```java

package application;

import javafx.beans.property.SimpleBooleanProperty;

public class HistoryScreen {
    private User<HealthData<?>> user;
    TableView<HealthData<?>> tableView;
    ObservableList<HealthData<?>> data;

    /**
     * Constructs a new HistoryScreen object.
     * @param user the User object for which to display the health data history
     * Pre: HealthdataEntry scene is working properly
     * PRECONDITION: user is not null
     */
    public HistoryScreen(User<HealthData<?>> user) {
        this.user = user;
        data = FXCollections.observableArrayList(user.getHealthDataList());
    }
    /**
     * Displays the health data history screen.
     * Pre: History button in the healthdataentry exists and the showhistoryscreen is functioning
     * POSTCONDITION: The health data history screen is displayed on a new Stage.
     */
    public void display() {
        Stage stage = new Stage();
        stage.setTitle("Health Data History");

        // Create columns for the table
        TableColumn<HealthData<?>, String> nameColumn = new TableColumn<>("Name");
        nameColumn.setCellValueFactory(param -> new SimpleStringProperty(param.getValue().getName()));

        TableColumn<HealthData<?>, String> dateColumn = new TableColumn<>("Date");
        dateColumn.setCellValueFactory(param -> new SimpleStringProperty(param.getValue().getDate().toString()));

        TableColumn<HealthData<?>, String> metricColumn = new TableColumn<>("Metric");
        metricColumn.setCellValueFactory(param -> new SimpleStringProperty(getMetricValue(param.getValue())));

        metricColumn.setCellFactory(column -> {
            return new TableCell<HealthData<?>, String>() {
                @Override
                protected void updateItem(String item, boolean empty) {
                    super.updateItem(item, empty);

                    if (empty || item == null) {
                        setText(null);
                    } else {
                        setText(item);
                    }
                }
            };
        });

        TableColumn<HealthData<?>, Boolean> editColumn = new TableColumn<>("Edit");
        editColumn.setCellValueFactory(param -> new SimpleBooleanProperty(true));
        editColumn.setCellFactory(param -> new TableCell<HealthData<?>, Boolean>() {
            private final Button editButton = new Button("Edit");

            {
                editButton.setOnAction(event -> {
                    HealthData<?> healthData = getTableView().getItems().get(getIndex());
                    Stage editStage = new Stage();
                    editStage.setTitle("Edit Health Data");

                    HealthDataEntry entryScreen = new HealthDataEntry(editStage, user);
//                    entryScreen.setCurrentHealthData(healthData); // Set the current health data

                    if (healthData instanceof CommonHealthData) {
                        CommonHealthData commonHealthData = (CommonHealthData) healthData;
                        String metric = commonHealthData.getMetric();

                        if (metric.equals("Blood Pressure")) {
                            Scene scene = entryScreen.showBloodPressureScene(commonHealthData, true, tableView);
                            editStage.setScene(scene);
                        } else if (metric.equals("Blood Glucose")) {
                            Scene scene = entryScreen.showBloodSugarScene(commonHealthData, true, tableView); // Pass the existing health data and the table view
                            editStage.setScene(scene);
                        } else if (metric.equals("BMI")) {
                            Scene scene = entryScreen.showBMIScene(commonHealthData, true, tableView);
                            editStage.setScene(scene);
                        } else if (metric.equals("Cholesterol")) {
                            Scene scene = entryScreen.showCholesterolScene(commonHealthData, true, tableView);
                            editStage.setScene(scene);
                        }
                    } else if (healthData instanceof CustomHealthData) {
                        CustomHealthData customHealthData = (CustomHealthData) healthData;

                        Scene scene = entryScreen.showCustomHealthNoteScene(customHealthData, true, tableView);
                        editStage.setScene(scene);
                    }

                    // Show the edit stage after setting the scene
                    editStage.show();
                });
            }

            @Override
            protected void updateItem(Boolean item, boolean empty) {
                super.updateItem(item, empty);

                if (empty) {
                    setGraphic(null);
                } else {
                    setGraphic(editButton);
                }
            }
        });

        // Create the table view
        tableView = new TableView<>();
        tableView.getColumns().addAll(nameColumn, dateColumn, metricColumn, editColumn);
        tableView.setItems(data);

        // Create a date picker for filtering
        DatePicker datePicker = new DatePicker();
        datePicker.setOnAction(event -> filterDataByDate(datePicker.getValue()));

        // Create a button to clear the filter
        Button clearFilterButton = new Button("Clear Filter");
        clearFilterButton.setOnAction(event -> {
            datePicker.setValue(null);
            data.setAll(user.getHealthDataList());
            calculateAverageMetrics(user.getHealthDataList());
        });

        // Create a label for displaying average metrics
        Label averageMetricsLabel = new Label();
        averageMetricsLabel.setId("averageMetricsLabel"); // Set an ID for the label

        // Create a back button
        Button backButton = new Button("Back");
        backButton.setOnAction(event -> {
            stage.close();
        });

        // Create a layout container
        VBox root = new VBox(datePicker, clearFilterButton, tableView, averageMetricsLabel, backButton);
        root.setSpacing(10);
        root.setPadding(new Insets(10));
        Scene scene = new Scene(root);
        stage.setScene(scene);
        stage.show();

        // Calculate and display the average metrics for all data
        calculateAverageMetrics(user.getHealthDataList());

    }
    /**
     * Filters the health data entries based on the selected date.
     * @param selectedDate the selected date for filtering
     * PRECONDITION: selectedDate is a valid LocalDate object or null
     * POSTCONDITION: The health data entries in the TableView are filtered based on the selected date.
