Министерство цифрового развития, связи и массовых коммуникаций Российской Федерации Ордена Трудового Красного Знамени  
Федеральное государственное бюджетное образовательное учреждение высшего образования   
«Московский технический университет связи и информатики»

Кафедра математическая кибернетика и информационные технологии

Курсовая работа

По Java-программированию

На тему «Разработка Таск-Менеджера»

Выполнили: студент гр. БСУ1901:

Тусеев В.Н.

Проверила: преп. Мосева М.С.

Москва 2021г.

Оглавление

Введение3

Техническое задание на разработку3

Описание программной реализации8

Описание процесса тестирования9

Руководство пользователя10

Заключение11

**Целью данной курсовой работы** является создание, так называемого, Таск-Менеджера.

**Техническое задание на разработку**

**Техническое задание**: разработать приложение, анимацию, игру или другое приложение для демонстрации своих знаний об основах программирования на Java.

Требования к разрабатываемому приложению:

* включать хотя бы 2 класса, не считая тестовый класс;
* инкапсулировать эти и дополнительные классы;
* предоставлять хотя бы 2 поля и 2 метода в каждом классе, не включая методы получения или установки;
* включать массив или список ArrayList;
* быть способной принять ввод пользователя хотя бы один раз; всегда принимать ввод пользователя без сбоя.

**Выполнение**

Метод createAndShowGUI() отвечает за формирование исходной формы приложения (установка кнопок, часов и различных панелей для отображения целей и задач).

public static void createAndShowGUI(){  
  
 Instruction();  
  
 mainForm = new JFrame("Курсовая");  
 mainForm.setSize(1440, 900);  
 mainForm.setLayout(new BorderLayout());  
 mainForm.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  
 mainForm.setLocation(200,20);  
 mainForm.setVisible(true);  
  
 //Установка верхне панели  
 upLine = new JPanel();  
 upLine.setLayout(new BorderLayout());  
 JLabel list = new JLabel("Перечень задач и целей");  
 list.setHorizontalAlignment(SwingConstants.CENTER);  
 list.setVerticalAlignment(SwingConstants.CENTER);  
 upLine.add(list);  
 mainForm.add(upLine, BorderLayout.NORTH);  
  
 midScroll = new JScrollPane();  
 midlLine = new JPanel();  
 midlLine.setLayout(new BoxLayout(midlLine,BoxLayout.Y\_AXIS));  
 midlLine.setVisible(true);  
 *midScroll* = new JScrollPane(*midlLine*);  
 *midScroll*.setVisible(true);  
 *mainForm*.add(*midScroll*);  
  
 //Установка нижней панели  
 *lowLine* = new JPanel();  
 *lowLine*.setLayout(new BorderLayout());  
 JButton ButtonAddNew = new JButton("Добавить цель");  
 JButton ButtonLoadNew = new JButton("Удалить выполненные");  
  
 Date d = new Date();  
 LabelData = new JLabel(d.toString());  
 lowLine.add(ButtonAddNew,BorderLayout.CENTER);  
 lowLine.add(LabelData,BorderLayout.EAST);  
 lowLine.add(ButtonLoadNew,BorderLayout.WEST);  
 mainForm.add(lowLine, BorderLayout.SOUTH);  
  
 //Включение кнопок  
 ButtonAddNew.addActionListener(handler);  
 ButtonLoadNew.addActionListener(handler);  
  
 //Установка часов  
 TimerTask timerTask = new RepeatedTime();  
 Timer timer = new Timer(true);  
  
 try{  
 while (mainForm.isActive()){  
 timer.scheduleAtFixedRate(timerTask,0,1000);  
 }  
 }catch (IllegalStateException e) {  
 System.out.println("");  
 }  
}

Здесь же вызываем метод Instruction(), который выведет нам окошко с инструкциями к данной работе, а также предоставит кнопки для загрузки данных в файл и выгрузке их из самого файла.

Создадим класс для работы с таймером, который необходим для работы времени.

