**Федеральное государственное бюджетное образовательное учреждение**

**высшего профессионального образования**

**Уфимский государственный авиационный технический университет**

**Кафедра вычислительной математики и кибернетики**

**Расчетно-графическая работа по курсу**

**«Системы реального времени»**

Вариант 3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Ф.И.О. | Дата | Оценка | Подпись |
| Выполнил студент группы ПРО-301в | Доронин С.Г. |  |  |  |
| Принял: | Ахуньянова К.Б. |  |  |  |

Уфа – 2014 г.

# Задание на расчетно-графическую работу

Разработать программу, которая для решения одной из подзадач вызывает консольное приложение. Необходимо считывать информацию, которую вызываемое (дочернее) приложение выводит в консоль, и показывать ее в окне нашего приложения Т.е. выдает и принимает данные дочернее приложение, а реально они отображаются на экран и вводятся с клавиатуры нашим, главным приложением – как будто всю работу делает главное приложение.

# Содержание

[Введение 4](#_Toc389582689)

[Этапы решения и их взаимосвязь 4](#_Toc389582690)

[Руководство пользователя 5](#_Toc389582691)

[Результаты 6](#_Toc389582692)

[Текст программы 7](#_Toc389582693)

[Запускающий класс 7](#_Toc389582694)

[Главная форма 7](#_Toc389582695)

[Класс запускающий внешние программы 8](#_Toc389582696)

[Текст используемых источников 10](#_Toc389582697)

# Введение

Часто бывают случаи когда есть хорошее консольное приложение и для него нужно сделать графический интерфейс.

Необходимо считывать информацию, которую вызываемое (дочернее) приложение выводит в консоль, и показывать ее в окне нашего приложения. И наоборот, когда дочернему приложению нужен ввод с клавиатуры, мы должны вводить текстовую строку в главном приложении, и эта строка будет отсылаться приложению дочернему. Т.е. выдает и принимает данные дочернее приложение, а реально они отображаются на экран и вводятся с клавиатуры нашим, главным приложением – как будто всю работу делает главное приложение.

Для решения задачи следует использовать механизм пайпов (pipes): в открытый пайп можно писать, как в любой другой поток вывода с последовательным доступом, или читать из него (подробнее см. в методических указаниях) Поскольку рассматриваемый тип консольных приложений использует стандартные потоки ввода-вывода именно для последовательного доступа (поток ввода – для чтения с клавиатуры, поток вывода – для вывода в окно), пайпом легко подменить стандартные потоки ввода и вывода консольного приложения (т.е., соответственно, клавиатуру и окно консоли). Сделать это можно при запуске дочернего процесса (функция CreateProcess).

# Этапы решения и их взаимосвязь

Для решения этой задачи были созданы 3 класса:

1. Main – класс для запуска программы
2. MainForm – главная форма
3. ProgramRunner – класс для запуска внешних программ

При запуске программы создается форма «**MainForm**». При введении команды в поле ввода и нажатия кнопки «**Run**», происходит запуск внешней программы с помощью класса «**ProgramRunner**». Данные вывода, запускаемой программы, считываются и выводится в правой части формы.

# Руководство пользователя

Программа запускается с файла **Cmd-GUI.jar** или с файла сценария **run.bat.** Для работы приложения требуется установленная JRE 1.6 или новее (Java Runtime Environment).

После запуска программы появится окно, имеющее вид (Рисунок 1).

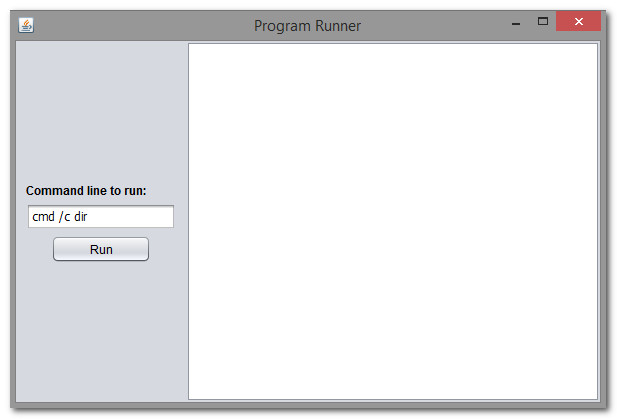


Рисунок 1. Главное окно программы

Сразу после запуска программы создается форма «**MainForm**».

