МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ  
«Харківський Політехнічний Інститут»  
  
Кафедра «Стратегічного Управління»

ЗВІТ

з лабораторної роботи № 5

# «Використання уніфікованої мови моделювання»

Перевірила: старший викладач  
Вільхівська О. В.  
Виконав: ст. гр. КН-27

Харків, 2018

Лабораторна робота № 5  
Тема: Використання уніфікованої мови моделювання  
Завдання

**1. Індивідуальне завдання**

Необхідно спроектувати програму, у якій здійснюється подання та обробка даних відповідно до індивідуального завдання. Слід створити діаграми варіантів використання, класів, послідовності та діяльності.

* Діаграма варіантів використання передбачає відображення функцій читання (в тому числі з XML-документу), модифікації, додавання, видалення, сортування (не менш ніж за двома ознаками), пошуку (не менш ніж за двома ознаками) та зберігання даних (в тому числі в XML-документі).
* Діаграма класів повинна включати всі абстрактні, похідні й допоміжні класи, необхідні для програмної реалізації. Діаграма не повинна містити стандартні класи Java.
* Діаграми послідовності повинні відображати взаємодію об'єктів програмного забезпечення під час реалізації окремих варіантів використання.
* Діаграми діяльності повинні відображати алгоритми окремих операцій.

Варіанти сутностей визначені в індивідуальному завданні до [лабораторної роботи № 5](http://iwanoff.inf.ua/algorithmization_2/LabTraining05.html) курсу "Алгоритмізація та програмування" (друга частина) попереднього семестру. На діаграмі класів слід відтворити структуру класів, яка сформувалася під час виконання попередніх лабораторних робіт цього курсу.

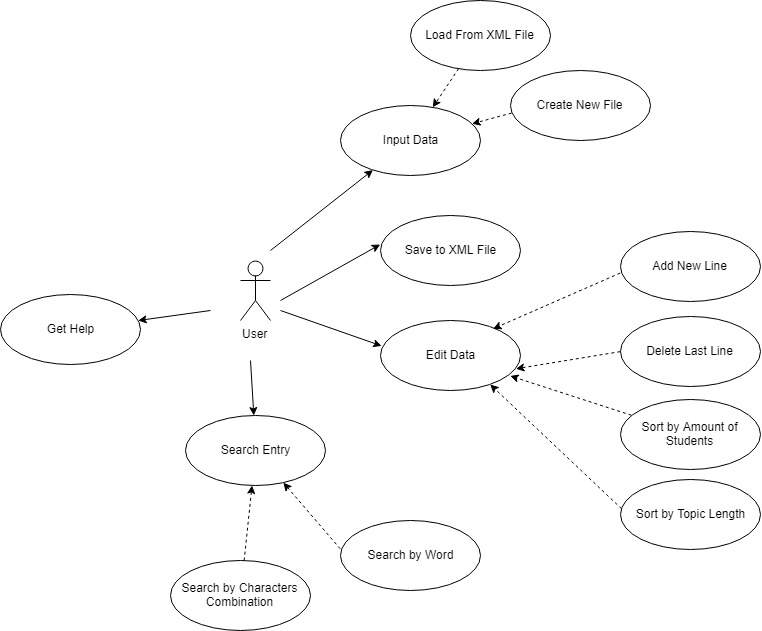
### 2. Побудова графіків довільних функцій

Спроектувати програму, в якій здійснюється побудова графіків довільних функцій. Слід створити діаграми варіантів використання, класів, послідовності та діяльності.

