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

ЗВІТ

з лабораторної роботи № 3

«Створення програм графічного інтерфейсу користувача»

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

Харків, 2018

Лабораторная робота № 1  
Тема: Создание прогрмм с графическим интерфейсом  
Задания

### Індивідуальне завдання

Необхідно реалізувати мовою Java за допомогою засобів JavaFX застосунок графічного інтерфейсу користувача, в якому здійснюється обробка даних індивідуальних завдань попередніх лабораторних робіт. Головне вікно повинно містити меню, в якому необхідно реалізувати такі функції:

* створення нового набору даних
* завантаження даних з XML-документу для редагування
* зберігання змінених даних в XML-документі
* пошук за ознаками, визначеними в лабораторній роботі № 5 курсу "Алгоритмізація та програмування" (друга частина)
* здійснення сортування за ознаками, визначеними в лабораторній роботі № 6 курсу "Алгоритмізація та програмування" (друга частина)
* отримання вікна "Про програму" з даними про програму і автора.

У лівій частині вікна слід розташувати рядки для введення скалярних даних, область відображення для результатів пошуку, а також кнопки, які забезпечують виконання основних функцій програми. В середній частині вікна слід розташувати таблицю для відображення та редагування даних.

### Перелік для опису місяців року

Створити перелік "Місяць". Необхідно визначати у конструкторі і зберігати кількість днів (для невисокосного року). Додати методи отримання попереднього та наступного місяця, а також функцію, яка повертає сезон для кожного місяця. Передбачити виведення місяців українською (російською) мовою. Створити статичну функцію виведення даних про усі місяці. Протестувати переліку в функції main() тестового класу.

### Дані про користувачів

Представити дані про користувачів у вигляді асоціативного масиву (ім'я / пароль) з припущенням, що всі імена користувачів різні. Вивести дані про користувачів з довжиною пароля більше 6 символів.

### Міні-калькулятор

Створити застосунок графічного інтерфейсу користувача, в якому після введення чисел у двох рядках типу TextField виконується одна з чотирьох арифметичних дій (залежно від вибраної кнопки RadioButton). Результат виводиться в інше текстове поле.

### Словник (додаткове завдання)

Розробити програму графічного інтерфейсу користувача перегляду слів невеличкого англо-українського (англо-російського) словника. Реалізувати функції пошуку слова, додавання нових слів.

