======= Main.java =======

package dde;  
  
import javafx.application.Application;  
import javafx.geometry.Insets;  
import javafx.scene.Scene;  
import javafx.scene.chart.LineChart;  
import javafx.scene.chart.NumberAxis;  
import javafx.scene.chart.XYChart;  
import javafx.scene.control.\*;  
import javafx.scene.layout.\*;  
import javafx.stage.FileChooser;  
import javafx.stage.Stage;  
  
import java.io.File;  
import java.io.FileWriter;  
import java.io.IOException;  
import java.util.HashMap;  
import java.util.List;  
import java.util.Map;  
  
public class Main extends Application {  
  
 private final ComboBox<String> methodCombo = new ComboBox<>();  
 private final Map<String, XYChart.Series<Number, Number>> methodSeries = new HashMap<>();  
 private double[] ts;  
 private double[] ys;  
  
 @Override  
 public void start(Stage stage) {  
 System.*out*.println("JavaFX стартует!");  
  
  
 // Поля ввода параметров  
 TextField aField = new TextField("1");  
 TextField tauField = new TextField("1");  
 TextField hField = new TextField("0.1");  
 TextField tMaxField = new TextField("10");  
 TextField functionField = new TextField("-a \* y\_tau1");  
 TextField tau2Field = new TextField("0"); // по умолчанию отключено  
 TextField phiField = new TextField("1"); // по умолчанию: φ(t) = 1  
  
  
 //это дополнительный комбо бокс в котором будут зраниться примеры  
 ComboBox<String> exampleCombo = new ComboBox<>();  
 exampleCombo.getItems().addAll(  
 "Без примера",  
 "Классика: -a \* y\_tau1",  
 "Колебания: -a \* y\_tau1 + y",  
 "Система x'/y': x'= -a\*y\_tau1, y'=x\_tau1"  
 );  
 exampleCombo.setValue("Без примера");  
  
  
 // Сетка с подписями  
 GridPane inputGrid = new GridPane();  
 inputGrid.setVgap(10);  
 inputGrid.setHgap(10);  
 inputGrid.setPadding(new Insets(10));  
 inputGrid.add(new Label("a:"), 0, 0);  
 inputGrid.add(aField, 1, 0);  
 inputGrid.add(new Label("τ (tau):"), 0, 1);  
 inputGrid.add(tauField, 1, 1);  
 inputGrid.add(new Label("h (шаг):"), 0, 2);  
 inputGrid.add(hField, 1, 2);  
 inputGrid.add(new Label("t max:"), 0, 3);  
 inputGrid.add(tMaxField, 1, 3);  
 //новый лейбл  
 inputGrid.add(new Label("f(t, y, yτ):"), 0, 5);  
 inputGrid.add(functionField, 1, 5);  
 //новая часть с значением доп задержки  
 inputGrid.add(new Label("τ₂ (доп. задержка):"), 0, 6);  
 inputGrid.add(tau2Field, 1, 6);  
 //новая часть с любым фи которые мы хотим  
 inputGrid.add(new Label("φ(t ≤ 0):"), 0, 7);  
 inputGrid.add(phiField, 1, 7);  
 //тут бокс с примерами  
 inputGrid.add(new Label("Пример:"), 2, 7);  
 inputGrid.add(exampleCombo, 3, 7);  
  
  
 //это код для того чтобы появлялся выбор из бокса с примерами  
 exampleCombo.setOnAction(e -> {  
 switch (exampleCombo.getValue()) {  
 case "Классика: -a \* y\_tau1" -> {  
 functionField.setText("-a \* y\_tau1");  
 phiField.setText("1");  
 aField.setText("1");  
 tauField.setText("1");  
 tau2Field.setText("0");  
 }  
 case "Колебания: -a \* y\_tau1 + y" -> {  
 functionField.setText("-a \* y\_tau1 + y");  
 phiField.setText("1");  
 aField.setText("2");  
 tauField.setText("1");  
 tau2Field.setText("0");  
 }  
 case "Система x'/y': x'= -a\*y\_tau1, y'=x\_tau1" -> {  
 functionField.setText("-a \* y\_tau1"); // эта часть не используется напрямую — запускаем отдельную кнопку “Система DDE”  
 phiField.setText("1");  
 aField.setText("1");  
 tauField.setText("1");  
 tau2Field.setText("0");  
 }  
 }  
 });  
  
