EDITOR

**package** spellcheck;

**import** javax.swing.\*; // \* znaci ucitavaj sve klase

**import** javax.swing.text.DefaultHighlighter;

**import** javax.swing.text.Highlighter;

**import** javax.swing.text.Highlighter.HighlightPainter;

**import** java.io.\*;

**import** java.nio.charset.StandardCharsets;

**import** java.util.ArrayList;

**import** java.util.List;

**import** java.awt.Color;

**import** java.awt.GridLayout;

**import** java.awt.event.\*;

**import** java.awt.event.WindowAdapter; // ucitava WindowAdapter

**import** java.awt.event.WindowEvent;

**public** **class** Editor **extends** JFrame **implements** ActionListener {

**private** JFrame frejm;

**private** SCheck sc;

**private** JTextArea tekst;

**private** JTextArea tekst2;

**private** String otekst=""; //originalni tekst

**private** String poslednjidir="D:\\Java\\SpellChecker";

**private** String otvorenfajl="";

**private** String recnik="D:\\Java\\SpellChecker\\recnik.txt";

Editor(){

sc=**new** SCheck();

sc.citajR(recnik);

frejm = **new** JFrame("Mini-Editor");

JPanel tekstarea = **new** JPanel();

tekstarea.setLayout(**new** GridLayout(1,2));

tekst = **new** JTextArea(100,200);

tekst2 = **new** JTextArea(100,200);

JScrollPane scroll = **new** JScrollPane(tekst); //pravi scroll pane i dodaje na njega text

JScrollPane scroll2 = **new** JScrollPane(tekst2);

scroll.setVerticalScrollBarPolicy(JScrollPane.***VERTICAL\_SCROLLBAR\_ALWAYS***);

scroll2.setVerticalScrollBarPolicy(JScrollPane.***VERTICAL\_SCROLLBAR\_ALWAYS***);

tekst.setCaretPosition(0);

tekst2.setCaretPosition(0);

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

JMenuBar meni = **new** JMenuBar(); //meni

JMenu fajl= **new** JMenu("Fajl"); //prva kolona

JMenuItem novi= **new** JMenuItem("Novi"); //opcije prve kolone

JMenuItem otvori= **new** JMenuItem("Otvori");

JMenuItem snimi= **new** JMenuItem("Snimi");

JMenuItem sacuvajna=**new** JMenuItem("Sacuvaj na");

fajl.add(novi); //dodavanje opcija u prvu kolonu

fajl.add(otvori);

fajl.add(snimi);

fajl.add(sacuvajna);

meni.add(fajl); //dodavanje cele prve kolone na meni

novi.addActionListener(**this**); //osluskuje dogadjaje na opciji

otvori.addActionListener(**this**);

snimi.addActionListener(**this**);

sacuvajna.addActionListener(**this**);

JMenu obrada= **new** JMenu("Obrada"); //druga kolona

JMenuItem ispravi= **new** JMenuItem("Ispravi"); //opcija druge kolone

obrada.add(ispravi);

meni.add(obrada);

ispravi.addActionListener(**this**);

JMenu recnik= **new** JMenu("Recnik");

JMenuItem izaberi=**new** JMenuItem("Izaberi");

recnik.add(izaberi);

meni.add(recnik);

izaberi.addActionListener(**this**);

frejm.setJMenuBar(meni);

frejm.add(tekstarea);

tekstarea.add(scroll);

tekstarea.add(scroll2);

frejm.setSize(1366,780);

frejm.setVisible(**true**);

tekst2.setEnabled(**true**);

}

**public** **void** actionPerformed(ActionEvent e) { //hvata akcije u stipu stringa

String s =e.getActionCommand();

**if** (s.equals("Snimi")) snimi();

**else** **if** (s.equals("Otvori")) otvori();

**else** **if**(s.equals("Sacuvaj na")) sacuvajna();

**else** **if**(s.equals("Izaberi")) izaberi();

**else** **if** (s.equals("Novi")) {

tekst.setText("");

tekst2.setText("");

otvorenfajl="";

}

**else** **if** (s.equals("Ispravi")) {

**try** {

otekst=tekst.getText();

tekst2.setText(ispravi());

tekst.setCaretPosition(0);

tekst2.setCaretPosition(0);

}

**catch**(Exception a){

JOptionPane.*showMessageDialog*(frejm,"Desio se problem"+a);

}

}

**else**

JOptionPane.*showMessageDialog*(frejm, "Vec ste uradili ispravak, prvo povratite tekst");

}

**public** **void** otvori() {

JFileChooser jfc = **new** JFileChooser(poslednjidir);

**int** odgovor = jfc.showOpenDialog(**null**);

**if**(odgovor == JFileChooser.***APPROVE\_OPTION***) {

File fajl = **new** File(jfc.getSelectedFile().getAbsolutePath());

**try** {

String s="",t="";

