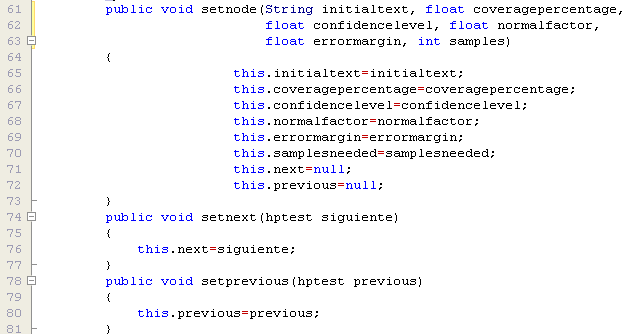
#### Diagramas.



**Figura 90**. Métodos sets () de la clase hptest.

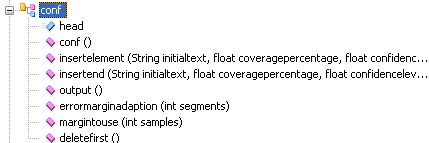
#### Código.

En estos métodos como su nombre lo indica sirven para establecer los valores dentro de los nodos, esto se realiza con un this.



**Figura 91**. Código de los Métodos sets () de la clase hptest.

## Clase conf.

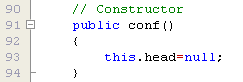


**Figura 92**. Métodos y variables de la clase conf.

### Constructor conf ().

#### Código.

Dicho constructor genera dentro de la clase conf una instancia de hptest, con el nombre de head y su valor inicializado es igual a nulo (null).

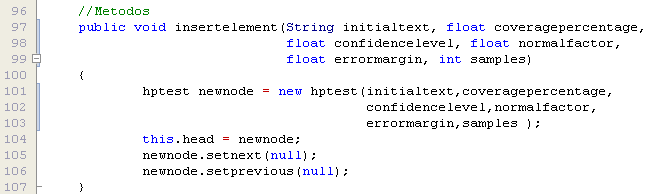


**Figura 93**. Código del constructor de la clase conf.

### Método insertelement ().

#### Código.

Dicho método sirve para crear nodos nuevos y establecer sus enlaces, esto mediante la palabra clave new y los setnext () y setprevious ().



**Figura 94**. Código del método insertelement () de la clase conf.

### Método insertend ().

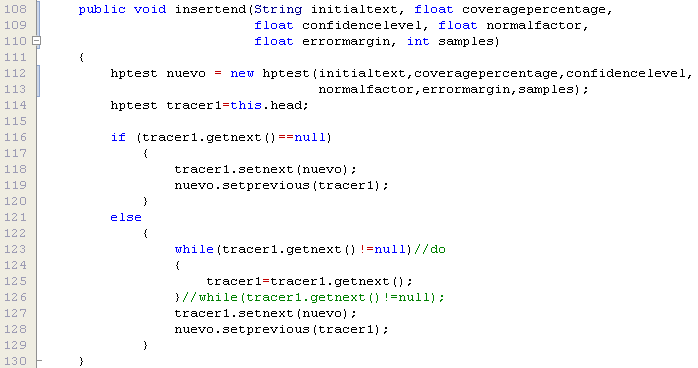
#### Diagrama de flujo.



**Figura 95**. Diagrama de flujo del método insertend () de la clase conf.

#### Código.

Dicho método inserta valores dentro de la lista, en el if del código de la figura 96 (línea de código 116). Funciona para insertar el primer valor de la lista el cual contendrá valores inválidos. En el else (fila 121) se insertan todos los demás elementos de la lista (6 líneas en el HPTC.txt implica seis elementos dentro de esta lista).



**Figura 96.** Código del método insertend () de la clase conf.

### Método output ().

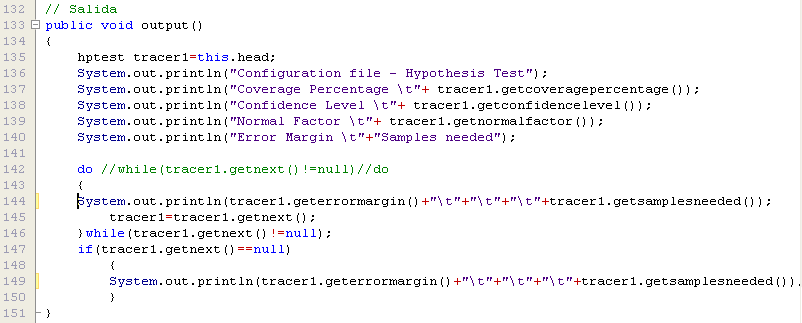
#### Diagrama de flujo.