  
 // В метод start() после создания полей ввода:  
 methodCombo.getItems().addAll("Метод Эйлера", "Метод Рунге-Кутты 4", "Метод Адамса-Бэшфорта");  
 methodCombo.setValue("Метод Эйлера");  
 inputGrid.add(new Label("Метод:"), 0, 4);  
 inputGrid.add(methodCombo, 1, 4);  
  
 Button solveButton = new Button("Рассчитать");  
 Button saveButton = new Button("Сохранить в CSV");  
  
 // График  
 NumberAxis xAxis = new NumberAxis();  
 NumberAxis yAxis = new NumberAxis();  
  
  
 xAxis.setLabel("t");  
 yAxis.setLabel("y(t)");  
 LineChart<Number, Number> lineChart = new LineChart<>(xAxis, yAxis);  
 lineChart.setTitle("Решение DDE");  
  
  
  
  
  
  
  
 // Обработка кнопки "Рассчитать"  
 // Проверка параметров  
 solveButton.setOnAction(e -> {  
 double a = Double.*parseDouble*(aField.getText());  
 double tau = Double.*parseDouble*(tauField.getText());  
 double h = Double.*parseDouble*(hField.getText());  
 double tMax = Double.*parseDouble*(tMaxField.getText());  
 double tau2 = Double.*parseDouble*(tau2Field.getText());  
  
  
 if (h <= 0 || tMax <= 0 || tau < 0)  
 throw new IllegalArgumentException("Параметры должны быть > 0");  
  
 DDESolver solver = new DDESolver();  
 solver.setInitialFunction(t -> 1.0);  
  
 solver.setTau2(tau2); // передаём в решатель  
  
  
 solver.setExpression(functionField.getText());  
  
  
 DDESolver.Method method = switch (methodCombo.getValue()) {  
 case "Метод Рунге-Кутты 4" -> DDESolver.Method.*RUNGE\_KUTTA\_4*;  
 case "Метод Адамса-Бэшфорта" -> DDESolver.Method.*ADAMS\_BASHFORTH\_2*;  
 default -> DDESolver.Method.*EULER*;  
 };  
 String phiExpr = phiField.getText();  
 solver.setInitialFunctionExpression(phiExpr);  
  
 solver.solve(method, a, tau, h, tMax);  
  
 List<Double> tList = solver.getTList();  
 List<Double> yList = solver.getYList();  
  
 // Для сохранения  
 ts = tList.stream().mapToDouble(Double::doubleValue).toArray();  
 ys = yList.stream().mapToDouble(Double::doubleValue).toArray();  
  
 // Построение графика  
 XYChart.Series<Number, Number> series = new XYChart.Series<>();  
 series.setName(methodCombo.getValue());  
 for (int i = 0; i < tList.size(); i++) {  
 series.getData().add(new XYChart.Data<>(tList.get(i), yList.get(i)));  
 }  
  
 lineChart.getData().clear();  
 lineChart.getData().add(series);  
  
  
 System.*out*.println("Выбран метод: " + method);  
 });  
 //кнопка справки о методах  
 Button helpButton = new Button("ℹ️ Справка о методах");  
 helpButton.setTooltip(new Tooltip("Объяснение численных методов"));  
  
 //обработчик этой команды  
 helpButton.setOnAction(e -> {  
 Alert alert = new Alert(Alert.AlertType.*INFORMATION*);  
 alert.setTitle("Справка о численных методах");  
 alert.setHeaderText("Краткое описание методов:");  
 alert.setContentText("""  
 Метод Эйлера: простой и быстрый, но может быть неточным.  
 Метод Рунге–Кутты 4 порядка: самый точный среди реализованных.  
 Метод Адамса–Бэшфорта 2 порядка: использует предсказание на основе предыдущих шагов.  
   
