­­МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

­­­

Лабораторна робота №6

з курсу «Об’єктно-орієнтоване програмування»

для студентів базового напрямку 6.08.04 "Комп’ютерні науки"

(заочна форма навчання)

Варіант 14

Виконав студент гр. КНз-2

Чалий Михайло

­­

Львів 2014

# Потоки та робота із файлами у Java

## Мета роботи

Одержати навички роботи з потоками і файлами у Java

## Теоретичні відомості

У багатьох випадках потрібно виводити результати на принтер, у файл, базу даних або передавати по мережі. Вхідні дані теж часто приходиться завантажувати із файла, бази даних або із мережі. Для того щоб абстрагуватись від особливостей конкретних пристроїв введення/виведення, в Java використовується поняття потоку (stream). Вважається, що у програму іде вхідний поток (input stream) символів Unicode або просто байтів, що сприймається в програмі методами read(). Із програми методами write() або print (), println() виводиться вихідний потік (output stream) символів або байтів. При цьому не має значення куди направлений потік: на консоль, на принтер, у файл або в мережу, методи write() і print() нічого про це не знають.

Три потоки визначені у класі System статичними полями in, out і err. Їх можна використовувати без будь-яких додаткових визначень. Вони називаються відповідно стандартним введенням (stdin), стандартним виведенням (stdout) і стандартним виведенням повідомлень (stderr). Ці стандартні потоки можуть бути зєднані з різними конкретними присторями введення/виведення. Потоки out і err — це екземпляри класу Printstream, який організовує вихідний потік байтів. Ці екземпляри виводять інформацію на консоль методами print(), println() i write(), яких в класі Printstream є близько двадцати для різних типів аргументів.

Потік err призначений для виведення системних повідомлень програми: трасування, повідомлень про помилки або про виконання певних етапів програми. Такі дані звичайно заносяться в спеціальні журнали, log-файли, а не виводяться на консоль. В Java є засоби перепризначення потоку, наприклад, з консолі у файл.

Потік in — це екземпляр класу inputstream. Він призначений для клавіатурного введення з консолі методами read(). Клас inputstream є абстрактним, тому реально використовується хтось із його підкласів.

Методи організації потоків зібрані у класи пакета java.io. Крім класів, які організують потік, в пакет java.io входять класи з методами перетворення потоку, наприклад, можна перетворити потік байтів, які утворюють цілі числа, в потік цих чисел. Ще одна можливість, представлена класами пакета java.io, — злити декілька потоків в один потік.

Отже, в Java присутні чотири ієрархії класів для створення, перетворення і злиття потоків, які безпосередньо розширюють клас object:

* Reader — абстрактний клас, в якому зібрані найзагальніші методи символьного введення;
* Writer — абстрактний клас, в якому зібрані найзагальніші методи символьного виведення;
* Inputstream — абстрактний клас з загальними методами байтового введення;
* Outputstream — абстрактний клас з загальними методами байтового виведення.

Методи роботи з потоками видають IOException, якщо відбулася помилка введення/виведення. Четвертий метод skip (long n) "промотує" потік з поточної позиції на n символів або байтів вперед. Ці елементи потоку не вводяться методами read(). Метод повертає реальну кількість пропущених елементів, яка може відрізнятися від n, наприклад потік може закінчиться. Поточний елемент потоку можна помітити методом mark (int n), а потім повернутися до поміченого елементу методом reset(), але не більше ніж через n елементів. Не всі підкласи реалізують ці методи, тому перед розстановкою поміток треба звернутися до логічного методу marksupported(), який повертає true, якщо реалізовані методи розстановки і повернення до поміток.

Класи, що входять в ієрархії потоків введення/виведення, показані на рис. 1 та 2.

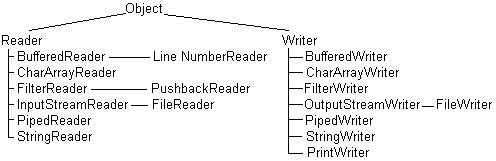


Рис. 1. Ієрархія символьних потоків

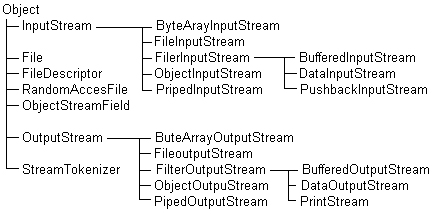


Рис. 2. Класи байтових потоків

## Завдання

Створити програму невеликого редактора (до 100 рядків) з використанням класу BufferedReader. Компоненти графічного вікна: напис "Введення тексту у файл" у області North, у області Center розміщена текстова область для введення тексту, у області South розміщені напис "Введіть ім'я файлу:", текстове поле для введення імені файлу і кнопка "Зберегти". Цей редактор вводить дані в масив рядків (розмір масиву - 100 елементів). Ознакою закінчення введення є символ "." в першій позиції рядка. При натисненні кнопки введені рядки запам'ятовуються в заданому файлі.