     */
    private void filterDataByDate(LocalDate selectedDate) {
        if (selectedDate != null) {
            List<HealthData<?>> filteredData = user.getHealthDataList().stream()
                    .filter(data -> data.getDate().toInstant().atZone(ZoneId.systemDefault()).toLocalDate().equals(selectedDate))
```

```java
                    .collect(Collectors.toList());
            data.setAll(filteredData);
            calculateAverageMetrics(filteredData);
        } else {
            data.setAll(user.getHealthDataList());
            calculateAverageMetrics(user.getHealthDataList());
        }
    }
    /**
     * Calculates and displays the average metrics for the given health data list.
     * @param healthDataList the list of health data entries
     * PRECONDITION: healthDataList is not null
     * POSTCONDITION: The average metrics are calculated and displayed in the 
averageMetricsLabel.
     */
    void calculateAverageMetrics(List<HealthData<?>> healthDataList) {
        double totalBMI = 0;
        double totalLDL = 0;
        double totalHDL = 0;
        int totalSystolicBP = 0;
        int totalDiastolicBP = 0;

        double totalGlucoseLevel = 0;
        int bmiCount = 0;
        int ldlCount = 0;
        int hdlCount = 0;
        int systolicBPCount = 0;
        int glucoseLevelCount = 0;
        int diastolicBPCount = 0;


        for (HealthData<?> healthData : healthDataList) {
            if (healthData instanceof CommonHealthData) {
                CommonHealthData commonHealthData = (CommonHealthData) healthData;
                String metric = commonHealthData.getMetric();

                if (metric.equals("BMI")) {
                    totalBMI += commonHealthData.calculateBMI();
                    bmiCount++;
                } else if (metric.equals("Cholesterol")) {
                    totalLDL += commonHealthData.getLdlCholesterol();
                    ldlCount++;
                    totalHDL += commonHealthData.getHdlCholesterol();
                    hdlCount++;
                } else if (metric.equals("Blood Pressure")) {
                    totalSystolicBP += commonHealthData.getSystolicBP();
                    systolicBPCount++;
                    totalDiastolicBP += commonHealthData.getDiastolicBP();
                    diastolicBPCount++;
                } else if (metric.equals("Blood Glucose")) {
                    totalGlucoseLevel += commonHealthData.getGlucoseLevel();
                    glucoseLevelCount++;
                }
            }
        }

        StringBuilder averageMetrics = new StringBuilder();
        if (bmiCount > 0) {
            double averageBMI = totalBMI / bmiCount;
            averageMetrics.append("Average BMI: ").append(String.format("%.2f", averageBMI)).append("\n");
        }
        if (ldlCount > 0) {
            double averageLDL = totalLDL / ldlCount;
            averageMetrics.append("Average LDL: ").append(String.format("%.2f", averageLDL)).append("\n");
        }
        if (hdlCount > 0) {
            double averageHDL = totalHDL / hdlCount;
            averageMetrics.append("Average HDL: ").append(String.format("%.2f", averageHDL)).append("\n");
        }
        if (systolicBPCount > 0) {
            double averageDiastolicBP = totalDiastolicBP / systolicBPCount;
            averageMetrics.append("Average Systolic BP: ").append(String.format("%.2f", averageDiastolicBP)).append("\n");
        }
        if (diastolicBPCount > 0) {
            double averageSystolicBP = totalSystolicBP / diastolicBPCount;
            averageMetrics.append("Average Systolic BP: ").append(String.format("%.2f", averageSystolicBP)).append("\n");
        }
        if (glucoseLevelCount > 0) {
            double averageGlucoseLevel = totalGlucoseLevel / glucoseLevelCount;
            averageMetrics.append("Average Glucose Level: ").append(String.format("%.2f", averageGlucoseLevel)).append("\n");
        }

        Label averageMetricsLabel = (Label) tableView.getScene().lookup("#averageMetricsLabel");
        if (averageMetricsLabel != null) {
            averageMetricsLabel.setText(averageMetrics.toString());
        }
        averageMetrics.append("If you edited the values, average values will be refreshed if you go back and click 'history' tab again");
    }
    /**
     * Retrieves the metric value for a health data entry.
     * @param healthData the health data entry
     * @return the metric value as a string
     * PRECONDITION: healthData is not null
     * POSTCONDITION: The metric value for the health data entry is returned as a string.
     */
    private String getMetricValue(HealthData<?> healthData) {
        if (healthData instanceof CommonHealthData) {
            CommonHealthData commonHealthData = (CommonHealthData) healthData;
            String metric = commonHealthData.getMetric();

            if (metric.equals("Blood Pressure")) {
                int systolic = commonHealthData.getSystolicBP();
                int diastolic = commonHealthData.getDiastolicBP();
                return "Systolic: " + systolic + ", Diastolic: " + diastolic;
            } else if (metric.equals("Cholesterol")) {
                int ldl = commonHealthData.getLdlCholesterol();
                int hdl = commonHealthData.getHdlCholesterol();
                int triglycerides = commonHealthData.getTriglycerideCholesterol();
                return "LDL: " + ldl + ", HDL: " + hdl + ", Triglycerides: " +
triglycerides;
            } else if (metric.equals("BMI")) {
                double weight = commonHealthData.getWeight();
                double height = commonHealthData.getHeight();
                double bmi = commonHealthData.calculateBMI();
                return "Weight: " + weight + ", Height: " + height + ", BMI: " + bmi;
            } else if (metric.equals("Blood Glucose")) {
                double glucoseLevel = commonHealthData.getGlucoseLevel();
                return "Glucose Level: " + glucoseLevel;
            } else {
                return "";
            }
        } else if (healthData instanceof CustomHealthData) {
            CustomHealthData customHealthData = (CustomHealthData) healthData;
            return "Custom note: " + customHealthData.getData(); // Return the metric
value directly
        } else {
            return "";
        }
    }
}
```