 Вы можете сравнить их точность, используя кнопку 'Сравнить методы' или 'График ошибки'.  
 """);  
 alert.showAndWait();  
 });  
  
  
 //добавление кнопки сравнить методы  
 Button compareButton = new Button("Сравнить методы");  
  
 compareButton.setOnAction(e -> {  
 double a = Double.*parseDouble*(aField.getText());  
 double tau = Double.*parseDouble*(tauField.getText());  
 double h = Double.*parseDouble*(hField.getText());  
 double tMax = Double.*parseDouble*(tMaxField.getText());  
 double tau2 = Double.*parseDouble*(tau2Field.getText());  
 String phiExpr = phiField.getText();  
 String expression = functionField.getText();  
  
 lineChart.getData().clear();  
  
 for (DDESolver.Method method : DDESolver.Method.*values*()) {  
 DDESolver solver = new DDESolver();  
  
 solver.setTau2(tau2);  
 solver.setInitialFunctionExpression(phiExpr);  
 solver.setExpression(expression);  
  
 solver.solve(method, a, tau, h, tMax);  
  
 XYChart.Series<Number, Number> series = new XYChart.Series<>();  
 series.setName(switch (method) {  
 case *EULER* -> "Эйлер";  
 case *RUNGE\_KUTTA\_4* -> "Рунге-Кутта 4";  
 case *ADAMS\_BASHFORTH\_2* -> "Адамс-Бэшфорта";  
 });  
  
 List<Double> tList = solver.getTList();  
 List<Double> yList = solver.getYList();  
  
 for (int i = 0; i < tList.size(); i++) {  
 series.getData().add(new XYChart.Data<>(tList.get(i), yList.get(i)));  
 }  
  
 lineChart.getData().add(series);  
 }  
  
 lineChart.setTitle("Сравнение методов: a=" + a + ", τ=" + tau + ", h=" + h);  
 });  
  
  
 //Добавление кнопки экспорта всех методов  
 Button exportAllButton = new Button("Экспорт всех методов");  
  
 //Обработчик кнопки экспорат всех методов  
 exportAllButton.setOnAction(e -> {  
 double a = Double.*parseDouble*(aField.getText());  
 double tau = Double.*parseDouble*(tauField.getText());  
 double h = Double.*parseDouble*(hField.getText());  
 double tMax = Double.*parseDouble*(tMaxField.getText());  
  
 int steps = (int) (tMax / h);  
 double[] tsLocal = new double[steps + 1];  
 double[][] ysLocal = new double[3][steps + 1];  
  
 int methodIndex = 0;  
 for (DDESolver.Method method : DDESolver.Method.*values*()) {  
 DDESolver solver = new DDESolver();  
 solver.setInitialFunction(t -> 1.0);  
  
 String phiExpr = phiField.getText();  
 solver.setInitialFunctionExpression(phiExpr);  
  
 solver.solve(method, a, tau, h, tMax);  
  
 List<Double> tList = solver.getTList();  
 List<Double> yList = solver.getYList();  
  
 for (int i = 0; i < tList.size(); i++) {  
 tsLocal[i] = tList.get(i);  
 ysLocal[methodIndex][i] = yList.get(i);  
 }  
 methodIndex++;  
 }  
  
 // Сохраняем  
 FileChooser fileChooser = new FileChooser();  
 fileChooser.setTitle("Сохранить все решения");  
 fileChooser.getExtensionFilters().add(new FileChooser.ExtensionFilter("CSV файлы", "\*.csv"));  
 File file = fileChooser.showSaveDialog(new Stage());  
  
 if (file != null) {  
 try (FileWriter writer = new FileWriter(file)) {  
 writer.write("t,Euler,RK4,Adams\n");  
 for (int i = 0; i < tsLocal.length; i++) {  
 writer.write(tsLocal[i] + "," + ysLocal[0][i] + "," + ysLocal[1][i] + "," + ysLocal[2][i] + "\n");  
 }  
 System.*out*.println("Файл успешно экспортирован!");  
 } catch (IOException ex) {  
 ex.printStackTrace();  
 }  
 }  
 });  
  
  
 // Обработка кнопки "Сохранить"  
 saveButton.setOnAction(e -> {  
 if (ts != null && ys != null) {  
 FileChooser fileChooser = new FileChooser();  
 fileChooser.setTitle("Сохранить как CSV");  
 fileChooser.getExtensionFilters().add(new FileChooser.ExtensionFilter("CSV files", "\*.csv"));  
 File file = fileChooser.showSaveDialog(stage);  
  
 if (file != null) {  
 try (FileWriter writer = new FileWriter(file)) {  
 writer.write("t,y\n");  
 for (int i = 0; i < ts.length; i++) {  
 writer.write(ts[i] + "," + ys[i] + "\n");  
 }  
  