FileReader fr = **new** FileReader(fajl, StandardCharsets.***ISO\_8859\_1***);

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

s=br.readLine();

**while**((t=br.readLine())!= **null**)

s=s+"\n"+t;

br.close();

tekst.setText(s);

}

**catch**(Exception e) {

JOptionPane.*showMessageDialog*(frejm, e.getMessage());

}

poslednjidir = fajl.getParent();

otvorenfajl = fajl.getAbsolutePath();

}

**else** JOptionPane.*showMessageDialog*(frejm, "Korisnik je odustao");

tekst.setCaretPosition(0);

otekst=tekst.getText();

tekst2.setText("");

}

**public** **void** snimi() {

**if**(!otvorenfajl.equals("")) {

File fajl = **new** File(otvorenfajl);

**try** {

FileWriter fw = **new** FileWriter(fajl,StandardCharsets.***ISO\_8859\_1***);

BufferedWriter bw = **new** BufferedWriter(fw);

bw.write(tekst.getText());

bw.flush();

bw.close();

}

**catch**(Exception e) {

JOptionPane.*showMessageDialog*(frejm, e.getMessage());

}

otekst=tekst.getText();

}

**else** sacuvajna();

}

**public** **void** sacuvajna() {

JFileChooser jfc = **new** JFileChooser(poslednjidir);

**int** odgovor = jfc.showSaveDialog(**null**);

**if** (odgovor == JFileChooser.***APPROVE\_OPTION***) {

File fajl = **new** File(jfc.getSelectedFile().getAbsolutePath());

**try** {

FileWriter fw = **new** FileWriter(fajl,StandardCharsets.***ISO\_8859\_1***);

BufferedWriter bw = **new** BufferedWriter(fw);

bw.write(tekst.getText());

bw.flush();

bw.close();

}

**catch**(Exception e) {

JOptionPane.*showMessageDialog*(frejm, e.getMessage());

}

poslednjidir = fajl.getParent();

otvorenfajl = fajl.getAbsolutePath();

otekst=tekst.getText();

}

}

**public** String ispravi() {

Highlighter highlight = tekst.getHighlighter();

highlight.removeAllHighlights();

snimi(); //ako je novi dokument onda mora prvo da se snimi

sc.citajT(otvorenfajl); //cita tekst

**if**(sc.prazan())

**return** "Niste uneli nista";

String[] reci=sc.reci(); //pokazivac na reci iz teksta

List<String> podvuceno = **new** ArrayList<>(); //lista podvucenih reci

String ceo="";

String zamena="";

String deo=otekst;

**for**(**int** i=0;i<reci.length;i++) {

**try** {

**int** p=0;

ceo+=deo.substring(p,deo.toUpperCase().indexOf(reci[i].toUpperCase(),0)); //svaki put uzima deo stringa sto se nalazi ispred reci

p=deo.toUpperCase().indexOf(reci[i].toUpperCase(),0)+reci[i].length(); //pocetna tacka se premesta na kraj reci kako bi se kasnije odsekao string

deo=deo.substring(p); //odseca ostatak stringa, kojim se dobija deo koji dalje treba proveriti

zamena=sc.proveri(reci[i]);

**if**(zamena.toUpperCase().equals(reci[i].toUpperCase())) { //proverava da li je zamena jednaka ukucanoj reci i da li je vec podvucena

**if**(!(podvuceno.contains(reci[i])))

{

podvuciTacan(reci[i]); //vrsi se podvlacenje

podvuceno.add(reci[i]);

}

}

**else**

{

**if**(!(podvuceno.contains(reci[i])))

{

podvuciGresku(reci[i]);

podvuceno.add(reci[i]);

}

}

ceo+=zamena; //na kraju se dodaje zamena izmedju odvojenih stringova

}

**catch**(Exception e) {

System.***out***.println("Desio se problem"+e);

}

}

podvuceno.clear();

**return** ceo; //na kraju se deo svede na prazan string i vraca se ceo tekst

}

**public** **void** izaberi() {

JFileChooser jfc = **new** JFileChooser(poslednjidir);

**int** odgovor = jfc.showOpenDialog(**null**);

**if**(odgovor == JFileChooser.***APPROVE\_OPTION***) {

recnik=jfc.getSelectedFile().getAbsolutePath();

sc.citajR(recnik);

Highlighter highlight = tekst.getHighlighter();

highlight.removeAllHighlights();

}

**else**

JOptionPane.*showMessageDialog*(frejm, "Korisnik je odustao");

}

**public** **void** podvuciGresku(String s) {

**try** {

Highlighter highlight = tekst.getHighlighter();

HighlightPainter paint = **new** DefaultHighlighter.DefaultHighlightPainter(Color.***yellow***);