**Figura 97**. Diagrama de flujo del método output de la clase conf.

#### Código.

La función output () sirve para imprimir en pantalla los valores de cada nodo de la lista concatenada conf, esto se muestra en el código de la figura 98.



**Figura 98**. Código del método output de la clase conf.

La lista de 6 elementos corresponden a las seis líneas dentro del archivo HPTC.txt, el cual contiene los nombres de las columnas y sus valores espaciados por un carácter de coma ”,”. Los valores de cada columna se pueden observar en la figura 99.



**Figura 99**. Datos del archivo de configuración inicial HPTC.txt

### Método errormarginadaption ().

#### Diagrama de flujo.

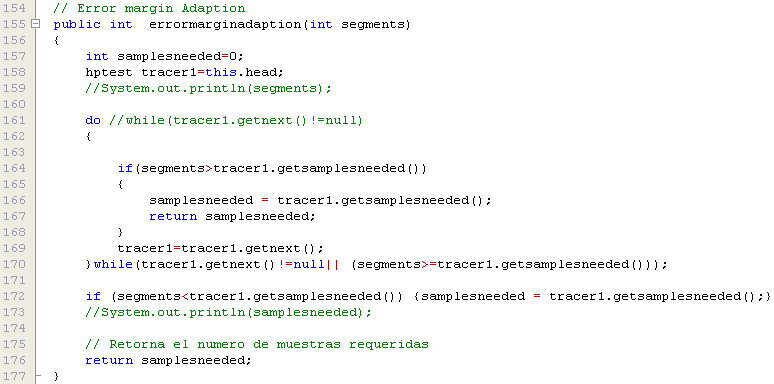


**Figura 100**. Diagrama de flujo del método errormarginadaption de la clase conf.

#### Código.

Este método recibe como argumento de entrada la cantidad de segmentos mediante la variable segments. Este método lo que realiza es la escogencia de la cantidad de segmentos a tratar, según la cantidad de segmentos encontrados en el archivo original generado por TEMS, en caso de que se tenga mas elementos de los necesarios se toman la cantidad de segmentos a tratar según la tabla del archivo HPTC.txt. En este archivo se contiene 6 posibles valores.

Deben de ser más de 105 muestras ya que es el mínimo establecido en el archivo de configuración HPTC, esto corresponde a una medición de una ruta mayor a 1050 metros. Este archivo contiene 6 posibles valores para la variable samplesneeded que son: 105, 143, 206, 321, 571 o 1285. El cual poseen un porcentaje de error de 0.035, 0.03, 0.025, 0.02, 0.015 o 0.01 respectivamente. Esto también se observa en la figura 99.



**Figura 101**. Código del método errormarginadpation () de la clase conf.

### Método margintouse ().

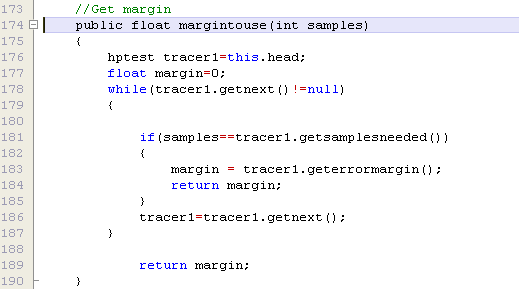
#### Diagrama de flujo.



**Figura 102**. Diagrama de flujo del método margintouse de la clase conf.

#### Código.

En dicho método se realiza un recorrido de la lista conf el cual compara muestras con valor del nodo apuntado, si son iguales extra el valor del margen de error, sino se apunta al siguiente nodo (figura 103 línea 181).

