|  |
| --- |
| package sample; |
|  |  |
|  | public class Cash { |
|  | private int[][][] array; |
|  |  |
|  | public int[][][] getArray() { |
|  | return array; |
|  | } |
|  |  |
|  | public void setArray(int[][][] array) { |
|  | this.array = array; |
|  | } |
|  |  |
|  | public void setArrayValue(int value, int page, int n, int m) { |
|  | array[page][n][m] = value; |
|  | } |
|  |  |
|  | public int getArrayValue(int page, int n, int m) { |
|  | return array[page][n][m]; |
|  | } |
|  |  |
|  | public Cash(int page,int n, int m) { |
|  | array = new int[page][n][m]; |
|  | } |
|  |  |
|  | } |

|  |
| --- |
| package sample; |
|  |  |
|  | import java.io.FileNotFoundException; |
|  | import java.io.FileReader; |
|  | import java.io.FileWriter; |
|  | import java.io.IOException; |
|  | import java.util.Scanner; |
|  |  |
|  | public class Memory { |
|  |  |
|  | private String fileName; |
|  |  |
|  | public Memory(String fileName) { |
|  | this.fileName = fileName; |
|  | } |
|  |  |
|  | public void randomArray(int[][][] array) { |
|  | for (int i = 0; i < array.length ; i++) { |
|  | for (int j = 0; j < array[i].length; j++) { |
|  | for (int k = 0; k < array[i][j].length; k++) { |
|  | array[i][j][k]= (int) (Math.random()\*10000+1); |
|  | } |
|  | } |
|  | } |
|  | } |
|  |  |
|  | public void writeArray(int[][][] array) { |
|  | try { |
|  | FileWriter writer = new FileWriter(fileName); |
|  | for (int i = 0; i < array.length; i++) { |
|  | for (int j = 0; j < array[i].length; j++) { |
|  | for (int k = 0; k < array[i][j].length; k++) { |
|  | writer.write(array[i][j][k] + " "); |
|  | } |
|  | writer.write("\n"); |
|  | } |
|  | writer.write("\n"); |
|  | } |
|  | writer.flush(); |
|  | } catch (IOException e) { |
|  | e.printStackTrace(); |
|  | } |
|  | } |
|  |  |
|  | public String readArray() { |
|  | String[] temp = new String[1]; |
|  | String result = new String(); |
|  | FileReader fr= null; |
|  | try { |
|  | fr = new FileReader(fileName); |
|  | } catch (FileNotFoundException e) { |
|  | e.printStackTrace(); |
|  | } |
|  | Scanner scan = new Scanner(fr); |
|  | while (scan.hasNextLine()) { |
|  | temp[0] = scan.nextLine(); |
|  | result+=temp[0] +"\n"; |
|  | } |
|  | return result; |
|  | } |
|  |  |
|  | } |