Хід роботи

### Завдання 1. Індивідуальне завдання

*Код програми*

*Вміст файлу AbstractSubject.java*

|  |
| --- |
| package task\_1;  import java.util.\*;  public abstract class AbstractSubject {  public abstract void setTopics(AbstractTopic[] topics);  public abstract void setName(String str);  public abstract void sortByTopicLength();  public abstract void sortByStudents();  public abstract void ClearTopics();    public abstract String getName();  public abstract AbstractTopic[] getTopics();  public abstract boolean getExams();  public abstract void setExams(boolean b);  // public abstract boolean addTopic(AbstractTopic topic);  public double getAverage() {  double average=0;  for (AbstractTopic topic: getTopics()) {  average+=topic.getAmount();  }  return average/=getTopics().length;  }    public static AbstractTopic[] addToArray(AbstractTopic[] arr, AbstractTopic item) {  AbstractTopic[] newArr;  if (arr!=null) {  newArr = new AbstractTopic[arr.length+1];  System.arraycopy(arr, 0, newArr, 0, arr.length);  }  else {  newArr = new AbstractTopic[1];  }  newArr[newArr.length-1]=item;  return newArr;  }  public AbstractTopic getMin() {  AbstractTopic lecture = getTopics()[0];  for (int i=1; i<getTopics().length; i++) {  if(getTopics()[i].getAmount()>lecture.getAmount()) {  lecture = getTopics()[i];  }  }  return lecture;  }  public AbstractTopic[] findWord(String word) {  System.out.println("\nInfo about lecture containing word " + word);  AbstractTopic[] at = {};  for (AbstractTopic topic: getTopics()) {  if (topic.containsWord(word)) {  System.out.println(topic);  addToArray(at,topic);  }  }  return at; // Р’РѕР·РІСЂР°С‰Р°РµС‚ РјР°СЃСЃРёРІ СЃ РЅР°Р№РґРµРЅРЅС‹РјРё Р»РµРєС†РёСЏРјРё  }  @Override  public String toString() { //overloading toString() method  String result = "Name: " + getName() + " Exams: " + getExams() + " Amount of practices: " + getTopics().length;  for (int i = 0; i < getTopics().length; i++) {  result += "\n" + getTopics()[i]; // Р Р°РЅСЊС€Рµ Р±С‹Р» РјРµС‚РѕРґ, РІРѕР·СЂР°С‰Р°СЋС‰РёР№ РєРѕРЅРєСЂРµС‚РЅС‹Р№ СЌР»РµРјРµРЅС‚  }  return result;  }  @Override  public boolean equals(Object obj) {  if (this==obj) return true;  if ((obj==null)||!(obj instanceof AbstractSubject)) return false;  AbstractSubject as = (AbstractSubject) obj;  if (!getName().equals(as.getName())) return false;  return Arrays.equals(getTopics(), as.getTopics());    }  public boolean addTopic(AbstractTopic topic) {  if (getTopics()!=null) {  for (AbstractTopic p : getTopics()) {  if (p.equals(topic)) {  return false;  }  }  }  setTopics(addToArray(getTopics(), topic));  return true;  }  protected AbstractSubject createSubject() {  setName("OOP");  System.out.println("\nAdding new practices to " + getName());  System.out.println(addTopic(new TopicWithData("21,9,2018", "Generics", 8)));  System.out.println(addTopic(new TopicWithData("2,10,2018", "Exceptions", 3)));  System.out.println(addTopic(new TopicWithData("16,10,2018", "GUI", 8)));  System.out.println(addTopic(new TopicWithData("30,10,2018", "Reflection", 7)));  System.out.println(addTopic(new TopicWithData("30,10,2018", "Reflection", 7)));  return this;  }  public void testSubject() {  sortByStudents(); // РЎРѕСЂС‚РёСЂСѓРµС‚ РїРѕ РєРѕР»РёС‡РµСЃС‚РІСѓ СЃС‚СѓРґРµРЅС‚РѕРІ РЅР° РїР°СЂРµ  System.out.println("\nSorting by students amount: \n" + this); // РЎ РїРѕРјРѕС‰СЊСЋ РїРµСЂРµРіСЂСѓР·РєРё toString() РІС‹РІРѕРґРёС‚ РІСЃСЋ РёРЅС„Сѓ Рѕ СЃРїРёСЃРєРµ Р»РµРєС†РёР№  sortByTopicLength(); // РЎРѕСЂС‚РёСЂСѓРµС‚ РїРѕ РґР»РёРЅРµ РЅР°Р·РІР°РЅРёСЏ Р»РµРєС†РёРё  System.out.println("\nSorting by the length of a topic name: \n" + this);  findWord("Reflection"); // РџРёС€РµС‚ РёРЅС„Сѓ Рѕ РїСЂРµРґРјРµС‚Р°С… СЃ РЅР°Р№РґРµРЅРЅС‹Рј СЃР»РѕРІРѕРј  }  public int topicsCount() {  // TODO Auto-generated method stub  return 0;  }  public AbstractTopic getTopic(int i) {  // TODO Auto-generated method stub  return null;  }  public void setTopic(int i, AbstractTopic topic) {  // TODO Auto-generated method stub    }  public boolean addTopic(String name, int amount, String date) {  // TODO Auto-generated method stub  return false;  }  } |

*Вміст файлу AbstractTopic.java*

|  |
| --- |
| package task\_1;  import java.util.\*;  public abstract class AbstractTopic implements Comparable<AbstractTopic>{  public abstract void setName(String name);  public abstract void setAmount(int amount);    public abstract String getDate();  public abstract String getName();  public abstract int getAmount();  public boolean containsWord(String word) {  StringTokenizer st = new StringTokenizer(getName());  String s;  while (st.hasMoreTokens()) {  s = st.nextToken();  if (s.toLowerCase().equals(word.toLowerCase())) {  return true;  }  }  return false;  }  @Override  public int compareTo (AbstractTopic at) {  return Integer.compare(getAmount(), at.getAmount());  }  @Override  public boolean equals(Object obj) {  if (this == obj)  return true;  if (obj==null || !(obj instanceof TopicWithData))  return false;  TopicWithData test = (TopicWithData) obj;  return (test.getDate().equals(getDate())) &&  test.getAmount() == getAmount() &&  test.getName() == getName();    }  @Override  public String toString() { //overloading toString() method  return ("Date: " + getDate() + " Subject: " + getName() + " Students: " + getAmount());  }  protected void testTopic() {  setDate("14,12,2018");  setName("Collections");  setAmount(5);  }  public void setDate(String date) {}  } |