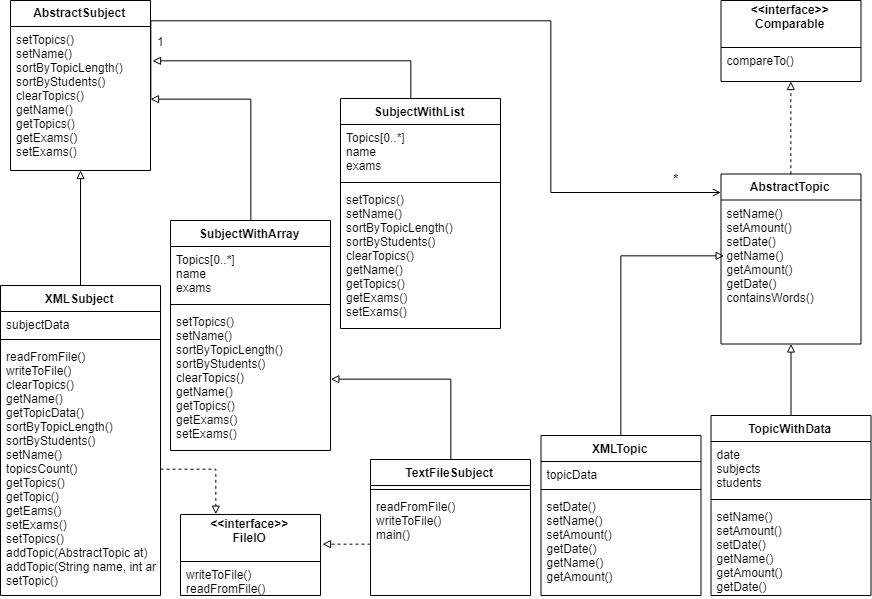
Хід роботи

**Завдання 1. Індивідуальне завдання**

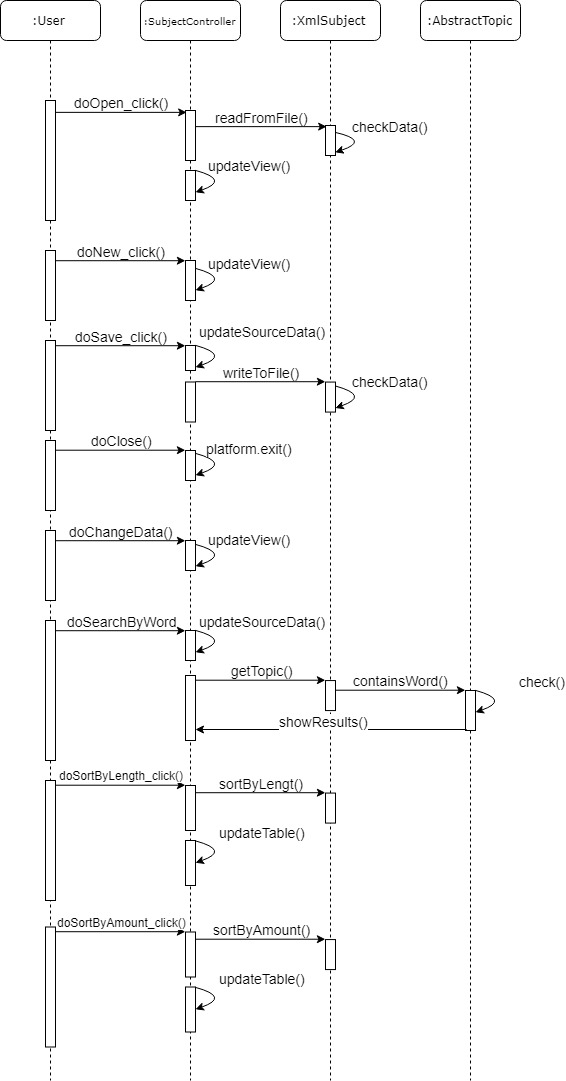
### 1. Діаграма варіантів використання



2. Діаграма класів



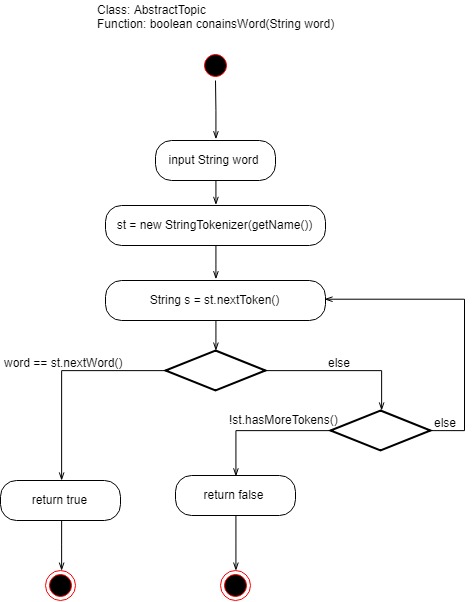
3. Діаграма послідовності



4. Діаграмі діяльності

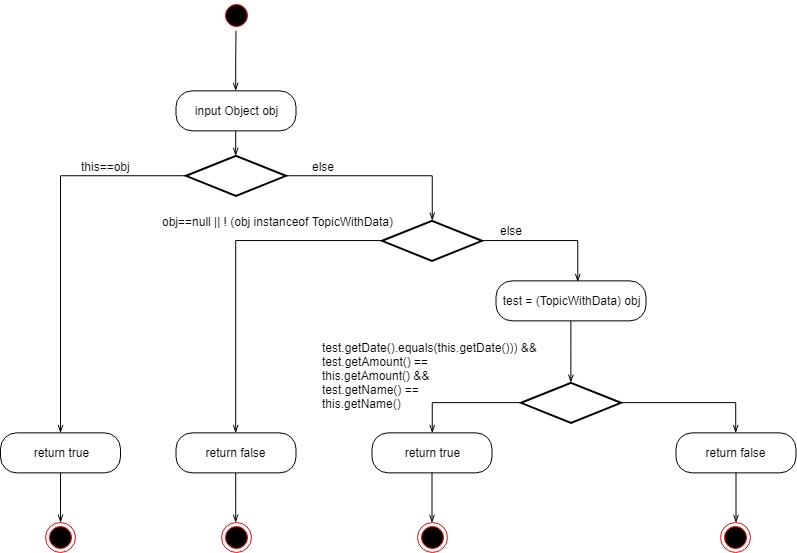
Class: AbstractTopic

Function: boolean containsWord(String word)



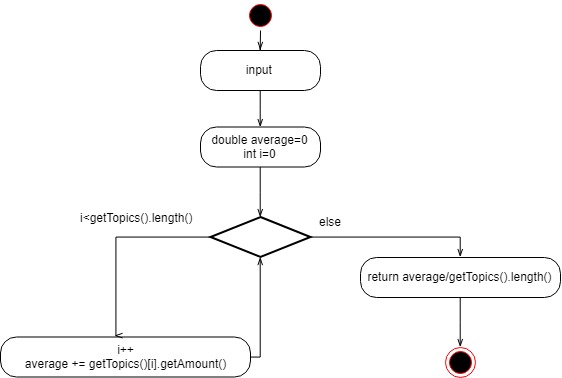
Class: AbstractTopic

Function: boolean equals(Object obj)



Class: AbstractSubject

Function: double getAverage ()



### Завдання 2. Побудова графіків довільних функцій

*Вміст GuiFX.java*

|  |
| --- |
| package task\_2;  import javafx.application.Application;  import javafx.fxml.FXMLLoader;  import javafx.scene.Scene;  import javafx.scene.chart.LineChart;  import javafx.scene.chart.NumberAxis;  import javafx.scene.chart.XYChart;  import javafx.scene.layout.BorderPane;  import javafx.stage.Stage;  import java.io.IOException;  import java.util.function.DoubleUnaryOperator;  public class GUIFX extends Application {  public void start(Stage primaryStage) {  FXMLLoader loader = new FXMLLoader();  try {  BorderPane root = (BorderPane)FXMLLoader.load(getClass().getResource("GUI.fxml"));  Scene scene = new Scene(root);  primaryStage.setScene(scene);  primaryStage.setTitle("Графики");  primaryStage.show();  } catch (IOException e) {  e.printStackTrace();  }  }  public static void main(String[] args) {  launch(args);  }  } |