Чтобы запустить внешнюю программу, нужно ввести команду в поле «**Command line to run**» и нажать на кнопку «**Run**».

После этого будет запущено другое приложение и результаты вывода этого приложения будут отображены в правой части формы (Рисунок 2).

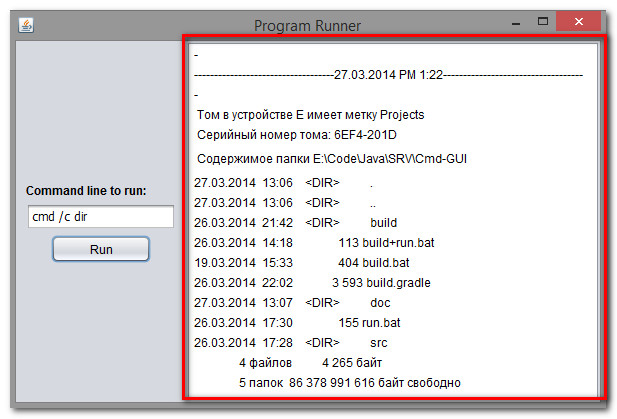


Рисунок 2. Отображение вывода запускаемой программы

При запуске другой команды, ее вывод добавится ниже с указанием времени выполнения команды (Рисунок 3).

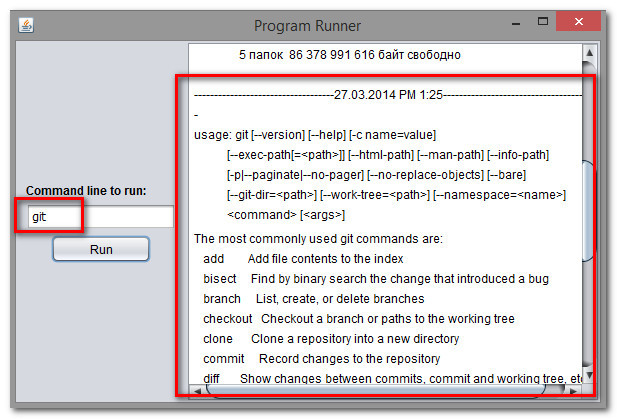


Рисунок 3. Добавленный вывод от другой программы

# Результаты

В результате выполнения работы, была создана программа, которая может запускать другие программы и отображать их вывод (графический интерфейс для консольных приложений).

# Текст программы

## Запускающий класс

/\*\*

\* Запускающий класс

\*/

public **class** Main {

/\*\* Запуск программы \*/

public static void main(String[] args) {

// Установка темы оформления

**try** {

UIManager.setLookAndFeel("javax.swing.plaf.nimbus.NimbusLookAndFeel");

} **catch** (Exception e1) {

e1.printStackTrace();

}

// Создание и запуск главной формы

**new** MainForm();

}

/\*\* Выводит установленные LookAndFeels \*/

**private** static void printInstalledLookAndFeels() {

**for** (UIManager.LookAndFeelInfo info : UIManager.getInstalledLookAndFeels()) {

System.out.println(info.getName() **+** ": " **+** info.getClassName());

}

}

/\*\* Системный LookAndFeel \*/

**private** static String getSystemLookAndFeel() {

**return** UIManager.getSystemLookAndFeelClassName();

}

}

## Главная форма

/\*\*

\* Главная форма

\*/

public **class** MainForm **extends** JFrame {

/\*\* Корневая панель \*/

**private** JPanel rootPanel;

/\*\* Компонент для вывода списка \*/

**private** JList outList;

/\*\* Модель для вывода списка \*/

**private** DefaultListModel outListModel;

/\*\* Поле для ввода команды \*/

**private** JTextField cmdLine;

/\*\* Кнопка запуска команды \*/

**private** JButton runButton;

/\*\* Конструктор \*/

public MainForm() {

super("Program Runner");