|  |
| --- |
| package sample; |
|  |  |
|  | import javafx.fxml.FXML; |
|  | import javafx.scene.control.\*; |
|  |  |
|  | public class Controller { |
|  |  |
|  | Memory memory; |
|  | Cash cash; |
|  |  |
|  | @FXML |
|  | private TextArea areaMemory; |
|  |  |
|  | @FXML |
|  | private TextArea areaCash; |
|  |  |
|  | @FXML |
|  | private TextField strOne; |
|  |  |
|  | @FXML |
|  | private TextField strkOne; |
|  |  |
|  | @FXML |
|  | private TextField elemOne; |
|  |  |
|  | @FXML |
|  | private TextField value; |
|  |  |
|  | @FXML |
|  | private Button btnRead; |
|  |  |
|  | @FXML |
|  | private Button btnCreate; |
|  |  |
|  | @FXML |
|  | private TextField strTwo; |
|  |  |
|  | @FXML |
|  | private TextField strkTwo; |
|  |  |
|  | @FXML |
|  | private TextField elemTwo; |
|  |  |
|  | @FXML |
|  | private Button btnWrite; |
|  |  |
|  | @FXML |
|  | private Label labelTime; |
|  |  |
|  | @FXML |
|  | private Label labelElement; |
|  |  |
|  | @FXML |
|  | private Label labelStr; |
|  |  |
|  | public void warning() { |
|  | Alert alert = new Alert(Alert.AlertType.WARNING); |
|  | alert.setTitle("Ошибка"); |
|  | alert.setHeaderText("Введите корректные данные!"); |
|  | alert.showAndWait(); |
|  | } |
|  |  |
|  | @FXML |
|  | public void createMemory() { |
|  | if(strkOne.getText().equals("") || strOne.getText().equals("") || elemOne.getText().equals("")) { |
|  | warning(); |
|  | return; |
|  | } |
|  | cash = new Cash(Integer.parseInt(strOne.getText()),Integer.parseInt(strkOne.getText()),Integer.parseInt(elemOne.getText())); |
|  | memory.randomArray(cash.getArray()); |
|  | memory.writeArray(cash.getArray()); |
|  | areaMemory.setText(memory.readArray()); |
|  | } |
|  |  |
|  | @FXML |
|  | public void readMemory() { |
|  | if(strkTwo.getText().equals("") || strTwo.getText().equals("") || elemTwo.getText().equals("")) { |
|  | warning(); |
|  | return; |
|  | } |
|  | if(areaMemory.getText().equals("")) { |
|  | Alert alert = new Alert(Alert.AlertType.WARNING); |
|  | alert.setTitle("Ошибка"); |
|  | alert.setHeaderText("Создайте ОП"); |
|  | alert.showAndWait(); |
|  | return; |
|  | } |
|  | String text = ""; |
|  | String str = ""; |
|  | String element = ""; |
|  | int page = Integer.parseInt(strTwo.getText()); |
|  | int n = Integer.parseInt(strkTwo.getText()); |
|  | int m = Integer.parseInt(elemTwo.getText()); |
|  | for (int i = 0; i < cash.getArray()[page].length; i++) { |
|  | for (int j = 0; j < cash.getArray()[page][i].length; j++) { |
|  | text += String.valueOf(cash.getArrayValue(page,i,j)) + " "; |
|  | if(i==n) { |
|  | str+=String.valueOf(cash.getArrayValue(page,i,j)) + " "; |
|  | if(j==m) { |
|  | element+=String.valueOf(cash.getArrayValue(page,i,j)); |
|  | } |
|  | } |
|  | } |
|  | text+="\n"; |
|  | } |
|  | areaCash.setText(text); |
|  | labelStr.setText(str); |
|  | labelElement.setText(element); |
|  | } |
|  |  |
|  | @FXML |
|  | public void changeCashValue() { |
|  | if(strkTwo.getText().equals("") || strTwo.getText().equals("") || elemTwo.getText().equals("")) { |
|  | warning(); |
|  | return; |
|  | } |
|  | if(value.getText().equals("")) { |
|  | Alert alert = new Alert(Alert.AlertType.WARNING); |
|  | alert.setTitle("Ошибка"); |
|  | alert.setHeaderText("Введите новое значение!"); |
|  | alert.showAndWait(); |
|  | return; |
|  | } |
|  | int page = Integer.parseInt(strTwo.getText()); |
|  | int n = Integer.parseInt(strkTwo.getText()); |
|  | int m = Integer.parseInt(elemTwo.getText()); |
|  | long startTime = System.nanoTime(); |
|  | cash.setArrayValue(Integer.parseInt(value.getText()),page,n,m); |
|  | memory.writeArray(cash.getArray()); |
|  | memory.readArray(); |
|  | readMemory(); |
|  | long endTime = System.nanoTime(); |
|  | long timeSpent = endTime - startTime; |
|  | String temp = String.valueOf(timeSpent/1000); |
|  | labelTime.setText(temp + " мс"); |
|  | } |
|  |  |
|  |  |
|  |  |
|  | public void initialize() { |
|  | memory = new Memory("memory.txt"); |
|  | } |
|  |  |
|  |  |
|  |  |
|  | } |

|  |
| --- |
| package sample; |
|  |  |
|  | import javafx.application.Application; |
|  | import javafx.fxml.FXMLLoader; |
|  | import javafx.scene.Parent; |
|  | import javafx.scene.Scene; |
|  | import javafx.stage.Stage; |
|  |  |
|  | public class Main extends Application { |
|  |  |
|  | @Override |
|  | public void start(Stage primaryStage) throws Exception{ |
|  | Parent root = FXMLLoader.load(getClass().getResource("sample.fxml")); |
|  | primaryStage.setTitle("Моделирование работы КЭШ-Памяти"); |
|  | primaryStage.setScene(new Scene(root, 800, 600)); |
|  | primaryStage.show(); |
|  | primaryStage.setResizable(false); |
|  | } |
|  |  |
|  |  |
|  | public static void main(String[] args) { |
|  | launch(args); |
|  | } |
|  | } |

