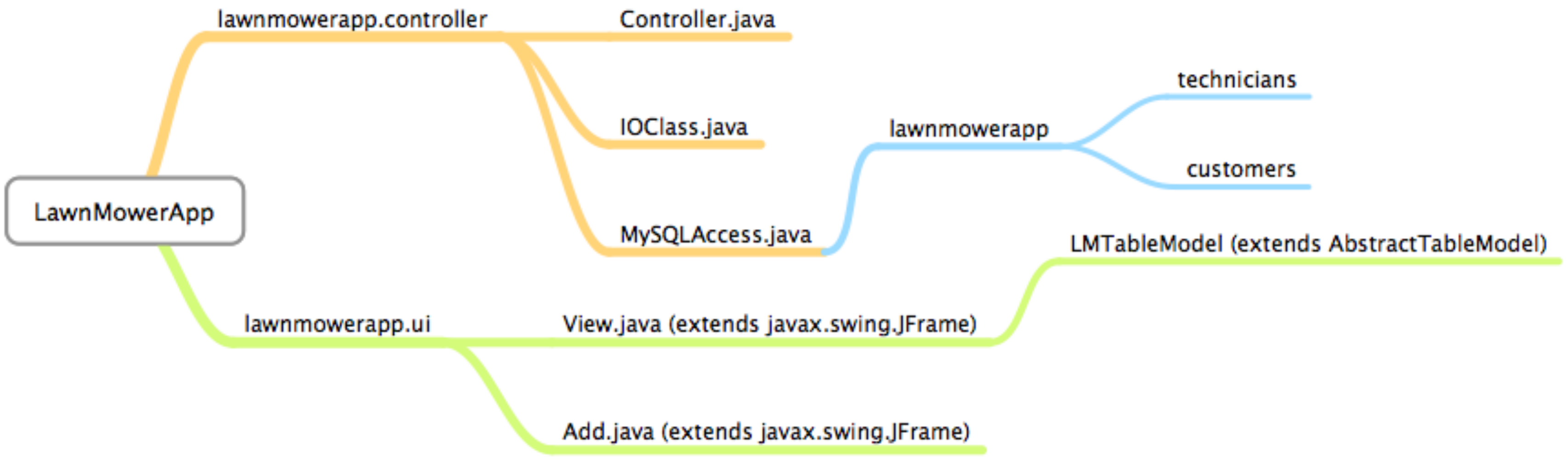
Structure and Organization



## Structure and Organization

lawnmowerapp.controller

- Controller.java
  - Handles calls to and from View to MySQLAccess
  - Modifies sql results into object arrays for the View
  - Functional methods (calculating date)

- IOClass.java
  - Prepares data when creating txt files for bills and task list

- MySQLAccess.java
  - Handles all direct calls to the database
  - Contains error detection when appropriate (bad requests)

## Structure and Organization

lawnmowerapp.ui

- View.java
  - Contains main method to run program
  - Provides simple UI for manipulation
- LMTableModel (inner class)
  - Table model to maintain technicians and customers
  - Contains logic to update both database and view
- Add.java
  - Create new technician or new customer

## Structure and Organization

Tables

<u>Customers</u>

| | |
|---|---|
| id | int |
| last_name | varchar |
| first_name | varchar |
| address | varchar |
| ori_signup | date |
| service_date | date |
| amount_owes | double |
| technician_id | int |
| completed | bit |
| paid | bit |

<u>Technicians</u>

| | |
|---|---|
| id | int |
| last_name | varchar |
| first_name | varchar |
| num_of_jobs | int |

## Technologies Used

Database
- MySQL
- MySQL Workbench

Database Connection
- MySQL Connector 5.1 (JDBC driver)

GUI and Functionality
- Java
- Java Swing

IDE
- NetBeans 8.0

## Thought Process

First time really creating and designing a database for an app

Used class based separation, each class has a specific purpose

Since using Swing, loosely implemented MVC

“Get it working, then make it better”

## What I Would’ve Done Differently

Better security for database calls

Better error handling for bad sql requests (return false)

Consistency throughout (parameters, order)

Handling data (Object[] vs. What You Need vs. Classes)

Deletion and better UI controls