*Вміст файлу XMLTopic.java*

|  |
| --- |
| package task\_1.xml;  import task\_1.xml.SubjectData;  import task\_1.xml.SubjectData.TopicData;  import task\_1.AbstractTopic;  public class XMLTopic extends AbstractTopic{  SubjectData.TopicData topicData;    public XMLTopic (SubjectData.TopicData topicData){  this.topicData = topicData;  }  @Override  public void setDate(String date) {  topicData.setDate(date);  }  @Override  public void setName(String name) {  topicData.setName(name);  }  @Override  public void setAmount(int amount) {  topicData.setAmount(amount);;  }  @Override  public String getDate() {  return topicData.getDate();  }  @Override  public String getName() {  return topicData.getName();  }  @Override  public int getAmount() {  return topicData.getAmount();  }  } |

*Вміст файлу XMLSubject.java*

|  |
| --- |
| package task\_1.xml;  import task\_1.AbstractTopic;  import task\_1.AbstractSubject;  import task\_1.xml.SubjectData;  import task\_1.xml.SubjectData.TopicData;  import task\_1.WriteToFile.FileIO;  import javax.xml.bind.JAXBContext;  import javax.xml.bind.JAXBException;  import javax.xml.bind.Marshaller;  import javax.xml.bind.Unmarshaller;  import java.io.FileInputStream;  import java.io.FileNotFoundException;  import java.io.FileWriter;  import java.io.IOException;  import java.util.Collections;  import java.util.Comparator;  public class XMLSubject extends AbstractSubject implements FileIO {  private SubjectData subjectData = new SubjectData();    @Override  public void readFromFile(String fileName) throws JAXBException, FileNotFoundException {  JAXBContext jaxbContext = JAXBContext.newInstance("task\_1");  Unmarshaller unmarshaller = jaxbContext.createUnmarshaller();  subjectData = (SubjectData) unmarshaller.unmarshal(new FileInputStream(fileName));  }  @Override  public void writeToFile(String fileName) throws Exception {  JAXBContext jaxbContext = JAXBContext.newInstance("task\_1");  Marshaller marshaller = jaxbContext.createMarshaller();  marshaller.setProperty(Marshaller.JAXB\_FORMATTED\_OUTPUT, Boolean.TRUE);  marshaller.marshal(subjectData, new FileWriter(fileName));  }  @Override  public void ClearTopics() {  subjectData.getTopicData().clear();  }  @Override  public void setName(String str) {  subjectData.setName(str);  }  public SubjectData.TopicData getTopicData(int i){  return subjectData.getTopicData().get(i);  }  @Override  public void sortByTopicLength() {  Collections.sort(subjectData.getTopicData(),  Comparator.comparing((t1)->t1.getName().length()));  }  @Override  public void sortByStudents() {  Collections.sort(subjectData.getTopicData(),  Comparator.comparing(SubjectData.TopicData::getAmount));  }  @Override  public String getName() {  return subjectData.getName();  }  @Override public int topicsCount() {  return subjectData.getTopicData().size();  }  @Override  public AbstractTopic[] getTopics() {  AbstractTopic[] topics = new AbstractTopic[topicsCount()];  for (int i = 0; i < topicsCount(); i++) {  topics[i] = new XMLTopic(subjectData.getTopicData().get(i));  }  return topics;  }  @Override  public AbstractTopic getTopic(int i) {  return new XMLTopic(subjectData.getTopicData().get(i));  }  @Override  public boolean getExams() {  return subjectData.isExams();  }  @Override  public void setExams(boolean b) {  subjectData.setExams(b);  }  @Override  public void setTopics(AbstractTopic[] topics) {  ClearTopics();  for (AbstractTopic topic : topics) {  addTopic(topic);  }  }  @Override  public boolean addTopic(AbstractTopic topic) {  SubjectData.TopicData topicData = new SubjectData.TopicData();  boolean result = subjectData.getTopicData().add(topicData);  setTopic(topicsCount() - 1, topic);  return result;  }  @Override  public boolean addTopic(String name, int amount, String date) {  SubjectData.TopicData topicData = new SubjectData.TopicData();  topicData.setName(name);  topicData.setAmount(amount);  topicData.setDate(date);  return subjectData.getTopicData().add(topicData);  }  @Override  public void setTopic(int i, AbstractTopic topic) {  subjectData.getTopicData().get(i).setAmount(topic.getAmount());  subjectData.getTopicData().get(i).setDate(topic.getDate());  subjectData.getTopicData().get(i).setName(topic.getName());  }  public static void main(String[] args) throws Exception {  XMLSubject subject = new XMLSubject();  try {  subject.readFromFile("src/task\_1/xml/OOP.xml");  subject.testSubject();  subject.writeToFile("src/task\_1/xml/ByNames.xml");  }  catch (FileNotFoundException e) {  System.out.println("Read failed");  e.printStackTrace();  }  catch (IOException e) {  System.out.println("Write failed");  e.printStackTrace();  }  catch (JAXBException e) {  e.printStackTrace();  System.out.println("Wrong format");  }  }  } |