 Alert alert = new Alert(Alert.AlertType.*INFORMATION*);  
 alert.setTitle("Готово");  
 alert.setHeaderText(null);  
 alert.setContentText("Файл успешно сохранён!");  
 alert.showAndWait();  
 } catch (IOException ex) {  
 ex.printStackTrace();  
 Alert alert = new Alert(Alert.AlertType.*ERROR*);  
 alert.setTitle("Ошибка");  
 alert.setHeaderText(null);  
 alert.setContentText("Не удалось сохранить файл.");  
 alert.showAndWait();  
 }  
 }  
  
 }  
 });  
  
 Button tableButton = new Button("Показать таблицу");  
  
 tableButton.setOnAction(e -> {  
 if (ts == null || ys == null) {  
 showAlert("Нет данных", "Сначала рассчитайте решение.");  
 return;  
 }  
  
 double tau = Double.*parseDouble*(tauField.getText());  
 double tau2 = Double.*parseDouble*(tau2Field.getText());  
 double h = Double.*parseDouble*(hField.getText());  
  
 TableView<TableRowData> table = new TableView<>();  
  
 TableColumn<TableRowData, Double> colT = new TableColumn<>("t");  
 colT.setCellValueFactory(data -> data.getValue().tProperty().asObject());  
  
 TableColumn<TableRowData, Double> colY = new TableColumn<>("y(t)");  
 colY.setCellValueFactory(data -> data.getValue().yProperty().asObject());  
  
 TableColumn<TableRowData, Double> colTau1 = new TableColumn<>("y(t - τ₁)");  
 colTau1.setCellValueFactory(data -> data.getValue().yTau1Property().asObject());  
  
 TableColumn<TableRowData, Double> colTau2 = new TableColumn<>("y(t - τ₂)");  
 colTau2.setCellValueFactory(data -> data.getValue().yTau2Property().asObject());  
  
 table.getColumns().addAll(colT, colY, colTau1);  
 if (tau2 > 0) table.getColumns().add(colTau2);  
  
 for (int i = 0; i < ts.length; i++) {  
 double t = ts[i];  
 double y = ys[i];  
 double yTau1 = getInterpolatedY(t - tau, ts, ys);  
 double yTau2 = getInterpolatedY(t - tau2, ts, ys);  
 table.getItems().add(new TableRowData(t, y, yTau1, yTau2));  
 }  
  
 Stage tableStage = new Stage();  
 tableStage.setTitle("Таблица значений");  
 tableStage.setScene(new Scene(table, 600, 400));  
 tableStage.show();  
 });  
  
 //кнопка проверки ошибки двух методов  
 Button errorButton = new Button("График ошибки");  
  
 //обработчик кнопки  
 errorButton.setOnAction(e -> {  
 double a = Double.*parseDouble*(aField.getText());  
 double tau = Double.*parseDouble*(tauField.getText());  
 double tau2 = Double.*parseDouble*(tau2Field.getText());  
 double h = Double.*parseDouble*(hField.getText());  
 double tMax = Double.*parseDouble*(tMaxField.getText());  
 String phiExpr = phiField.getText();  
 String expr = functionField.getText();  
  
 DDESolver solver1 = new DDESolver();  
 solver1.setTau2(tau2);  
 solver1.setInitialFunctionExpression(phiExpr);  
 solver1.setExpression(expr);  
 solver1.solve(DDESolver.Method.*EULER*, a, tau, h, tMax);  
  
 DDESolver solver2 = new DDESolver();  
 solver2.setTau2(tau2);  
 solver2.setInitialFunctionExpression(phiExpr);  
 solver2.setExpression(expr);  
 solver2.solve(DDESolver.Method.*RUNGE\_KUTTA\_4*, a, tau, h, tMax);  
  
 List<Double> tList1 = solver1.getTList();  
 List<Double> yList1 = solver1.getYList();  
 List<Double> yList2 = solver2.getYList();  
  