*Вміст GUIController.java*

|  |
| --- |
| **package** task\_2;  **import** java.io.IOException;  **import** java.io.PrintWriter;  **import** java.net.URL;  **import** java.util.ResourceBundle;  **import** java.util.function.DoubleBinaryOperator;  **import** java.util.function.DoubleUnaryOperator;  **import** javax.script.Bindings;  **import** javax.script.ScriptContext;  **import** javax.script.ScriptEngine;  **import** javax.script.ScriptEngineManager;  **import** javax.script.ScriptException;  **import** javax.tools.JavaCompiler;  **import** javax.tools.ToolProvider;  **import** javafx.application.Platform;  **import** javafx.event.ActionEvent;  **import** javafx.fxml.FXML;  **import** javafx.fxml.Initializable;  **import** javafx.scene.chart.LineChart;  **import** javafx.scene.chart.XYChart;  **import** javafx.scene.control.Label;  **import** javafx.scene.control.TableColumn;  **import** javafx.scene.control.TableView;  **import** javafx.scene.control.TextField;  **import** javafx.scene.control.TableColumn.CellEditEvent;  **import** javafx.scene.control.cell.PropertyValueFactory;  **import** javafx.scene.control.cell.TextFieldTableCell;  **public** **class** GUIController **implements** Initializable{  @FXML TextField textFieldF;  @FXML LineChart<Number,Number> chart;    **private** **double** x=0;  **private** **static** DoubleUnaryOperator *f*=x->x; // Default    **private** ScriptEngineManager factory = **new** ScriptEngineManager();  **private** ScriptEngine engine = factory.getEngineByName("JavaScript");  **private** Bindings bindings = engine.getBindings(ScriptContext.***ENGINE\_SCOPE***);    @FXML **public** **void** doFChanged(ActionEvent event) **throws** ScriptException {  *f* = (x)->{  **try** {  **return** Double.*valueOf*(engine.eval("y = "+textFieldF.getText()).toString());  } **catch** (ScriptException e) {  e.printStackTrace();  }  **return** x;  };  updateView();  }    @FXML **public** **void** doClear(ActionEvent event) {  textFieldF.clear();  chart.getData().clear();  }  @FXML **public** **void** doExit(ActionEvent event) {  Platform.*exit*();  }  **public** **static** **void** setF(DoubleUnaryOperator func) {  *f* = func;  System.***out***.println(func);  }  **public** LineChart<Number, Number> updateView() {  chart.setCreateSymbols(**false**);  chart.getData().clear();  **double** xTo = 10, xFrom = -10;  **double** h = (xTo - xFrom) / 100;  // Додаємо ім'я і точки першої функції:  XYChart.Series<Number, Number> fSeries = **new** XYChart.Series<>();  fSeries.setName("f(x)");  **for** (**double** x = xFrom; x <= xTo; x += h) {  engine.put("x", **new** Double(x));  fSeries.getData().add(**new** XYChart.Data<>(x, *f*.applyAsDouble(x)));  }  chart.getData().addAll(fSeries);  **return** chart;  }  **public** **static** **void** main(String[] args) {  GUIFX.*main*(**new** String[] {});  }  @Override  **public** **void** initialize(URL location, ResourceBundle resources) {  updateView();  }  } |