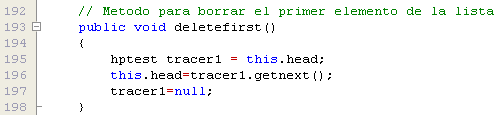


**Figura 103**. Código del método margintouse de la clase conf.

### Método deletefirst ().

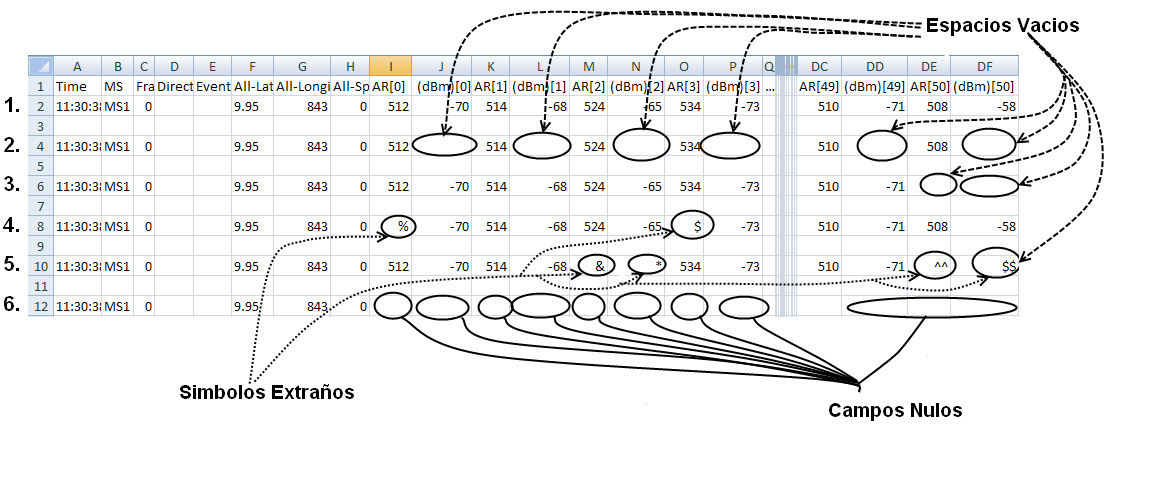
#### Código.

Dicho método es utilizado para eliminar el primer elemento de la lista el cual contiene valores inválidos.



**Figura 104**. Código del método deletefirst de la clase conf.

# Tipo de archivos que soporta la Herramienta.



**Figura 105**. Formato del archivo leído visto en Excel.

En la figura 105:

1. La primer fila del archivo txt visto en Excel corresponde al nombre de las columnas. Esta primera fila es leída pero sus valores no son procesados.
2. En la segunda fila (al lado del 1.), aparece el formato normal que soporta la herramienta. Posee todos los campos llenos con valores validos en todas sus columnas, por lo que dicha valores de la fila será ingresados dentro de la lista concatenada.
3. En la fila 4 (al lado del 2.), aparece una línea con valores dbm en blanco, esta fila será ignorada por el programa y se pasara a la siguiente lectura, sin ingresar estos valores dentro de la lista concatenada
4. En la fila 6 (al lado del 3.), al final de esta aparecen valores en blanco, en este caso, se procesara en la lista concatenada y se colocaran valores en estas columnas de 0 y -150 dBm con el fin de no afectar el resto de mediciones de dicha fila.
5. En la fila 8 (al lado del 4.), aparecen símbolos que no se podrán convertir de una cadena de caracteres a un entero (parse.int()). En este caso debido a que sucedió en la columna de ARFCN[0], dicha fila será ignorada por el sistema debido a que este valor es convertido de un “%” a un valor de 0 entero. (Esta fila no es procesada dentro de la lista concatenada)
6. En la fila 10 (al lado del 5), aparecen símbolos extraños en ARFCN[2], ARFCN[2]dBm, ARFCN[50] y ARFCN[50]dBm, por lo que dichos valores son cambiados por “0” y -150dBm, para que estos no afecten las mediciones de los demás valores, por lo que los valores de dicha fila serán ingresados dentro de la lista concatenada.
7. En la fila 12 (al lado del 6), aparecen los valores de ARFCN[0] y ARFCN[0]dBm hasta ARFCN[49] y ARFCN[49]dBm con valores nulos. Dicho fila será ignorada y no se ingresaran valores dentro de la lista concatenada.