 XYChart.Series<Number, Number> errorSeries = new XYChart.Series<>();  
 errorSeries.setName("Ошибка |Эйлер - РК4|");  
  
 for (int i = 0; i < tList1.size(); i++) {  
 double error = Math.*abs*(yList1.get(i) - yList2.get(i));  
 errorSeries.getData().add(new XYChart.Data<>(tList1.get(i), error));  
 }  
  
 lineChart.getData().clear();  
 lineChart.getData().add(errorSeries);  
 lineChart.setTitle("График ошибки между методами: Эйлер и РК4");  
 });  
  
 Button systemButton = new Button("Система DDE");  
  
 systemButton.setOnAction(e -> {  
 double a = Double.*parseDouble*(aField.getText());  
 double tau = Double.*parseDouble*(tauField.getText());  
 double h = Double.*parseDouble*(hField.getText());  
 double tMax = Double.*parseDouble*(tMaxField.getText());  
  
 int steps = (int) (tMax / h);  
 double[] ts = new double[steps + 1];  
 double[] xs = new double[steps + 1];  
 double[] ys = new double[steps + 1];  
  
 for (int i = 0; i <= steps; i++) {  
 ts[i] = i \* h;  
 }  
  
 // φ(t) = 1 по умолчанию для x и y при t <= 0  
 for (int i = 0; i <= steps; i++) {  
 if (ts[i] <= 0) {  
 xs[i] = 1.0;  
 ys[i] = 1.0;  
 } else {  
 double t = ts[i];  
 double t\_tau = t - tau;  
 double xTau = getSystemYAt(t\_tau, ts, xs);  
 double yTau = getSystemYAt(t\_tau, ts, ys);  
 xs[i] = xs[i - 1] + h \* (-a \* yTau);  
 ys[i] = ys[i - 1] + h \* (xTau);  
 }  
 }  
  
 XYChart.Series<Number, Number> xSeries = new XYChart.Series<>();  
 xSeries.setName("x(t)");  
 XYChart.Series<Number, Number> ySeries = new XYChart.Series<>();  
 ySeries.setName("y(t)");  
  
 for (int i = 0; i <= steps; i++) {  
 xSeries.getData().add(new XYChart.Data<>(ts[i], xs[i]));  
 ySeries.getData().add(new XYChart.Data<>(ts[i], ys[i]));  
 }  
  
 lineChart.getData().clear();  
 lineChart.getData().addAll(xSeries, ySeries);  
 lineChart.setTitle("Система DDE: x'(t) = -a y(t-τ), y'(t) = x(t-τ)");  
 });  
  
  
 // ==== Первый столбик: Параметры ====  
 VBox column1 = new VBox(10,  
 new Label("Параметры:"),  
 new Label("a:"), aField,  
 new Label("τ (tau):"), tauField,  
 new Label("τ₂:"), tau2Field,  
 new Label("h (шаг):"), hField,  
 new Label("t max:"), tMaxField  
 );  
 column1.setPadding(new Insets(10));  
 column1.setPrefWidth(180);  
  
  
// ==== Второй столбик: Выражения ====  
 VBox column2 = new VBox(10,  
 new Label("Функции:"),  
 new Label("φ(t):"), phiField,  
 new Label("f(t, y, ...):"), functionField,  
 new Label("Пример:"), exampleCombo,  
 new Label("Метод:"), methodCombo  
 );  
 column2.setPadding(new Insets(10));  
 column2.setPrefWidth(260);  
  
  
// ==== Третий столбик: Кнопки ====  
 VBox column3 = new VBox(10,  
 new Label("Действия:"),  
 solveButton,  
 compareButton,  
 errorButton,  
 systemButton,  
 tableButton,  
 saveButton,  
 exportAllButton,  
 helpButton  
 );  
 column3.setPadding(new Insets(10));  
 column3.setPrefWidth(200);  
  
  
// ==== Горизонтальный ряд из 3 колонок ====  
 HBox topRow = new HBox(30, column1, column2, column3);  
  
