Stephen Holman

Kansas City Energy Report

**How to Set Up and Run This Program:**

1. Unzip the file and place the contents in a separate folder.
2. In MySQL, create a schema named “kc\_energy” and use the SQL text files “calculate\_meter\_balance”, “calculate\_customer\_balance”, and “create\_charge” (preferably in that order) to create stored procedures by those names in the schema.
3. In your Java IDE, make sure that all Java classes are all in the same Java project, package folder, and package. The simplest ways to do this are to set the KC\_Energy folder as the Java project folder or to create a new Java project and have it use the “src” folder included in the ZIP file.
4. Add the included JAR files, “itextpdf-5.5.13.1” and “mysql-connector-j-8.0.32”, to the build path of the Java project.
5. Compile all of the Java class files, then run GUI\_Window.java.
6. To use the mock data created for this project, run the SQL script “kc\_energy\_setup” to create the necessary tables in the kc\_energy schema (if GUI\_Window.java has never been run before) and use the “calculate\_all\_customer\_balances” file to create a stored procedure, just like in step 2. Then execute the “customer\_mock\_data”, “meter\_mock\_data”, and “charge\_mock\_data” SQL scripts IN THAT EXACT ORDER. This should add all the mock data to the schema and accurately calculate the balances for all customers, meters, and charges.

**Link to Presentation:**

<https://umsystem.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=994ffd3c-f7ba-4bdf-8cfe-affa0076bae5>

**Class Diagrams:**

Diagram

Description automatically generated

**Feature List:**

|  |  |  |
| --- | --- | --- |
| **Feature** | **Implemented (Full/Partial)** | **Notes** |
| Add New Customer | Full |  |
| Edit Customer Settings | Full |  |
| Delete Customer | Full | Deletion cascades; all meters for customer are also destroyed |
| Search Customer List | Full | Case sensitive |
| See Customer Details | Full |  |
| Print Customer Info to PDF | Full | Currently saves PDFs to Java project folder |
| Add Meter to Customer | Full |  |
| Edit Meter Settings | Full |  |
| Delete Meter from Customer | Full | Deletion cascades; all charge records for meter destroyed |
| See Meter Details | Full |  |
| Add Charge to Meter | Full | Charge amount is calculated upon creation with current tariff |
| Set Charge to Paid/Unpaid | Full | Most details should not be editable by design |
| Delete Charge | Full |  |

**Explanations:**

kc\_energy\_setup.sql: The specifications for the SQL tables in the MySQL database. Every entry into the customer, meter, or charge table is automatically assigned an auto-incrementing, unique ID number. For the purposes of this program and document, a customer’s “ID” and “account number” may be used interchangeably. Every meter has a foreign key that links it to the ID number of a customer, and every charge has a foreign key that links it to the ID number of a meter. The parameters of each Decimal attribute in the tables indicate the max values that these attributes should handle; for example, the OutstandingBalance attribute in the customer table can handle numbers less than 10,000,000,000 with two decimal places. For entered values, the range of acceptable values is enforced on the Java side of the program, so numbers greater than 1000 cannot be entered into the Tariff value of a meter, for example. For calculated values, the range of values is big enough to not overflow in any reasonable context.

GUI\_Window.java: The class that holds the main function of the program and that is an extension of JFrame. The constructor creates a connection to the MySQL server with the information in the Strings “dbURL”, “username”, and “password”. This connection is used by all the other classes to query the database. Then, it checks to see if the necessary tables exist in the server and creates them if not. This means that either GUI\_Window.java or kc\_energy\_setup.sql can be used to create the tables. A GUI\_Window object has a set size (defined by the constants WINDOW\_WIDTH and WINDOW\_HEIGHT) and has its content replaced by instances of the other Java classes to interact with the user and the database. Every time the content is replaced, the resetContent() method must be called to briefly resize the window and make sure the content is displayed properly.