//работа с таймером (его обновление)  
public static class RepeatedTime extends TimerTask{  
 public void run(){  
 Date dataTime = new Date();  
 *LabelData*.setText(dataTime.toString());  
 *LabelData*.updateUI();  
 }  
}

Данный класс «наблюдает» за нажатыми кнопками и вызывает соответствующие действия для этих кнопок:

public static class ButtonHandler implements ActionListener{  
 public void actionPerformed(ActionEvent e) {  
 String command = e.getActionCommand();  
 if (command.equals("Добавить цель")){  
 *ButtonAdd*();  
 } else if (command.equals("Принять")) {  
 *ButtonNewPurpose*();  
 *GiveField*.setText("");  
 } else if (command.equals("Загрузить в файл")){  
 try {  
 *LoadToFile*();  
 *forFile* = "";  
 } catch (IOException ex) {  
 ex.printStackTrace();  
 }  
 } else if (command.equals("Очистить файл")){  
 try {  
 if (*file*.length() != 0)  
 *clearFile*();  
 *kolPurpose* = 0;  
 } catch (IOException ex) {  
 ex.printStackTrace();  
 }  
 } else if (command.equals("Удалить выполненные")){  
 if(!*arrCheckPanel*.isEmpty() && !*ArrCheckBox*.isEmpty()) {  
 try {  
 *check*();  
 } catch (NullPointerException ex) {  
 ex.printStackTrace();  
 }  
 }  
 } else if (command.equals("Загрузить из файла")){  
 try {  
 *ReadTextFile*();  
 } catch (IOException ex) {  
 ex.printStackTrace();  
 }  
 }  
  
 }  
}

Метод ButtonAdd() отвечает за формирование нового окна, в которое будет записываться новая цель.

//Кнопка для открытия доп окна для ввода задачи  
public static void ButtonAdd(){  
 //Новое окно для добавления задачи  
 *FrameAdd* = new JFrame();  
 *FrameAdd*.setSize(375,205);  
 *FrameAdd*.setLayout(null);  
 *FrameAdd*.setLocation(400,400);  
 *FrameAdd*.setVisible(true);  
  
 *GiveField* = new TextArea();  
 *FrameAdd*.add(*GiveField*);  
 *GiveField*.setSize(200,100);  
 *GiveField*.setLocation(20,25);  
  
 JButton newPurpose = new JButton("Принять");  
 newPurpose.setSize(100,40);  
 *FrameAdd*.add(newPurpose);  
 newPurpose.addActionListener(*handler*);  
 newPurpose.setLocation(240,35);  
  
 JButton ButtonLoadToFile = new JButton("Загрузить в файл");  
 ButtonLoadToFile.addActionListener(*handler*);  
 ButtonLoadToFile.setSize(100,40);  
 *FrameAdd*.add(ButtonLoadToFile);  
 ButtonLoadToFile.setLocation(240,75);  
}

Метод ButtonNewPurpose() добавляет новую задачу в основное окно.

//Кнопка при добавлении задачи в основное окно  
public static void ButtonNewPurpose(){  
  
 JPanel newPurpose = new JPanel();  
 newPurpose.setLayout(new BoxLayout(newPurpose,BoxLayout.*X\_AXIS*));  
 newPurpose.setVisible(true);  
  
 //Установка списка  
 JComboBox<Object> importance = new JComboBox<>();  
 importance.addItem("Важно");  
 importance.addItem("Не очень важно");  
 importance.addItem("Можно отложить");  
 newPurpose.add(importance);  
  
 //Установка текста самой задачи  
 *forFile* += Integer.*toString*(*kolPurpose*) + ".) " + *GiveField*.getText();  
 TextArea area = new TextArea(*forFile*);  
 area.setSize(500,30);  
 area.setLocation(100,30);  
 newPurpose.add(area);  
 newPurpose.setSize(200,800);  
  
 //Установка CheckBox  
 *checkBox* = new JCheckBox(Integer.*toString*(*kolPurpose*));  
 newPurpose.add(*checkBox*);  
 *checkBox*.setLocation(650,30);  
  
 *midlLine*.add(newPurpose);  
  
 *arrCheckPanel*.put(*checkBox*, newPurpose);  
 *ArrCheckBox*.put(*kolPurpose*,*checkBox*);  
  
 *kolPurpose*++;  
  
}

Метод LoadToFile() отвечает за загрузку всех задач в текстовый файл.