*Вміст файлу SubjectFX.java*

|  |
| --- |
| package task\_1.FX;  import javafx.application.Application;  import javafx.fxml.FXMLLoader;  import javafx.scene.Scene;  import javafx.scene.layout.BorderPane;  import javafx.stage.Stage;  public class SubjectsFX extends Application {  @Override  public void start(Stage primaryStage) {  try {  BorderPane root = (BorderPane)FXMLLoader.load(getClass().getResource("SubjectsForm.fxml"));  Scene scene = new Scene(root, 700, 500);  primaryStage.setScene(scene);  primaryStage.setTitle("Учебные курсы");  primaryStage.show();  }  catch(Exception e) {  e.printStackTrace();  }  }  public static void main(String[] args) {  launch(args);  }  } |

*Вміст файлу SubjectController.java*

|  |
| --- |
| package task\_1.FX;  import javafx.application.Platform;  import javafx.collections.FXCollections;  import javafx.collections.ObservableList;  import javafx.event.ActionEvent;  import javafx.fxml.FXML;  import javafx.fxml.Initializable;  import javafx.scene.control.\*;  import javafx.scene.control.Alert.AlertType;  import javafx.scene.control.TableColumn.CellEditEvent;  import javafx.scene.control.cell.PropertyValueFactory;  import javafx.scene.control.cell.TextFieldTableCell;  import javafx.stage.FileChooser;  import javafx.util.converter.IntegerStringConverter;  import task\_1.xml.XMLSubject;  import task\_1.AbstractTopic;  import javax.xml.bind.JAXBException;  import java.io.File;  import java.io.IOException;  import java.net.URL;  import java.util.ArrayList;  import java.util.List;  import java.util.ResourceBundle;  public class SubjectsController implements Initializable {  private XMLSubject subject = new XMLSubject();  private ObservableList<AbstractTopic> observableList;    public static FileChooser getFileChooser(String title) {  FileChooser fileChooser = new FileChooser();  // Починаємо шукати з поточної теки:  fileChooser.setInitialDirectory(new File("src/task\_1/FX"));  // Встановлюємо фільтри для пошуку файлів:  fileChooser.getExtensionFilters().add(  new FileChooser.ExtensionFilter("XML-файли (\*.xml)", "\*.xml"));  fileChooser.getExtensionFilters().add(  new FileChooser.ExtensionFilter("Лалалэй (\*.\*)", "\*.\*"));  // Вказуемо заголовк вікна:  fileChooser.setTitle(title);  return fileChooser;  }    @FXML TextField textFieldSubject;  @FXML TextField textFieldText;  @FXML TextArea textAreaResults;    @FXML TableView<AbstractTopic> tableViewTopics;  @FXML TableColumn<AbstractTopic, String> tableColumnName;  @FXML TableColumn<AbstractTopic, Integer> tableColumnAmount;  @FXML TableColumn<AbstractTopic, String> tableColumnDate;      @Override  public void initialize(URL location, ResourceBundle resources) {  // TODO Auto-generated method stub  tableViewTopics.setPlaceholder(new Label(""));  }    @FXML public void nameChanged(ActionEvent event) {  // Коли користувач змінив дані в textFieldCountry,  // автоматично оновлюємо назву:  subject.setName(textFieldSubject.getText());  }    @FXML public void doSearchByWord(ActionEvent event) {  // Оновлюємо дані:  updateSourceData();  textAreaResults.setText("");  for (int i = 0; i < subject.topicsCount(); i++) {  AbstractTopic topic = subject.getTopic(i);  if (topic.containsWord(textFieldText.getText())) {  showResults(topic);  }  }  }  @FXML public void doSearchBySubstring(ActionEvent event) {  // Оновлюємо дані:  updateSourceData();  textAreaResults.setText("");  for (int i = 0; i < subject.topicsCount(); i++) {  AbstractTopic topic = subject.getTopic(i);  if (topic.getName().toLowerCase().contains(textFieldText.getText().toLowerCase())) {  showResults(topic);  }  }  }  @FXML public void doAbout(ActionEvent event) {  Alert alert = new Alert(AlertType.INFORMATION);  alert.setTitle("О программе...");  alert.setHeaderText("Данные об учебном курсе");  alert.setContentText("Версія 1.0");  alert.showAndWait();  }  private void showResults(AbstractTopic topic) {  textAreaResults.appendText("Тема: " + topic.getName() + "\n");  textAreaResults.appendText("Студентов: " + topic.getAmount() + "\n");  textAreaResults.appendText("Лекция за: " + topic.getDate() + "\n");  }  @FXML public void doOpen(ActionEvent event) {  FileChooser fileChooser = getFileChooser("Открыть XML-файл");  File file;  if ((file = fileChooser.showOpenDialog(null)) != null) {  try {  subject.readFromFile(file.getCanonicalPath());  // Заповнюємо текстові поля прочитаними даними:  textFieldSubject.setText(subject.getName());  //textField.setText(country.getArea() + "");  textAreaResults.setText("");  // Очищаємо та оновлюємо таблицю:  tableViewTopics.setItems(null);  updateTable();  }  catch (IOException e) {  showError("Файл не знайдено");  }  catch (JAXBException e) {  e.printStackTrace();  showError("Неправильний формат файлу");  }  }  }  public static void showError(String message) {  Alert alert = new Alert(AlertType.ERROR);  alert.setTitle(":(");  alert.setHeaderText(message);  alert.showAndWait();  }  private void updateSourceData() {  // Переписуємо дані в модель з observableList  subject.ClearTopics();  subject.setName(textFieldSubject.getText());  for (AbstractTopic topic : observableList) {  subject.addTopic(topic);  }  }  @FXML public void doNew(ActionEvent event) {  subject = new XMLSubject();  observableList = null;  textFieldSubject.setText("");  textFieldText.setText("");  textAreaResults.setText("");  tableViewTopics.setItems(null);  tableViewTopics.setPlaceholder(new Label(""));  }  @FXML  public void doSave(ActionEvent event) {  FileChooser fileChooser = getFileChooser("Сохранить XML-файл");  File file;  if ((file = fileChooser.showSaveDialog(null)) != null) {  try {  updateSourceData(); // оновлюємо дані в моделі  subject.writeToFile(file.getCanonicalPath());  showMessage("Сохранено");  }  catch (Exception e) {  showError("Помилка запису в файл");  }  }  }  public static void showMessage(String message) {  Alert alert = new Alert(AlertType.INFORMATION);  alert.setTitle("");  alert.setHeaderText(message);  alert.showAndWait();  }  @FXML public void doSortByStudentsAmount(ActionEvent event) {  updateSourceData();  subject.sortByStudents();  updateTable();  }  @FXML public void doSortByName(ActionEvent event) {  updateSourceData();  subject.sortByTopicLength();  updateTable();  }  @FXML public void doRemove(ActionEvent event) {  // Не можемо видалити рядок, якщо немає даних:  if (observableList == null) {  return;  }  // Якщо є рядки, видаляємо останній:  if (observableList.size() > 0) {  observableList.remove(observableList.size() - 1);  }  // Якщо немає рядків, вказуємо, що дані відсутні:  if (observableList.size() <= 0) {  observableList = null;  }  }  @FXML  public void doExit(ActionEvent event) {  Platform.exit(); // коректне завершення застосунку JavaFX  }  @FXML public void doAdd(ActionEvent event) {  subject.addTopic("",0,"");  updateTable();  }    private void updateTable() {  // Заповнюємо observableList:  List<AbstractTopic> list = new ArrayList<>();  for (int i = 0; i < subject.topicsCount(); i++) {  list.add(subject.getTopic(i));  }  observableList = FXCollections.observableList(list);  tableViewTopics.setItems(observableList);    // Вказуємо для колонок зв'язану з ними властивість і механізм редагування  // залежно від типу комірок:    tableColumnDate.setCellValueFactory(new PropertyValueFactory<>("Date"));  tableColumnDate.setCellFactory(TextFieldTableCell.forTableColumn());  tableColumnDate.setOnEditCommit(t -> updateDate(t));    tableColumnName.setCellValueFactory(new PropertyValueFactory<>("Name"));  tableColumnName.setCellFactory(TextFieldTableCell.forTableColumn());  tableColumnName.setOnEditCommit(t -> updateName(t));    tableColumnAmount.setCellValueFactory(new PropertyValueFactory<>("Amount"));  tableColumnAmount.setCellFactory(TextFieldTableCell.forTableColumn(new IntegerStringConverter()));  tableColumnAmount.setOnEditCommit(t -> updateAmount(t));  }    private void updateDate(CellEditEvent<AbstractTopic, String> t) {  // Оновлюємо коментарі:  AbstractTopic at = t.getTableView().getItems().get(t.getTablePosition().getRow());  at.setDate(t.getNewValue());  }  private void updateName(CellEditEvent<AbstractTopic, String> t) {  // Оновлюємо коментарі:  AbstractTopic at = t.getTableView().getItems().get(t.getTablePosition().getRow());  at.setName(t.getNewValue());  }  private void updateAmount(CellEditEvent<AbstractTopic, Integer> t) {  // Оновлюємо коментарі:  AbstractTopic at = t.getTableView().getItems().get(t.getTablePosition().getRow());  at.setAmount(t.getNewValue());  }    } |