Customer\_Window.java: The first page that gets loaded into GUI\_Window that displays basic customer information from the database in a table that the user can select from. In the constructor, this class sets up the GUI components and fills the JTable with information from the Customer MySQL table using Object array sqlData as a median. If there is a search criterion included in the constructor, the query to get the customer data includes a LIKE section; else, the query just gets all information from the table. In either case, the results are ordered by AccountNumber, organized in a JTable, and displayed in a JScrollPane. If the user uses the Search button, it will load a new Customer\_Window, with the current contents of searchField as a parameter, into the GUI\_Window content pane. If the user decides to add a customer, the button will load an Edit\_Customer\_Window container with no associated customer ID into the GUI\_Window. If the user decides to delete a customer, a confirmation window will appear; if the user confirms, then the customer will be deleted and a new Customer\_Window with no search parameters will be loaded into GUI\_Window. Deleting customers will cascade through the database, so all meters and charges associated with this customer will also be deleted automatically. If the user chooses to view a customer, a new Meter\_Window with the appropriate account number as a parameter will be loaded into GUI\_Window.

Edit\_Customer\_Window.java: A page that allows the user to manipulate the information of a customer. If the constructor is called with accNum equal to -1, then the user is creating a new customer and all the fields are initially blank. If accNum is equal to anything else, then the user is modifying the customer with that account number and all the fields are pre-filled with that customer’s information from the database. The user must enter valid information in all relevant fields to confirm their submissions; upon clicking “Confirm”, all the entries are checked for errors and any mistakes found are reported via a message box. If the current country selected from the list is the United States, then the ZIP code and state components become active and must be filled; else, they are disabled and send “null” for those data fields in the database. If the “Confirm” button is pressed and no errors are found, then the data is either inserted or updated to the customer table in the database and the GUI\_Window has its content set back to the previous page. If the “Back” button is pressed, no updates are made to the database and the GUI\_Window has its content set back to the previous page.

Meter\_Window.java: A page that shows the user all the meters associated with a customer as well as the customer’s personal information. The accNum parameter in the constructor contains the ID of the customer being queried. It queries the customer table for all the information about the customer from the customer table, then queries the meter table for all meters that have the same custID as the accNum parameter. The ResultSet from the latter query is converted into an Object array sqlData and then displayed in a table. The user can edit the settings of the customer, setting GUI\_Window’s content to an Edit\_Customer\_Window object. The user can also add a meter, setting GUI\_Window’s content to an Edit\_Meter\_Window object. If the user decides to delete a meter, a confirmation window will appear; if the user confirms, then the meter will be deleted and a new Meter\_Window for this customer will be loaded into GUI\_Window. Deletion instead requires the system to recalculate the total balance of the customer using the stored procedure “calculate\_customer\_balance”. If the user selects the “Print Bill to PDF” button, the Meter\_Window object will create a PDF detailing the user information, meters, and charges of the associated customer and save it to the KC\_Energy folder. The new PDF’s name will have the customer’s ID number and the creation date in the title. The user can select a meter from the table and view it, setting the GUI\_Window’s content to a Charge\_Window with the appropriate MeterID passed as a parameter. Selecting the “Back” button will set GUI\_Window’s content to a new Customer\_Window with no search parameters.

Edit\_Meter\_Window.java: A page that allows the user to manipulate the energy type and energy tariff of a meter. If the constructor is called with inputMeter set to -1, then the user is creating a new meter for the customer with the account number equal to inputCID. If not, then the user is updating the meter with meterID equal to inputMeter. The user can set the energy type to one of the generic types in the combination box or enter their own as long as it fits the size limit (100 characters) and contains no backslashes. The tariff for the meter must be a positive number below $1000 per unit of energy and is recorded as a BigDecimal value to ensure precision is maintained. If this page is meant to update an existing meter, it should be noted that the new tariff will only affect new charges on the meter; any pre-existing charges will keep their original tariff rates and total costs. The customer the meter is associated with is determined by how the user got to the Edit\_Meter\_Window page and is passed by inputCID. If the user clicks the “Confirm” button, the page will check if the entered information is valid. If it is, then the page will send the information to the database and set GUI\_Window’s content to the previous page. If not, it will display a message box detailing the errors that occurred. If the “Back” button is pressed, no updates are made to the database and the GUI\_Window has its content set back to the previous page.

