**Algorithm: ScenarioComposer**

Class 🡨 ScenarioComposer

Function 🡨 \_\_init\_\_(self):

self.conn = **Database.getConnection()**

self.cursor 🡨 self.conn.cursor()

Function 🡨 getInstructors(self):

Self.cursor.execute(‘SELECT id, name, hours, schedule FROM instructors WHERE active = 1')

instructors 🡨 self.listToDictionary(self.cursor.fetchall())

instructors 🡨 self.jsonToList(instructors,2)

Return 🡨 instructors

Function 🡨 getRooms(self):

self.cursor.execute('SELECT id, name, type, schedule FROM rooms WHERE active = 1')

rooms 🡨 self.listToDictionary(self.cursor.fetchall())

rooms 🡨 self.jsonToList(rooms,2)

Return 🡨 rooms

Function 🡨 getSubjects(self):

self.cursor.execute('SELECT id, name, hours, code, description, instructors, divisible, type FROM subjects')

subjects 🡨 self.listToDictionary(self.cursor.fetchall())

subjects 🡨 self.jsonToList(subjects, 4)

subjects 🡨 self.stringToInt(subjects, 4)

Return 🡨 subjects

Function 🡨 getSections(self):

self.cursor.execute('SELECT id, name, schedule, subjects, stay FROM sections WHERE active = 1')

sections 🡨 self.listToDictionary(self.cursor.fetchall())

sections 🡨 self.jsonToList(sections, 1)

sections 🡨 self.jsonToList(sections, 2)

sections 🡨 self.stringToInt(sections, 2)

Return 🡨 sections

Function 🡨 getSharings(self):

self.cursor.execute('SELECT id, subjectId, sections FROM sharings WHERE final = 1')

sharings 🡨 self.listToDictionary(self.cursor.fetchall())

sharings 🡨 self.jsonToList(sharings, 1)

sharings 🡨 self.stringToInt(sharings, 1)

Return 🡨 sharings

Function 🡨 listToDictionary(self, toDict):

Return 🡨 {entry[0]: list(entry[1:]) for entry in toDict}

Function 🡨 jsonToList(self, dictionary, index):

FOR key, value in dictionary.items():

dictionary[key][index] 🡨 json.loads(value[index])

END FOR Loop

Return 🡨 dictionary

Function 🡨 stringToInt(self, dictionary, index):

FOR key, value in dictionary.items():

dictionary[key][index] 🡨 list(map(int, value[index]))

END FOR Loop

Return 🡨 dictionary

Function 🡨 closeConnection(self):

self.conn.commit()

self.conn.close()

Function 🡨 getScenarioData(self):

data = {

'instructors': self.getInstructors(),

'sharings': self.getSharings(),

'sections': self.getSections(),

'subjects': self.getSubjects(),

'rooms': self.getRooms()

}

**self.closeConnection()**

Return 🡨 data