**Extern library:**

* Chương trình dùng [JDBC41 Postgresql Driver, Version 9.3-1102](http://jdbc.postgresql.org/download/postgresql-9.3-1102.jdbc41.jar) để connect và query giữa java – Postgresql
* Dùng <http://www.jfree.org/jfreechart/> để biểu diễn dữ liệu và tính toan trên đồthị
* Dùng <https://code.google.com/p/jsyntaxpane/> để hiển thị log-status với syntax highlight
* Dùng <http://toedter.com/jcalendar/> để chọn ngày giờ
* Dùng [javax.swing.SwingWorker](http://docs.oracle.com/javase/tutorial/uiswing/concurrency/worker.html) để tạo multi threads background task cho trương trình
* Dùng <http://www.eclipse.org/windowbuilder/> Plugin cho Eclipse để tạo dao diện cho trương trình

**Mô tả chương trình:**

* Từ dòng 3->71 là import các extern / intern librarys cần thiết cho trương trình
* Từ dòng 78 -> 143 là khai báo các biến cho class Density\_Calc . các biến này có global scope cho Class này.
* Từ dòng 147->158 là trương trình chính (java app nào cũng có void main này)
* Từ dòng 164->622 là tạo dao diện cho trương trình: public Density\_Calc(). Trong phần này thì các button, label, textbox, combobox, progressbar …….. sẽ được tạo ra trong Jframe , các actionlistener sẽ được thêm vào cho các button => khi button click thì các task tương ứng sẽ được chạy. Tất cả phần này sẽ được gọi trong void main ở trên (dòng 151)
* **Phần code quan trọng bắt đầu từ dòng 624-> hết**

624: // function append text to Jeditorpane

**public** **void** append\_txt(JEditorPane ed, String s){…}

cái này để hiện thêm 1 đoạn text vào log box bên phải của Load Density from DB tab

634: //get index of one value in JList

**public** **int** get\_index\_of(DefaultListModel a, Object value){

cái này sẽ return index của 1 Object trong jList. Nếu ko tìm thấy thì return -1.

643: // function to reset progressBar status

**public** **void** reset\_progress(){

reset lại thanh progressBar

650: // function to reset progressBar\_1 status

**public** **void** reset\_progress1(){

reset lai thanh progressBar\_1

655: //funtion to calculate distand between to point

**public** String distand\_between(String site1, String site2){

Tính khoảng cách giữa 2 may đo ra km. Cái này sẽ chạy 1 câu lệnh SQL trong DB và tính khoảng cách ra mét rồi chia 1000.

682: //function to get time periode between two time point : return in minute

**public** String diff\_in\_minute(String first\_time, String last\_time){

Tính thời gian = minute giữa 2 thời điểm bất kỳ.

707: /\*\*

\* funtion to calculate if next point has Density Sprung

\* return null or array[site,tsp] of next sprung point

\*/

**public** String[] next\_sprung(String next\_point, String start\_time,**int** min\_duration\_in\_minute, **int** max\_duration\_in\_minute, **float** sprung, **int** fahr\_streifen, String fahrzeug){…}

Function này để tính cho Stoßwelle. cái nay tính trong khoảng thời gian từ ***min\_duration\_in\_minute*** đến ***max\_duration\_in\_minute*** cho 1 máy đo kế tiếp xem có density sprung >= điều kiện đưa vào ko. Nếu có thì sẽ reuturn lại tên máy đo và thời gian của điểm có Sprung. Tùy theo loại Xe (PKW, LKW, ALL và làn đường thí se tính khác nhau).

802: /\*\*

\* purpose for this Class: to save next Sprung point in list to display continuous Stoßwelle-Strecke

\*/

**public** **class** StossPoint{…}

cái class này được sử dụng để luu lại tất cả các points của Stoßwelle vào 1 list. Sau đó mình vẽ đồ thị của Stoßewelle thì sẽ vẽ ra tất cả các Points chứ ko chỉ vẽ 1 điểm. 1 điểm Stoßpoint sẽ được lưu = tên máy đo, time và vận tốc của Stoßwelle giữa 2 máy.

815: // funtion to check if element in list ; return object if exist or null if not

**public** StossPoint check\_in\_list(String site\_to\_check, String tsp\_to\_check, List<StossPoint> stPoints){

cái này để check xem 1 point có thuộc Stoßwelle-List ko. Nếu có thì vẽ lên đồ thị.

826: // Task import InduktivSchleife from mst

**class** import\_from\_mst\_Task **extends** SwingWorker<Void, Void>{

cái sẽ sql tất cả các máy trong bảng mst và load vào Source-induktiv-Schleife

883: // Task import InduktivSchleife from mst

**class** import\_from\_mdp\_Task **extends** SwingWorker<Void, Void>{

cái sẽ sql tất cả các máy trong bảng mdp và load vào Source-induktiv-Schleife

935: // Task import InduktivSchleife from file

**class** import\_from\_file\_Task **extends** SwingWorker<Void, Void>{

cái này đọc tất cả cái máy đo từ 1 file csv và load vào Source-induktiv-Schleife

986: // Task export InduktivSchleife to file

**class** export\_induktiv\_Task **extends** SwingWorker<Void, Void>{

sql tất cả các máy đo trong bảng mdp rồi ghi ra file

1051: // Task export Density to file

**class** export\_density\_Task **extends** SwingWorker<Void, Void>{

tính toán density cho 1 list các máy đo rồi ghi ra file csv như sau:

Dòng đầu tiên là ghi tất cả các may đo

Dòng thứ 2 là ghi Fahrstreifen

Dòng thứ 3 là ghi tổng cộng có bao nhiêu dữ liệu

Từ dong thứ 4 trở đi là ghi theo từng thời điểm:

**site+";"+tsp+";"+density\_lkw+";"+density\_pkw+";"+density\_all**

1159: // Task to show all density from file to table (without set Sprung)

**class** show\_density\_on\_table\_without\_set\_sprung **extends** SwingWorker<Void, Void>{

Đọc dữ liệu từ csv file và hiện tất cả ra bảng

1262: // Task show show density on chart with set Sprung

**class** show\_density\_on\_chart\_with\_set\_sprung **extends** SwingWorker<Void, Void>{

Đọc dữ liệu từ csv file và hiện ra đồ thị với điều kiện Density Sprung lớn hơn 1 giá trị đầu vào

1676: // Task to show all density on chart (without set Sprung)

**class** show\_density\_on\_chart\_without\_set\_sprung **extends** SwingWorker<Void, Void>{

Đọc dữ liệu từ csv file và hiện tất density ra đồ thị

2087: // Task to show Stoßwelle on chart

**class** show\_stosswellen\_on\_chart **extends** SwingWorker<Void, Void>{

Đọc dữ liệu từ csv file , Tính Stoßwelle va hiện ra trên đồ thị.

Đọc theo từng điểm(từng dòng của csv file-bỏ đi 3 dòng đầu): rồi tính density của điểm kế tiếp qua funtion next\_sprung() trong khoảng 2min.-10min.\*khoảng cách giữa 2 máy đo (km).

Nếu thỏa mãn 3 lần liên tiếp thì sẽ tính vận tốc rồi vẽ trên đồ thị.

Cái này tính toán lâu là mỗi lần tính cho 1 điểm thì phai sql vào DB để tính. Mà có rất nhiều điểm => chỉ nên giới hạn khoảng 20 máy đo trong vong 1 tháng trở lại.