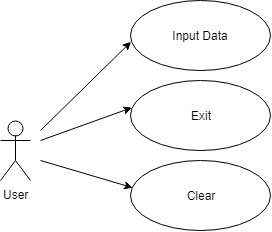
*Вміст GUI.fxml*

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <?import javafx.scene.image.\*?>  <?import javafx.scene.chart.\*?>  <?import javafx.scene.control.\*?>  <?import java.lang.\*?>  <?import javafx.scene.layout.\*?>  <BorderPane maxHeight=*"-Infinity"* maxWidth=*"-Infinity"* minHeight=*"-Infinity"* minWidth=*"-Infinity"* prefHeight=*"400.0"* prefWidth=*"600.0"* xmlns=*"http://javafx.com/javafx/8"* xmlns:fx=*"http://javafx.com/fxml/1"* fx:controller=*"task\_2.GUIController"*>  <center>  <LineChart fx:id=*"chart"* BorderPane.alignment=*"CENTER"*>  <xAxis>  <NumberAxis side=*"BOTTOM"* />  </xAxis>  <yAxis>  <NumberAxis side=*"LEFT"* />  </yAxis>  </LineChart>  </center>  <bottom>  <AnchorPane prefHeight=*"93.0"* prefWidth=*"600.0"* BorderPane.alignment=*"CENTER"*>  <children>  <Label layoutX=*"50.0"* layoutY=*"21.0"* text=*"Функция f(x)"* />  <TextField fx:id=*"textFieldF"* layoutX=*"48.0"* layoutY=*"47.0"* onAction=*"#doFChanged"* prefHeight=*"25.0"* prefWidth=*"267.0"* />  </children>  </AnchorPane>  </bottom>  <top>  <MenuBar BorderPane.alignment=*"CENTER"*>  <menus>  <Menu mnemonicParsing=*"false"* text=*"Файл"*>  <items>  <MenuItem mnemonicParsing=*"false"* onAction=*"#doClear"* text=*"Очистить"* />  <MenuItem mnemonicParsing=*"false"* onAction=*"#doExit"* text=*"Выход"* />  </items>  </Menu>  </menus>  </MenuBar>  </top>  </BorderPane> |

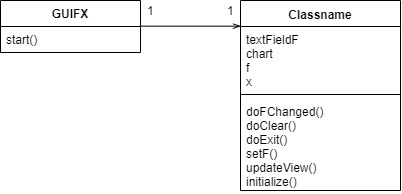
*Приклад виконання програми*

|  |
| --- |
|  |

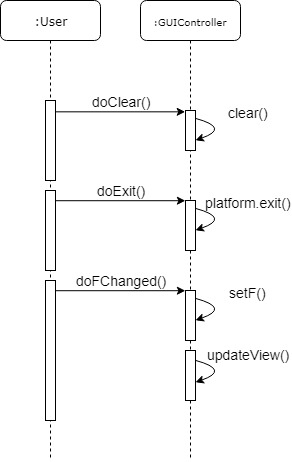
1. Диаграмма вариантов использования



1. Диаграмма классов



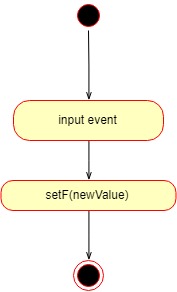
1. Діаграма послідовності



1. Діаграми діяльності

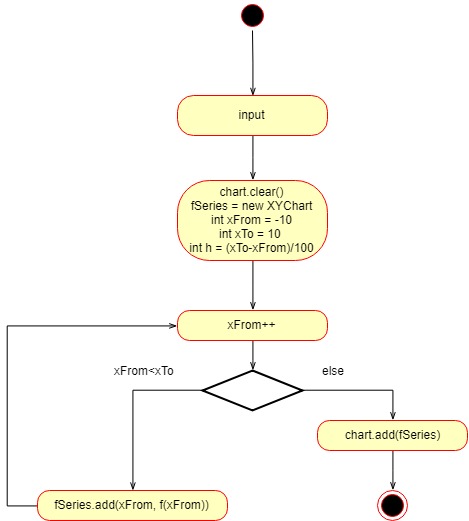
Class: GUIController.java

Function: doFChanged



Class: GUIController.java

Function: doUpdateView()



Висновки

UML діаграми найчастіше використовують для великих проектів. Перед розробкою самого кода треба розробити певні діаграми, що вдовольнятимуть вимогам користувача. UML діаграми є актуальними для програмісті, що керують проектом з технічної точки зору. Коли необхідним є коротоко й лаконічно описати проект UML є першим варінтом для досягнення цієй мети. У цій лабораторній були приведені діаграми класів, варіантів використування, послідовності, діяльності для двох невеликих проектів.