**if** (false) System.out.println(">> constructor");

setContentPane(rootPanel);

setDefaultCloseOperation(WindowConstants.EXIT\_ON\_CLOSE);

setSize(600, 400);

// Добавление обработчиков событий

addActions();

setVisible(true);

}

/\*\* Добавление обработчиков событий \*/

void addActions() {

// Обработчик нажатия кнопки запуска команды

runButton.addActionListener(new ActionListener() {

**@**Override

public void actionPerformed(ActionEvent e) {

// Извлечение введенных команд

String cmdLineString **=** cmdLine.getText();

String[] strings **=** cmdLineString.split(" ");

List**<**String**>** commandList **=** **new** LinkedList**<**String**>**(Arrays.asList(strings));

// Запуск команды

String programName **=** commandList.remove(0);

ProgramRunner programRunner **=** **new** ProgramRunner(programName, commandList);

programRunner.run();

// Добавление разделителя

addSeparator();

// Получение вывода программы

List**<**String**>** outStringList **=** programRunner.getOutStringList();

// Отображение вывода программы

**for** (String out : outStringList) {

outListModel.addElement(out);

}

}

});

}

/\*\* Добавление разделителя в вывод \*/

**private** void addSeparator() {

DateFormat df **=** **new** SimpleDateFormat("dd.MM.yyyy a h:mm");

outListModel.addElement("-");

String separator **=** "-----------------------------------";

outListModel.addElement(separator **+** df.format(new Date()) **+** separator);

outListModel.addElement("-");

}

/\*\* Ручное создание компонентов \*/

**private** void createUIComponents() {

**if** (false) System.out.println(">> createUIComponents");

// Установка модели для компонента вывода списка

outListModel **=** **new** DefaultListModel();

outList **=** **new** JList(outListModel);

}

}

## Класс запускающий внешние программы

/\*\*

\* Запускатор внешних программ

\*/

public **class** ProgramRunner implements Runnable {

/\*\* Программа для запуска \*/

**private** **final** String programName;

/\*\* Аргументы для запуска \*/

**private** **final** List**<**String**>** args;

/\*\* Команда для запуска с аргументами \*/

**private** **final** List**<**String**>** command;

/\*\* Список строк вывода \*/

**private** **final** List**<**String**>** outStringList;

/\*\* Конструктор \*/

public ProgramRunner(String programName, List**<**String**>** args) {

this.programName **=** programName;

this.args **=** args;

command **=** **new** ArrayList**<**String**>**();

command.add(programName);

command.addAll(args);

outStringList **=** **new** ArrayList**<**String**>**();

}

/\*\* Запускатор программы \*/

**@**Override

public void run() {

**try** {

Process process **=** **new** ProcessBuilder(command).start();

InputStream is **=** process.getInputStream();

InputStreamReader isr **=** **new** InputStreamReader(is, "866");

BufferedReader br **=** **new** BufferedReader(isr);

System.out.printf("Command: %s\n", command);

String line;

**while** ((line **=** br.readLine()) **!=** null) {

outStringList.add(line);

}

} **catch** (IOException e) {

e.printStackTrace();

}

}

/\*\* Возвращает имя запускаемой программы \*/

public String getProgramName() {

**return** programName;

}

/\*\* Возвращает аргументы запускаемой программы \*/

public List**<**String**>** getArgs() {

**return** args;

}

/\*\* Возвращает вывод запускаемой программы \*/

public List**<**String**>** getOutStringList() {

**return** outStringList;

}

}

# Текст используемых источников

1. javatalks.ru [Электронный ресурс] // Форум Java программистов - <http://javatalks.ru/topics/15890> Режим доступа – свободный.
2. rsdn.ru [Электронный ресурс] // Русскоязычный сайт, посвящённый разработке программного обеспечения - <http://www.rsdn.ru/forum/java/2651137.all> Режим доступа – свободный.
3. sql.ru [Электронный ресурс] // все про SQL, базы данных, программирование и разработку информационных систем - <http://www.sql.ru/forum/328999> Режим доступа – свободный.