Charge\_Window.java: A page that shows the user all the charges associated with a meter as well as the total usage and balance of the meter. The metNum parameter in the constructor contains the ID of the meter being queried. It queries the meter table for all the information about the meter from the meter table, makes another small query to find the name of the customer whom the meter belongs to, and then queries the charge table for all charges that have the same metID as the metNum parameter. The ResultSet from the last query is converted into an Object array sqlData and then displayed in a table. The user can edit the energy type and tariff of the meter with the “Edit Meter” button, setting GUI\_Window’s content to an Edit\_Meter\_Window object. The user can add a charge to this meter, setting GUI\_Window’s content to an Add\_Charge\_Window object. If the user decides to delete a charge, a confirmation window will appear; if the user confirms, then the charge will be deleted and a new Charge\_Window for the same meter will be loaded into GUI\_Window. The user can also select a charge from the table and set it to “Paid” or “Unpaid”, which will create a new Charge\_Window in GUI\_Window as well. Adding, setting, or deleting a charge automatically recalculates the energy use and balance of the meter and the customer attached to it. Selecting the “Back” button will set GUI\_Window’s content to a new Meter\_Window for the customer attached to the meter.

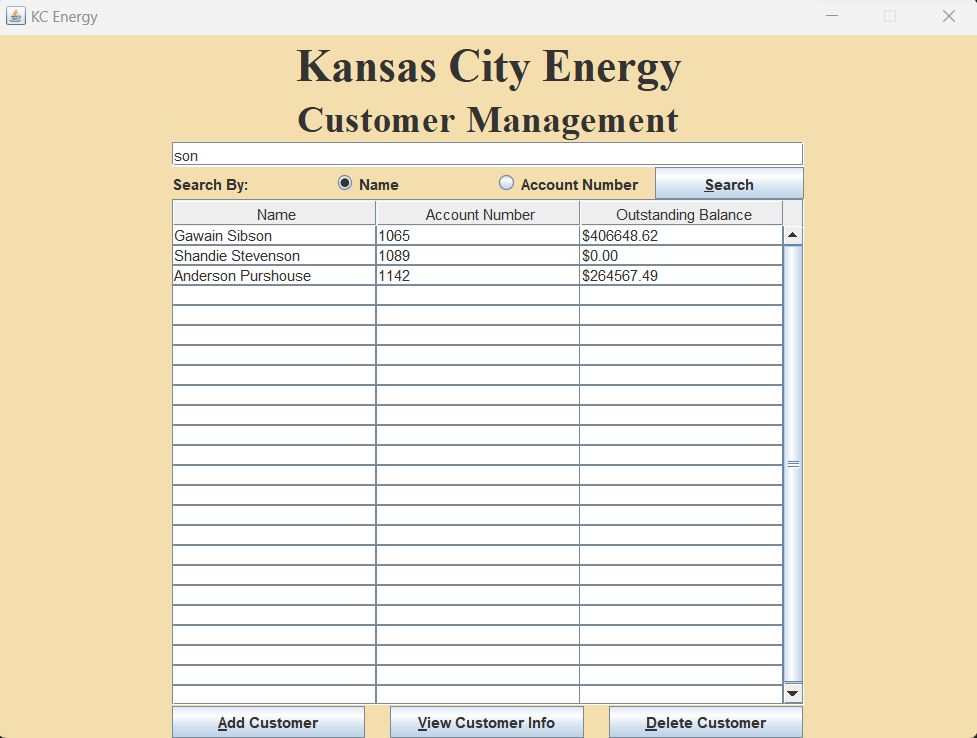
Add\_Charge\_Window.java: A page that allows the user to add a new charge to a meter. The constructor must always have a valid meter number for the charge to be associated with, passed with the inputMeter parameter. The user must set the charge amount to a value between $0.01 and $999.99 dollars, which is stored in a BigDecimal value to ensure precision is maintained. The user can use the check box to specify whether the new charge has already been paid or not. If the user clicks “Confirm”, the page will check the amount entered into the text field. If it is valid, a new charge with that amount is added to the meter, the balances of the customer and the meter are recalculated, and the GUI\_Window’s content is set to a new Charge\_Window for the appropriate meter. If it is not valid, a message box appears asking for a valid input. Selecting the “Back” button will set GUI\_Window’s content back to a new Charge\_Window for the meter without making an update to the database.