  
// ==== Главный контейнер: три столбца сверху + график ====  
 VBox root = new VBox(20, topRow, lineChart);  
 root.setPadding(new Insets(10));  
 lineChart.setPrefHeight(500); // жирный график  
 stage.setScene(new Scene(root, 1200, 800));  
 stage.setTitle("DDE Solver");  
 stage.show(); // <-- ОЧЕНЬ ВАЖНО  
  
  
 solveButton.setTooltip(new Tooltip("Рассчитать решение с выбранным методом"));  
 compareButton.setTooltip(new Tooltip("Построить графики всех трёх методов"));  
 errorButton.setTooltip(new Tooltip("Показать разницу между методами"));  
 systemButton.setTooltip(new Tooltip("Решение системы с задержкой"));  
 tableButton.setTooltip(new Tooltip("Открыть таблицу значений"));  
 saveButton.setTooltip(new Tooltip("Сохранить текущее решение в файл CSV"));  
 exportAllButton.setTooltip(new Tooltip("Сохранить все три метода в один CSV-файл"));  
  
  
  
 }  
  
 private void showErrorDialog(String ошибкаВычислений, String message) {  
 }  
  
 private void updateChart(DDESolver solver, String methodName, double a, double tau) {  
 XYChart.Series<Number, Number> series = new XYChart.Series<>();  
 series.setName(String.*format*("%s (a=%.2f, τ=%.2f)", methodName, a, tau));  
  
 List<Double> tList = solver.getTList();  
 List<Double> yList = solver.getYList();  
  
 if (tList.size() != yList.size()) {  
 showErrorDialog("Ошибка данных", "Несовпадение размеров массивов");  
 return;  
 }  
  
 }  
 private void showAlert(String title, String message) {  
 Alert alert = new Alert(Alert.AlertType.*ERROR*);  
 alert.setTitle(title);  
 alert.setHeaderText(null);  
 alert.setContentText(message);  
 alert.showAndWait();  
 }  
  
 private double getSystemYAt(double t, double[] ts, double[] ys) {  
 if (t <= ts[0]) return 1.0;  
 if (t >= ts[ts.length - 1]) return ys[ys.length - 1];  
  
 for (int i = 0; i < ts.length - 1; i++) {  
 if (ts[i] <= t && t < ts[i + 1]) {  
 double t1 = ts[i];  
 double t2 = ts[i + 1];  
 double y1 = ys[i];  
 double y2 = ys[i + 1];  
 return y1 + (y2 - y1) \* (t - t1) / (t2 - t1);  
 }  
 }  
 return ys[0];  
 }  
  
  
 private double getInterpolatedY(double t, double[] ts, double[] ys) {  
 if (t <= ts[0]) return ys[0];  
 if (t >= ts[ts.length - 1]) return ys[ys.length - 1];  
  
 for (int i = 0; i < ts.length - 1; i++) {  
 if (ts[i] <= t && t < ts[i + 1]) {  
 double t1 = ts[i];  
 double t2 = ts[i + 1];  
 double y1 = ys[i];  
 double y2 = ys[i + 1];  
 return y1 + (y2 - y1) \* (t - t1) / (t2 - t1);  
 }  
 }  
 return ys[0];  
 }  
  
  
  
  
 public static void main(String[] args) {  
 *launch*();  
 }  
 }

======= CSVExporter.java =======

package dde;  
  
import javafx.stage.FileChooser;  
import javafx.stage.Stage;  
  
import java.io.File;  
import java.io.PrintWriter;  
import java.io.IOException;  
  
public class CSVExporter {  
 public static void save(double[] ts, double[] ys) {  
 FileChooser fileChooser = new FileChooser();  
 fileChooser.setTitle("Сохранить как CSV");  
 fileChooser.setInitialFileName("dde\_output.csv");  
 fileChooser.getExtensionFilters().add(new FileChooser.ExtensionFilter("CSV файлы", "\*.csv"));  
  
 File file = fileChooser.showSaveDialog(new Stage());  
 if (file != null) {  
 try (PrintWriter writer = new PrintWriter(file)) {  
 writer.println("t,y(t)");  
 for (int i = 0; i < ts.length; i++) {  
 writer.println(ts[i] + "," + ys[i]);  
 }  
 System.*out*.println("CSV успешно сохранён: " + file.getAbsolutePath());  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
}