*Вміст файлу OOP.xml (застосовувався для прикладу)*

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <SubjectData Name="OOP" Exams="false" >  <TopicData Date="21/9/2018" Name="Generics" Amount="8" />  <TopicData Date="2/10/2018" Name="Exceptions" Amount="3" />  <TopicData Date="16/10/2018" Name="GUI" Amount="8" />  <TopicData Date="30/10/2018" Name="Reflection" Amount="7" />  </SubjectData> |

*Приклад виконання програми*

|  |
| --- |
|  |

### Завдання 2. Перелік для опису місяців року

*Код програми*

|  |
| --- |
| package task\_2;  import task\_2.task\_2.Months;  public class task\_2 {   enum Months {  JAN("Январь", 31), FEB("Февраль", 28), MAR("Март", 31),  APR("Апрель", 30), MAY("Май", 31), JUN("Июнь", 30),  JUL("Июль", 31), AUG("Август", 31), SEP("Сентябрь", 30),  OCT("Октябрь", 31), NOV("Ноябрь", 30), DEC("Декабрь", 31);    private int days;  private String name;    public String getSeason() {   switch(this.ordinal()) {  case 0:  case 1:  case 11: return "Зима";  case 2:  case 3:  case 4: return "Весна";  case 5:  case 6:  case 7: return "Лето";   case 8:  case 9:  case 10: return "Осень";  default: return "This is Sparta!";  }  }  @Override  public String toString() {  return this.name + " " + this.days + " " + this.getSeason();    }  private Months(String name, int days){  this.name = name;  this.days = days;  }  Months next() {  return values()[(ordinal()+1) % values().length];  }  Months prev() {  return values()[((ordinal()-1)+values().length) % values().length];  }  }  public static void main(String[] args) {  Months jan = Months.JAN;  System.out.println(jan.next() + "/ " + jan.prev());  Months dec = Months.DEC;  System.out.println(dec.next() + "/ " + dec.prev());  }  } |

*Приклад виконання програми*

|  |
| --- |
| Февраль 28 Зима/ Декабрь 31 Зима Январь 31 Зима/ Ноябрь 30 Осень |