**int** p=0;

**while**((p=tekst.getText().toUpperCase().indexOf(s.toUpperCase(),p))>=0){ //p dobija vrednost prvog stringa koji odgovara reci

highlight.addHighlight(p,p+s.length(),paint);

p+=s.length(); //p preskace tu rec posle bojenja, samim time je ne pronalazi na sledecoj iteraciji

}

}

**catch**(Exception e) {

System.***out***.println("Desio se problem"+e);

}

}

**public** **void** podvuciTacan(String s) {

**try** {

Highlighter highlight = tekst.getHighlighter();

HighlightPainter paint = **new** DefaultHighlighter.DefaultHighlightPainter(Color.***green***);

**int** p=0;

**while**((p=tekst.getText().toUpperCase().indexOf(s.toUpperCase(),p))>=0){

highlight.addHighlight(p,p+s.length(),paint);

p+=s.length();

}

}

**catch**(Exception e) {

System.***out***.println("Desio se problem"+e);

}

}

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

Editor editor= **new** Editor();

}

}

SCHECK

package spellcheck;

import java.nio.file.Files;

import java.nio.file.Paths;

import java.util.ArrayList;

import java.util.List;

public class SCheck {

private String[] recnik;

private String[] tekst;

public void citajT(String naziv) {

try {

tekst=Files.readAllLines(Paths.get(naziv)).toArray(new String[0]);

List<String> reci = new ArrayList<>(); //lista reci

for(int i=0;i<tekst.length;i++) {

String[] pom=tekst[i].split("");

if(tekst[i].equals(""))continue;

pom=tekst[i].split("[\t\n\"\',; .:<>@|#$!+=-]+"); //razdvajanje linije na reci

for(int j=0;j<pom.length;j++) {

reci.add(pom[j]); //ubacivanje svake reci u listu

}

}

tekst = reci.toArray(new String[reci.size()]); //konverzija liste reci u niz tekst

reci.clear();

}

catch(Exception e) {

System.out.print("Desio se problem "+ e);

}

}

public void citajR(String naziv) {

try {

recnik = Files.readAllLines(Paths.get(naziv)).toArray(new String[0]);

List<String> reci = new ArrayList<>(); //lista reci

for(int i=0;i<recnik.length;i++) {

String[] pom=recnik[i].split("[\t\n\"\',; .:<>@|#$!+=-]+"); //razdvajanje linije u reci

for(int j=0;j<pom.length;j++)

reci.add(pom[j]); //dodavanje reci u listu reci

}

recnik = reci.toArray(new String[reci.size()]); //konverzija liste reci u niz recnik

reci.clear();

}

catch(Exception e) {

System.out.print("Desio se problem "+ e);

}

}

public boolean prazan() {

if(tekst == null)

return true;

else

return false;

}

public double levenstein(String s, String t) {

s = s.toLowerCase();

t = t.toLowerCase();

int p=s.length(); //duzine reci

int q=t.length();

int[][] m = new int[p+1][q+1]; //matrica

for(int i=0;i<=p;i++)

m[i][0]=i;

for(int i=0;i<=q;i++)

m[0][i]=i;

for(int i=1;i<=p;i++)

for(int j=1;j<=q;j++)

{

int min=m[i-1][j-1];

if(s.charAt(i-1) != t.charAt(j-1))

min++;

if(min>m[i][j-1]+1)

min=m[i][j-1]+1;

if(min>m[i-1][j]+1)

min=m[i-1][j]+1;

m[i][j]=min;

}

return (double) m[p][q]/(p+q);

}

public String[] reci() {

return tekst;

}

public String proveri(String s) throws Exception{

if(recnik == null)

throw new NullPointerException();

for(int i=0;i<recnik.length;i++)

if (s.toUpperCase().equals(recnik[i].toUpperCase())) //proverava da li je rec u recniku

return s;

double min = levenstein(recnik[0],s);

List<String> reci = new ArrayList<>(); //lista reci koje su najblize trazenoj

for(int i=1;i<recnik.length;i++) { //prolazi se kroz recnik i porede se reci

double r = levenstein(recnik[i],s); //za svaku se racuna levenstajnovo rastojanje

if(r<min) {

min = r; //ukoliko je rastojanje trenutne reci manje od dosadasnjeg minimuma, min se menja

reci.clear(); //brisu se reci koje su bile na dosadasnjem minimumu

}

if(r == min) { //ukoliko je trenutna rec na jednakom rastojanju, a nije u listi reci, dodaje se

if(!reci.contains(recnik[i]))reci.add(recnik[i]);

}

}

String[] pom1=reci.toArray(new String[reci.size()]); //pomocna promenljiva za konverziju liste u niz, kako bi se lakse odradio ispis kasnije

String pom=s+"("+pom1[0]; //promenljiva koja se vraca

for(int i=1;i<pom1.length;i++)

pom+=", "+pom1[i]; //slaganje reci u promenljivu koja se vraca

return pom+")";

}

}