======= TableRowData =======

package dde;  
  
import javafx.beans.property.\*;  
  
public class TableRowData {  
 private final DoubleProperty t = new SimpleDoubleProperty();  
 private final DoubleProperty y = new SimpleDoubleProperty();  
 private final DoubleProperty yTau1 = new SimpleDoubleProperty();  
 private final DoubleProperty yTau2 = new SimpleDoubleProperty();  
  
 public TableRowData(double t, double y, double yTau1, double yTau2) {  
 this.t.set(t);  
 this.y.set(y);  
 this.yTau1.set(yTau1);  
 this.yTau2.set(yTau2);  
 }  
  
 public DoubleProperty tProperty() { return t; }  
 public DoubleProperty yProperty() { return y; }  
 public DoubleProperty yTau1Property() { return yTau1; }  
 public DoubleProperty yTau2Property() { return yTau2; }  
}

======= GraphWindow.java =======

package dde;  
  
import javafx.application.Application;  
import javafx.scene.Scene;  
import javafx.scene.chart.\*;  
import javafx.scene.layout.StackPane;  
import javafx.stage.Stage;  
  
import java.util.List;  
  
public class GraphWindow extends Application {  
  
 public static List<Double> *tData*;  
 public static List<Double> *yData*;  
  
 @Override  
 public void start(Stage stage) {  
 stage.setTitle("График y(t)");  
  
 NumberAxis xAxis = new NumberAxis();  
 xAxis.setLabel("t");  
  
 NumberAxis yAxis = new NumberAxis();  
 yAxis.setLabel("y(t)");  
  
 LineChart<Number, Number> lineChart = new LineChart<>(xAxis, yAxis);  
 lineChart.setTitle("Решение DDE");  
  
 XYChart.Series<Number, Number> series = new XYChart.Series<>();  
 series.setName("y(t)");  
  
 for (int i = 0; i < *tData*.size(); i++) {  
 series.getData().add(new XYChart.Data<>(*tData*.get(i), *yData*.get(i)));  
 }  
  
 lineChart.getData().add(series);  
  
 StackPane root = new StackPane();  
 root.getChildren().add(lineChart);  
  
 Scene scene = new Scene(root, 800, 600);  
 stage.setScene(scene);  
 stage.show();  
 }  
}

======= DDESolver.java =======

package dde;  
  
import net.objecthunter.exp4j.Expression;  
import net.objecthunter.exp4j.ExpressionBuilder;  
  
  
import java.util.ArrayList;  
import java.util.List;  
import java.util.function.DoubleUnaryOperator;  
  
public class DDESolver {  
  
 public enum Method {  
 *EULER*,  
 *RUNGE\_KUTTA\_4*,  
 *ADAMS\_BASHFORTH\_2* }  
 private String phiExprStr = "1";  
  
 private final List<Double> tList = new ArrayList<>();  
 private final List<Double> yList = new ArrayList<>();  
 private DoubleUnaryOperator phi = t -> 1.0;  
  
 private String expressionStr = "-a \* y\_tau1"; // значение по умолчанию  
 private double tau2 = 0;  
  
 public void setInitialFunctionExpression(String expr) {  
 this.phiExprStr = expr;  
 }  
  
 public void setInitialFunction(DoubleUnaryOperator phi) {  
 this.phi = phi;  
 }  
  
 public void setExpression(String expr) {  
 this.expressionStr = expr;  
 }  
  
 public void setTau2(double tau2) {  
 this.tau2 = tau2;  
 }  
  
 public List<Double> getTList() { return tList; }  
 public List<Double> getYList() { return yList; }  
  
 private double getPhi(double t) {  
 try {  
 Expression expr = new ExpressionBuilder(phiExprStr)  
 .variable("t")  
 .build()  
 .setVariable("t", t);  
 return expr.evaluate();  
 } catch (Exception e) {  
 System.*err*.println("Ошибка в выражении φ(t): " + e.getMessage());  
 return 1.0; // fallback  
 }  
 }  
  