//Кнопка для загрузки в файл  
public static void LoadToFile() throws IOException {  
 BufferedWriter bf = new BufferedWriter(new FileWriter(*file*,true));  
 bf.write(*forFile*+"\n");  
 bf.close();  
}

Метод clearFile() отвечает за очистку файла от всех задач

//Кнопка очистки файла  
public static void clearFile() throws IOException {  
 FileWriter fl = new FileWriter(*file*);  
 fl.write("");  
 fl.close();  
}

Метод ReadTextFile() отвечает за загрузку задач из файла в основное окно

//Загрузка из файла  
public static void ReadTextFile() throws IOException {  
 FileReader fr = new FileReader(*file*);  
 BufferedReader reader = new BufferedReader(fr);  
 String line = reader.readLine();  
 while (line != null){  
  
 JPanel rr = new JPanel();  
 rr.setLayout(new BoxLayout(rr, BoxLayout.*X\_AXIS*));  
 rr.setVisible(true);  
  
 JComboBox<Object> im = new JComboBox<>();  
 im.addItem("Важно");  
 im.addItem("Не очень важно");  
 im.addItem("Можно отложить");  
 rr.add(im);  
  
  
 TextArea ta = new TextArea(line);  
 rr.add(ta);  
  
 JCheckBox cb = new JCheckBox(Integer.*toString*(*kolPurpose*));  
 rr.add(cb);  
  
 *midlLine*.add(rr);  
  
 *arrCheckPanel*.put(*checkBox*, rr);  
 *ArrCheckBox*.put(*kolPurpose*,*checkBox*);  
  
 *kolPurpose*++;  
 line = reader.readLine();  
 }  
 reader.close();  
 fr.close();  
   
}

Метод Instruction() отвечает за отображение инструкции

//Вывод инструкции  
public static void Instruction(){  
 JFrame instrict = new JFrame();  
 instrict.setSize(345,200);  
 instrict.setVisible(true);  
 instrict.setLayout(new BorderLayout());  
  
 JButton button = new JButton("Очистить файл");  
 button.setSize(150,30);  
 button.addActionListener(*handler*);  
 instrict.add(button,BorderLayout.*SOUTH*);  
  
 JButton butLoad = new JButton("Загрузить из файла");  
 butLoad.setSize(150,30);  
 butLoad.addActionListener(*handler*);  
 instrict.add(butLoad,BorderLayout.*NORTH*);  
  
 String StrInstr = "Для того, чтобы очистить файл, нажмите кнопку;" + "\n" +  
 "Для того, чтобы загрузить цель в фал, сначала" + "\n" + "нажмите Принять (задача добавится в приложение)," +  
 "\n"+"а после нажмите кнопку Загрузить;";  
 TextArea ins = new TextArea(StrInstr);  
 instrict.add(ins,BorderLayout.*CENTER*);  
 ins.setText(StrInstr);  
 ins.setVisible(true);  
  