**Screenshots:**

The “homepage” or Customer Viewing Page that displays all the customer’s in a table and lets the user select a customer to examine, add or delete customers, or search for customers based on the contents of the search bar near the top of the screen.

Table

Description automatically generated

The same page after making a search. Now the table only displays the customers that have the phrase “son” in their names. 

The page used to create new customers. The program will determine if the entered information is acceptable and will tell the user what needs to be fixed if anything is wrong; if not, the new customer is entered into the database and the program returns to the Customer Viewing Page.

Chart, bar chart

Description automatically generated

The Meter Viewing Screen that displays all the meters associated with a customer in the database. The user can now edit the settings of the selected customer, add or delete meters from this customer, go back to the Customer Viewing Screen, create a bill detailing all of the meters and charges for this customer, or select a meter.

Table

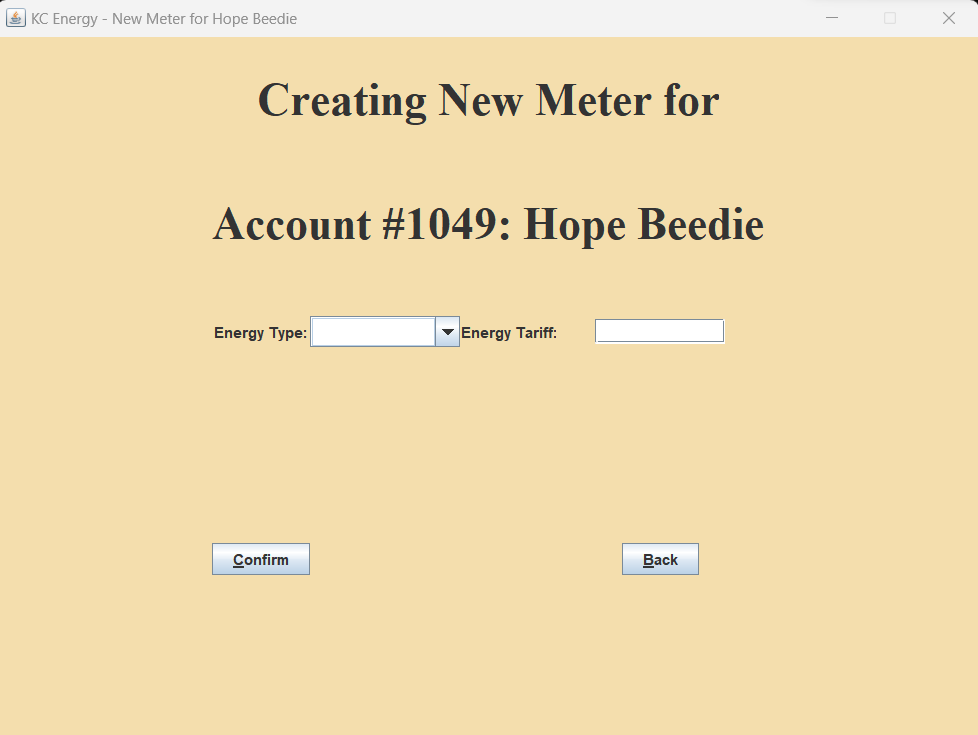
Description automatically generated

The page used to edit a customer’s settings. All the boxes are pre-filled with what is already in the database, and any new inputs must pass the same standards that new customer information has.

Chart

Description automatically generated

The page for adding a new meter to a customer. The user can select an energy type from the dropdown list or enter their own, as well as a tariff rate for the meter. These things can be changed later.



The Charge Viewing Page displaying the charges linked to a particular meter. From here the user can edit the properties of the meter, add or delete charges, set a charge to “Paid” or “Unpaid”, or return to the Meter Viewing Page. Adding charges or changing the payment status of a charge will automatically recalculate the energy usage and balance of both the customer and the meter.

Table

Description automatically generated

A page for editing the properties of a meter. A newly entered tariff rate will ONLY affect new charges on this meter; any pre-existing charges will retain the tariff rate and cost at the time of their creation.

Graphical user interface, application

Description automatically generated

The page for creating a new charge on a meter. As the user enters the charge amount, the label below it will automatically update to reflect the final cost of the charge based on the current tariff.

Graphical user interface, application, Word

Description automatically generated