 private double f(double t, double y, double yTau1, double yTau2, double a) {  
 try {  
 Expression expr = new ExpressionBuilder(expressionStr)  
 .variables("t", "y", "y\_tau1", "y\_tau2", "a")  
 .build()  
 .setVariable("t", t)  
 .setVariable("y", y)  
 .setVariable("y\_tau1", yTau1)  
 .setVariable("y\_tau2", yTau2)  
 .setVariable("a", a);  
 return expr.evaluate();  
 } catch (Exception e) {  
 e.printStackTrace();  
 return 0;  
 }  
 }  
  
 public void solve(Method method, double a, double tau, double h, double tMax) {  
 tList.clear();  
 yList.clear();  
  
 switch (method) {  
 case *EULER* -> solveEuler(a, tau, h, tMax);  
 case *RUNGE\_KUTTA\_4* -> solveRK4(a, tau, h, tMax);  
 case *ADAMS\_BASHFORTH\_2* -> solveAdams(a, tau, h, tMax);  
 }  
 }  
  
 private void solveEuler(double a, double tau, double h, double tMax) {  
 int steps = (int)(tMax / h);  
 for (int i = 0; i <= steps; i++) {  
 double t = i \* h;  
 double y;  
  
 if (t <= 0) {  
 y = getPhi(t);  
 } else {  
 double yTau1 = getYAt(t - tau, h);  
 double yTau2 = getYAt(t - tau2, h);  
 double yPrev = yList.get(i - 1);  
 y = yPrev + h \* f(t, yPrev, yTau1, yTau2, a);  
 }  
  
 tList.add(t);  
 yList.add(y);  
 }  
 }  
  
 private void solveRK4(double a, double tau, double h, double tMax) {  
 int steps = (int)(tMax / h);  
 for (int i = 0; i <= steps; i++) {  
 double t = i \* h;  
 double y;  
  
 if (t <= 0) {  
 y = getPhi(t);  
 } else {  
 double yPrev = yList.get(i - 1);  
  
 double k1 = f(t, yPrev, getYAt(t - tau, h), getYAt(t - tau2, h), a);  
 double k2 = f(t + h / 2, yPrev + h \* k1 / 2, getYAt(t + h / 2 - tau, h), getYAt(t + h / 2 - tau2, h), a);  
 double k3 = f(t + h / 2, yPrev + h \* k2 / 2, getYAt(t + h / 2 - tau, h), getYAt(t + h / 2 - tau2, h), a);  
 double k4 = f(t + h, yPrev + h \* k3, getYAt(t + h - tau, h), getYAt(t + h - tau2, h), a);  
  
 y = yPrev + (h / 6) \* (k1 + 2 \* k2 + 2 \* k3 + k4);  
 }  
  
 tList.add(t);  
 yList.add(y);  
 }  
 }  
  
 private void solveAdams(double a, double tau, double h, double tMax) {  
 int steps = (int)(tMax / h);  
  
 for (int i = 0; i <= 2; i++) {  
 double t = i \* h;  
 double y;  
 if (t <= 0) {  
 y = getPhi(t);  
 } else {  
 double yTau1 = getYAt(t - tau, h);  
 double yTau2 = getYAt(t - tau2, h);  
 double yPrev = yList.get(i - 1);  
 y = yPrev + h \* f(t, yPrev, yTau1, yTau2, a);  
 }  
 tList.add(t);  
 yList.add(y);  
 }  
  
 for (int i = 3; i <= steps; i++) {  
 double t = i \* h;  
  
 double f\_n = f(t, yList.get(i - 1), getYAt(t - tau, h), getYAt(t - tau2, h), a);  
 double f\_n1 = f(t - h, yList.get(i - 2), getYAt(t - h - tau, h), getYAt(t - h - tau2, h), a);  
  
 double y = yList.get(i - 1) + h / 2 \* (3 \* f\_n - f\_n1);  
  
 tList.add(t);  
 yList.add(y);  
 }  
 }  
  
 private double getYAt(double t, double h) {  
 if (t <= 0) return phi.applyAsDouble(t);  
 ;  
  
 int index = (int)(t / h);  
 if (index >= yList.size() - 1) return yList.get(yList.size() - 1);  
  
 double t1 = tList.get(index);  
 double t2 = tList.get(index + 1);  
 double y1 = yList.get(index);  
 double y2 = yList.get(index + 1);  
  
 return y1 + (y2 - y1) \* (t - t1) / (t2 - t1);  
 }  
}