**Rules:** Rules are predefined constraints and guidelines which help in PCB design

**BoardRules.java:** This file contains functions that help provide features in the rules drop down menu like adding, removing, and modifying values of Vias, nets, net classes, or clearance matrix.

This file contains functions like:

* **get\_trace\_half\_width()** which returns the trace halfwidth used for routing with the input net on the input layer.
* **set\_default\_trace\_half\_width()** which Changes the default trace halfwidth used for routing on all layers to the input value.
* **get\_default\_net\_class()** which returns the default net rule for which no special rule was set.
* **create\_default\_via\_rule()** which helps Create a default via rule for p\_net\_class with name p\_name. If more than one via infos with the same layer range are found, only the via info with the smallest pad size is inserted.
* **change\_clearance\_class\_no()** which Changes the clearance class index of all objects on the board
* **remove\_clearance\_class()** which removes a clearance class provided an index.
* **get\_igonre\_conducion()** which if set true makes router ignore conduction areas.

**ClearanceMatrix.java:** This file addresses the features in the clearance matrix with help of functions.

* **remove\_class():** Removes the class with index p\_index from the clearance matrix.
* **append\_class():** Appends a new clearance class to the clearance matrix and initializes it with the values of the default class. Returns false, if a clearance class with name p\_class\_name is already existing.
* **clearance\_compensation\_value():** Returns the clearance compensation value of p\_clearance\_class\_no on layer p\_layer.
* **get\_layer\_count():** Return the layer count of this clearance matrix

It consists of classes like:

* **Row:** contains a row of entries of the clearance matrix
* **MatrixEntry:** this is equivalent to a single entry of the clearance matrix

**DefaultItemClearanceClasses.java:**

A clearance class is a name attached with a set of specific rules(clearance) for the PCB.

The file DefaultItemClearanceClasses.java enables functions for the inner working of clearance matrix.

* **get():** Returns the number of the default clearance class for the input item class.
* **set():** Sets the index of the default clearance class of the input item class in the clearance matrix to p\_index.
* **set\_all():** Sets the indices of all default item clearance classes to p\_index.

**Net.java:**

This file describes properties for an electrical net.

* **get\_terminal\_items():** Returns the pins and conduction areas of this net.
* **get\_pins():** Returns the pins of this net.
* **get\_items():** Returns all items of this net.
* **get\_trace\_length():** Returns the cumulative trace length of all traces on the board belonging to this net.
* **get\_via\_count():** Returns the count of vias on the board belonging to this net.
* **contains\_plane():** Indicates, if this net contains a power plane. Used by the autorouter for setting the via costs to the cheap plane via costs. May also be true, if a layer covered with a conduction\_area of this net is a signal layer.

**NetClass.java:**

This file describes routing rules for the individual nets.

* **trace\_width\_is\_layer\_dependent():** Returns true, if the trace width of this class is not equal on all layers.
* **trace\_width\_is\_inner\_layer\_dependent():** Returns true, if the trace width of this class is not equal on all inner layers.
* **set\_all\_inner\_layers\_active():** Activates or deactivates all inner layers for routing
* **is\_active\_routing-layer():** Returns if the layer with index p\_layer\_no is active for routing

**NetClasses.java:**

Contains an array of Net Classes for routing stores in a vector.

**Nets.java:**

Describes the electrical nets on the board. Contains an array of Nets in the form of a vector.

* **max\_net\_no():** Returns the biggest net number on the board.
* **get():** Used to return net with help of a value like input net number or name.
* **new\_net():** Creates a new number, and adds it to the end of the nets array.
* **add():** Adds a new net with default properties with the input name.
* **get\_board():** Gets the board of this net list. Used to get access to the items of the net.
* **set\_board():** Sets the board of this net list. Used to get access to the items of the net.

**ViaInfo.java:**

Information about a combination of via\_padstack, via clearance class and drill\_to\_smd\_allowed used in interactive and auto-routing.

* **get\_name():** Gets name of the Via selected.
* **set\_name():** Sets name of the Via selected to the input value.
* **attach\_smd\_allowed():** checks if it allows attachment with smd.
* **set\_attach\_smd\_allowed():** sets attachment with smd.
* **compareTo():** compares two vias.

**ViaInfos.java:**

Contains a list of different ViaInfo, which can be used in interactive and automatic routing.

* **add():** adds a new Via Info to the list.
* **count():** Counts total number of vias in the list.
* **get():** Returns Via Info provided the name or number.
* **name\_exists():** Checks if a via with the given input name exists.
* **remove():** Removes a ViaInfo(given) from the list of Via Infos.

**ViaRule.java:**

Contains an array of vias used for routing. Vias at the beginning of the array are preferred to later vias.

* **remove\_via():** Removes via from the rule, returns false if via is not in the rule.
* **contains():** Checks if given viaInfo in the ViaInfos list is in the ViaRule.
* **contains\_padstack():** Returns true, if this via contains a padstack(given).
* **get\_layer\_range():** Searches for Via in the ViaRule from a one given layer to another given layer.
* **swap():** Swaps ViaInfo of 2 given ViaInfos, returns false if they were not found.