### Завдання 3. Дані про користувачів

*Код програми*

|  |
| --- |
| package task\_3;  import java.util.Map;  import java.util.SortedMap;  import java.util.TreeMap;  public class task\_3 {  public static void main (String[] args) {  SortedMap<String, String> users = new TreeMap<>();  users.put("Mike", "1231sdq3123");  users.put("Sean", "412aw");  users.put("Michael", "02-2r23");    for(Map.Entry<?, String> entry : users.entrySet()) {  if (entry.getValue().length()>6) {  System.out.println(entry.getKey() + " " + entry.getValue());  }  }  }    } |

*Приклад виконання програми*

|  |
| --- |
| Michael 02-2r23  Mike 1231sdq3123 |

### Завдання 4. Міні-калькулятор

*Код програми*

|  |
| --- |
| package task\_4;  import javafx.application.Application;  import javafx.event.Event;  import javafx.geometry.Pos;  import javafx.scene.Scene;  import javafx.scene.control.Alert;  import javafx.scene.control.Alert.AlertType;  import javafx.scene.layout.FlowPane;  import javafx.scene.control.Button;  import javafx.scene.control.TextField;  import javafx.stage.Stage;  import javafx.scene.control.RadioButton;  import javafx.scene.control.ToggleGroup;  public class task\_4 extends Application {  private Button button;  private TextField field1, field2, field3;    enum Command{  SUM, DIF, MUL, DIV;  }  private Command command;    @Override  public void start(Stage stage) throws Exception {  stage.setTitle("The World's Greates Calculator");  FlowPane rootNode = new FlowPane(10,10);    RadioButton radioButtonFirst = new RadioButton("Сложить");  radioButtonFirst.setOnAction(ev -> command = Command.SUM);    RadioButton radioButtonSecond = new RadioButton("Вычесть");  radioButtonSecond.setOnAction(ev -> command = Command.DIF);    RadioButton radioButtonThird = new RadioButton("Умножить");  radioButtonThird.setOnAction(ev -> command = Command.MUL);    RadioButton radioButtonFourth = new RadioButton("Поделить");  radioButtonFourth.setOnAction(ev -> command = Command.DIV);    ToggleGroup radioGroup = new ToggleGroup();  radioButtonFirst.setToggleGroup(radioGroup);  radioButtonSecond.setToggleGroup(radioGroup);  radioButtonThird.setToggleGroup(radioGroup);  radioButtonFourth.setToggleGroup(radioGroup);    rootNode.setAlignment(Pos.CENTER);  Scene scene = new Scene(rootNode, 200, 200);  stage.setScene(scene);  button = new Button("Ответ");  button.setOnAction(this::buttonClick);  field1 = new TextField();  field2 = new TextField();  field3 = new TextField();  rootNode.getChildren().addAll(field1, field2, button, radioButtonFirst, radioButtonSecond, radioButtonThird, radioButtonFourth, field3 );  stage.show();  }  private void buttonClick(Event event) {  try {  int i = Integer.parseInt(field1.getText());  int j = Integer.parseInt(field2.getText());  int k = 777;  switch(command) {  case SUM:  k = i+j;  break;  case DIF:  k = i-j;  break;  case MUL:  k = i\*j;  break;  case DIV:  k = i/j;  break;  }  field3.setText(k + "");  }  catch (NumberFormatException e1) {  Alert alert = new Alert(AlertType.ERROR);  alert.setTitle("Помилка");  alert.setHeaderText("Хибні дані!");  alert.showAndWait();  }  }    public static void main(String[] args) {  launch(args);  }  } |

