JAVA Topology API Documentation

Topology Topologies.readJSON(filename)

Loads a .json file into a Java (Topology) object

Parameters:

- fileName: String

  represents the name of the .json file from where a topology is to be loaded

Returns:

- Topology:

  object containing the topology ID and a list of component objects

void Topologies.writeJSON(topID, filename)

Saves (Topology) object into a .json file

Parameters:

- topID: String

  represents the ID of the topology in memory that is to be saved in a .json file

- fileName: String
  represents the name of the .json file in which the topology is to be saved

Returns:

- None

List<Topology> Topologies.queryTopologies()

Returns a list of (Topology) objects stored in the memory

Parameters:

- None

Returns:

- List<Topology>
  Java List of (Topology) objects

List<Component> Topologies.deleteTopology(topID)

Deleted a certain topology from the memory

Parameters:

- topID: String
  represents the ID of the topology to be deleted

Returns:

- None

List<Component> Topologies.queryDevices(topID)

Returns a list of (Component) objects of a specified topology

Parameters:

- topID: String
  represents the ID of the topology whose devices are queried

Returns:

- List<Component>
  Java List of (Component) objects of the specified topology

List<Component> Topologies. queryDevicesWithNetlistNode (topID , NetlistNodeID)

Returns a list of (Component) objects of components of a topology that are connected to a specific Netlist Node

Parameters:

- topID: String
  represents the ID of the topology whose devices are queried
- NetlistNodeID: String

  Represents the ID of the node that is checked for the components connected to it

Returns:

- List<Component>
  Java List of (Component) objects of components the specified topology that are connected to the specified netlist node

Comments

- I chose to implement the requirements using Java since I felt I could accomplish all the requirements best through it. It has many useful dependencies, provides automatic garbage collection, and runs in a managed environment so all of its pointers are smart pointers. It is also the language I am most familiar with out of the available options.

- I split the ResistorMeasure and NmosMeasure into 2 classes based on the assumption that one of has int attributes and the other has float attributes, but they can be combined together in the case that this is not required.

Thank You