}

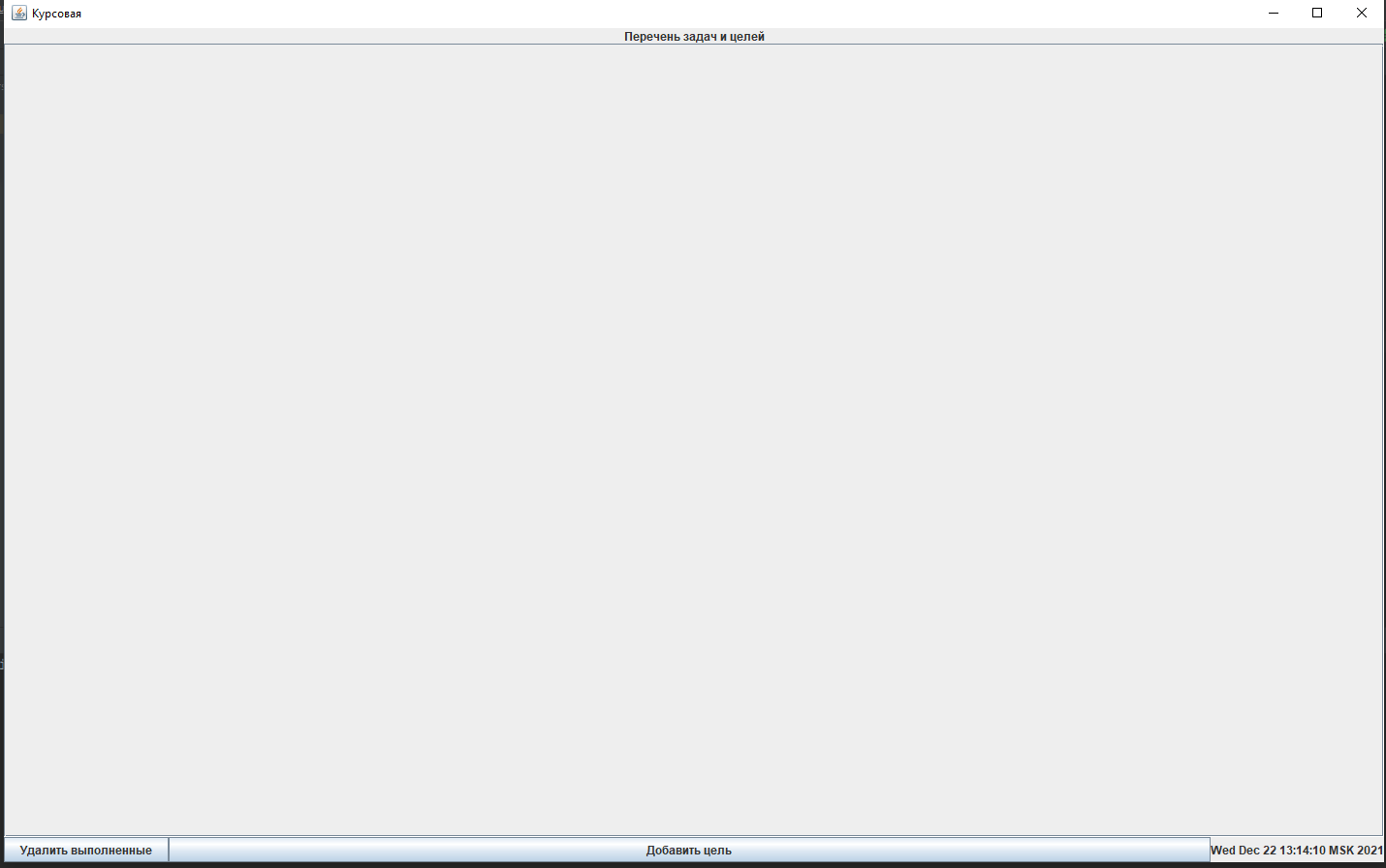


Рис.1 – Форма для вывода целей и задач

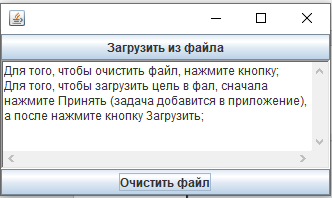


Рис.2 – Окно с инструкцией

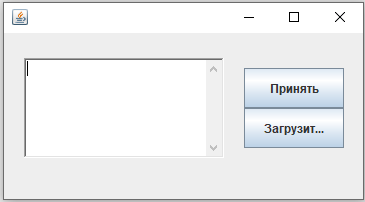


Рис.3 – Окно с добавлением задачи

**Руководство пользователя**

1. Для запуска необходимо предварительно скачать файлы и установить Java и среду разработки на свой ПК.

2. Запускаем приложение “IntelliJ IDEA”, затем открываем наш проект с помощью кнопок “File”и “Open”, далее вводим путь к папке с нашей игрой.

**Список использованных источников**

1. Герберт, Шилдт Java. Руководство для начинающих / Шилдт Герберт. - М.: Диалектика / Вильямс, 2014.

2. Давыдов, Станислав IntelliJ IDEA. Профессиональное программирование на Java / Станислав Давыдов , Алексей Ефимов. - М.: БХВ-Петербург, 2015.

Ссылка на проект (GitHub): https://github.com/TuzZz1/kursovayaJava.git