*Приклад виконання програми*

|  |
| --- |
|  |

### Завдання 5. Словник (додаткове завдання)

*Код програми*

|  |
| --- |
| package task\_5;  import java.io.BufferedReader;  import java.io.BufferedWriter;  import java.io.File;  import java.io.FileReader;  import java.io.FileWriter;  import java.io.IOException;  import java.io.PrintWriter;  import java.util.Scanner;  import java.util.StringTokenizer;  import javafx.application.Application;  import javafx.event.ActionEvent;  import javafx.event.Event;  import javafx.geometry.Pos;  import javafx.scene.Scene;  import javafx.scene.control.Alert;  import javafx.scene.control.Button;  import javafx.scene.control.Label;  import javafx.scene.control.TextField;  import javafx.scene.input.KeyCode;  import javafx.scene.control.Alert.AlertType;  import javafx.scene.layout.FlowPane;  import javafx.scene.layout.HBox;  import javafx.scene.layout.VBox;  import javafx.stage.FileChooser;  import javafx.stage.Stage;  import java.util.Map;  import java.util.TreeMap;  public class task\_5 extends Application {    private boolean RE = false;  private double a, b;  private FileChooser chooser;  private File file;  private Map<String, String> dict;    private TextField srcTextField = new TextField();  private TextField newWord = new TextField();    private Label desText = new Label();  private Label lbl1 = new Label("English");  private Label lbl2 = new Label("Russian");  private void addNew() {    String key, value;  if (RE) {  key = newWord.getText();  value = srcTextField.getText();  }  else {  key = srcTextField.getText();  value = newWord.getText();  }  dict.put(key, value);  try {  BufferedWriter bw = new BufferedWriter(new FileWriter("src/task\_5/Dictionary.txt", true));  bw.newLine();  bw.write(key + " " + value);  bw.close();  newWord.setVisible(false);  desText.setText("Добавлено слово " + key);  } catch (IOException e) {  e.printStackTrace();  }  }  private void translate(Event event) {  String src = srcTextField.getText();  if (RE) {  if (dict.containsValue(src)) {  newWord.setVisible(false);  desText.setText(dict.get(src));  }  }  else {  if (dict.containsKey(src)) {  newWord.setVisible(false);  desText.setText(dict.get(src));  }  else  {  desText.setText("Слово не найдено. Можно добавить в словарь ниже");  newWord.setVisible(true);  }  }    }    @Override  public void start(Stage stage) throws Exception {      newWord.setVisible(false);  newWord.setOnKeyPressed(event -> {if (event.getCode() == KeyCode.ENTER) addNew();});  stage.setTitle("The World's Greatest Dictionary");  HBox rootNode = new HBox();  VBox vBoxSrc = new VBox();  VBox vBoxDes = new VBox();  //VBox vBoxButtons = new VBox();    Button button1 = new Button("Translate");  //button1.setAlignment(Pos.BOTTOM\_LEFT);  button1.setOnAction(this::translate);  Button button2 = new Button("<>");    vBoxSrc.getChildren().addAll(lbl1, srcTextField, button1);  vBoxDes.getChildren().addAll(lbl2, desText, newWord);  // vBoxButtons.getChildren().addAll(button1, button2);  rootNode.getChildren().addAll(vBoxSrc, vBoxDes);      Scene scene = new Scene(rootNode, 600, 100);  stage.setScene(scene);  rootNode.setAlignment(Pos.CENTER);      stage.show();    dict = new TreeMap<>();  try (BufferedReader br = new BufferedReader(new FileReader("src/task\_5/dictionary.txt"))){  Scanner scanner = new Scanner(br);  while (scanner.hasNextLine()){  StringTokenizer st = new StringTokenizer(scanner.nextLine());  dict.put(st.nextToken(), st.nextToken());  }  }  for (Map.Entry<String, String> entry :dict.entrySet()) {  System.out.println(entry.getValue() + " " + entry.getKey());  }  }  public static void main(String[] args) {  launch(args);  }  } |

*Вміст Dictionary.txt*

|  |
| --- |
| pineapple ананас  pear груша  apple яблоко  peach персик  orange апельсин |

*Приклад виконання програми*

|  |
| --- |
|  |

Висновки

Бібліотека javax.swing пропонує розробникові низку стандартних класів, які можна використовувати для проектування графічного інтерфейсу користувача. Ця бібліотека розширила попередню менш вдалу бібліотеку AWT (Abstract Window Toolkit) засоби якої використовує для обробки подій, роботи з графікою тощо.

На відміну від компонентів AWT, компоненти Swing є "легковагими" (lightweight). Це означає, що компоненти Swing використовують засоби Java для відображення елементів графічного інтерфейсу користувача на поверхні вікна, без використання компонентів операційної системи.

Перелік - список можливих значень, які може отримувати змінна цього типу. У найпростішій своїй формі переліки Java аналогічні відповідним конструкціям C++ і C#. Переліки представлять собою класи, об'єкти яких є заздалегіть визначеними.