## Реалізація

Лістінг 1. Editor.java

**package** editor;

**import** java.awt.EventQueue;

**import** javax.swing.JFrame;

**import** javax.swing.JLabel;

**import** java.awt.BorderLayout;

**import** javax.swing.JOptionPane;

**import** javax.swing.JTextField;

**import** javax.swing.SwingConstants;

**import** javax.swing.JPanel;

**import** javax.swing.JButton;

**import** javax.swing.JTextArea;

**import** java.awt.event.ActionListener;

**import** java.awt.event.ActionEvent;

**import** java.awt.Insets;

**import** java.awt.Rectangle;

**import** javax.swing.border.LineBorder;

**import** java.awt.Color;

**import** java.io.FileWriter;

**import** java.io.IOException;

**public** **class** Editor {

**private** JFrame frame;

**private** JTextField fileNameField;

**private** JTextArea textArea;

/\*\*

\* Launch the application.

\*/

**public** **static** **void** main(String[] args) {

EventQueue.*invokeLater*(**new** Runnable() {

**public** **void** run() {

**try** {

Editor window = **new** Editor();

window.frame.setVisible(**true**);

} **catch** (Exception e) {

e.printStackTrace();

}

}

});

}

/\*\*

\* Create the application.

\*/

**public** Editor() {

initialize();

}

/\*\*

\* Initialize the contents of the frame.

\*/

**private** **void** initialize() {

frame = **new** JFrame();

frame.setBounds(100, 100, 574, 375);

frame.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);

JLabel captionLabel = **new** JLabel("Введення тексту у файл");

frame.getContentPane().add(captionLabel, BorderLayout.*NORTH*);

JPanel panel = **new** JPanel();

frame.getContentPane().add(panel, BorderLayout.*SOUTH*);

JLabel fileNameLabel = **new** JLabel("Введіть ім'я файлу:");

panel.add(fileNameLabel);

fileNameLabel.setHorizontalAlignment(SwingConstants.*LEFT*);

fileNameField = **new** JTextField();

fileNameField.setText("file.txt");

panel.add(fileNameField);

fileNameField.setColumns(20);

JButton saveButton = **new** JButton("Зберегти");

saveButton.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent arg0) {

saveFile();

}

});

panel.add(saveButton);

textArea = **new** JTextArea();

textArea.setBorder(**new** LineBorder(Color.*LIGHT\_GRAY*, 1, **true**));

textArea.setBounds(**new** Rectangle(2, 2, 2, 2));

textArea.setMargin(**new** Insets(10, 2, 2, 2));

frame.getContentPane().add(textArea, BorderLayout.*CENTER*);

}

**private** **void** saveFile(){

String fileName = **this**.fileNameField.getText();

**if** (fileName == **null** || fileName.length() == 0){

JOptionPane.*showMessageDialog*(frame,"Будьте ласкаві, введіть ім'я файлу");

**return**;

}

String text = **this**.textArea.getText();

**try** {

FileWriter fw = **new** FileWriter(fileName, **false**);

fw.write(text);

fw.close();

JOptionPane.*showMessageDialog*(frame,"Файл збережено");

} **catch** (IOException e) {

e.printStackTrace();

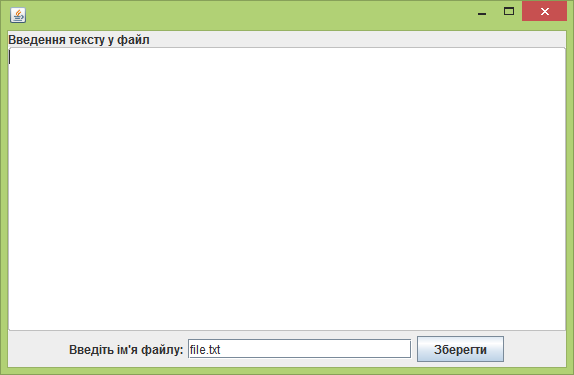
JOptionPane.*showMessageDialog*(frame, "Помилка запису файла\n" + e.getMessage());

}

}

}

## Результат



## Висновки

Одержав навички роботи з потоками і файлами у Java.