------------------------------ 1 A ZADATAK -----------------------------

----------main.c-----------

#include "myfactory.h"

#include <stdio.h>

#include <stdlib.h>

typedef char const\* (\*PTRFUN)();

struct Animal{

PTRFUN\* vtable;

// vtable entries:

// 0: char const\* name(void\* this);

// 1: char const\* greet();

// 2: char const\* menu();

};

// parrots and tigers defined in respective dynamic libraries

void animalPrintGreeting(struct Animal \*animal) {

printf("%s pozdravlja %s!\n", animal->vtable[0](animal), animal->vtable[1]());

}

void animalPrintMenu(struct Animal \*animal) {

printf("%s voli %s!\n", animal->vtable[0](animal), animal->vtable[2]());

}

int main(int argc, char \*argv[]){

for (int i=0; i<argc; ++i){

struct Animal\* p=(struct Animal\*)myfactory(argv[i], "Modrobradi");

if (!p){

printf("Creation of plug-in object %s failed.\n", argv[i]);

continue;

}

animalPrintGreeting(p);

animalPrintMenu(p);

free(p);

};

printf("---MyFactory2--\n");

for (int i=0; i<argc; ++i){

struct Animal\* p=(struct Animal\*)myfactory(argv[i], "Modrobradi");

if (!p){

printf("Creation of plug-in object %s failed.\n", argv[i]);

continue;

}

animalPrintGreeting(p);

animalPrintMenu(p);

free(p);

};

}

----------myfactory.c-----------

#include "myfactory.h"

#include <dlfcn.h>

#include <stddef.h>

#include <stdlib.h>

#include <stdio.h>

typedef void\* (\*CONSTRUCOTR)(char const \*);

typedef int (\*INTFUN)();

typedef void\* (\*INITIALIZATION)(void\*, char const \*);

void \*myfactory(char const \*libname, char const \*ctorarg) {

void\* handle = dlopen(libname, RTLD\_LAZY);

if(! handle) return NULL;

CONSTRUCOTR c = (CONSTRUCOTR) dlsym(handle, "create");

return c(ctorarg);

}

void \*myfactory2(char const \*libname, char const \*ctorarg) {

void\* handle = dlopen(libname, RTLD\_LAZY);

if(! handle) return NULL;

INTFUN sizeOf = (INTFUN) dlsym(handle, "sizeOf");

INITIALIZATION init = (INITIALIZATION) dlsym(handle, "construct");

//void\* animal = (void\*) malloc(sizeOf());

char animal1[sizeOf()];

void\* animal = (void\*) animal1;

init(animal,ctorarg);

return animal;

}

----------parrot.c-----------

#include <stdlib.h>

typedef char const\* (\*PTRFUN)();

struct Parrot

{

PTRFUN \*vtable;

const char\* name;

};

char const\* greet(void) {

return "pipi!";

}

char const\* menu(void) {

return "jabuku";

}

char const\* name(struct Parrot \*parrot){

return parrot->name;

}

PTRFUN table[3] = {

(PTRFUN) name,

(PTRFUN) greet,

(PTRFUN) menu

};

void construct(struct Parrot \*p, char \*name) {

p->name = name;

p->vtable = table;

}

void\* create(char \*name) {

//gomila

struct Parrot \*parrot = (struct Parrot \*)malloc(sizeof(struct Parrot));

construct(parrot, name);

return (void\*) parrot;

}

int sizeOf(){

sizeof(struct Parrot);

}

------------------------------ 1 b ZADTAK ------------------------------

----------Main.java----------

package hr.fer.zemris.ooup.lab3;

import java.io.File;

import java.net.MalformedURLException;

import java.net.URL;

import java.net.URLClassLoader;

import java.util.HashMap;

import java.util.Map;

import java.util.Map.Entry;

import hr.fer.zemris.ooup.lab3.model.Animal;

import hr.fer.zemris.ooup.lab3.model.AnimalFactory;

public class Main {

public static void main(String[] args) {

Map<String, String> životinje = new HashMap<String, String>();

životinje.put("Miško","Parrot");

for (Entry<String, String> entry : životinje.entrySet()) {

Animal a = AnimalFactory.newInstance(entry.getValue(), entry.getKey());

a.animalPrintGreeting();

a.animalPrintMenu();

}

}

}

----------Animal.java----------

package hr.fer.zemris.ooup.lab3.model;

public abstract class Animal {

public abstract String name();

public abstract String greet();

public abstract String menu();

public void animalPrintGreeting() {

System.out.println(name() + " pozdravlja sa " + greet());

}

public void animalPrintMenu() {

System.out.println(name() + " voli " + menu());

}

}

----------AnimalFactory.java----------

package hr.fer.zemris.ooup.lab3.model;

import java.io.File;

import java.lang.reflect.Constructor;

import java.lang.reflect.InvocationTargetException;

import java.net.MalformedURLException;

import java.net.URL;

import java.net.URLClassLoader;

import java.util.HashMap;

import java.util.Map;

public class AnimalFactory {

static Map<String,Class<Animal>> loaderi = new HashMap<>();

@SuppressWarnings("unchecked")

public static Animal newInstance(String animalKind, String name) {

Class<Animal> clazz = null;

Animal animal = null;

try {

ClassLoader parent = AnimalFactory.class.getClassLoader();

URLClassLoader newClassLoader = new URLClassLoader(

new URL[] {

// Dodaj jedan direktorij (završava s /)

new File("D:/java/plugins/").toURI().toURL(),

// Dodaj jedan konkretan JAR (ne završava s /)

new File("D:/java/plugins-jarovi/zivotinje.jar").toURI().toURL()

}, parent);

if(!loaderi.containsKey(animalKind))

loaderi.put(animalKind, (Class<Animal>) Class.forName("hr.fer.zemris.ooup.lab3.model.plugins."+animalKind, true, newClassLoader));

clazz = loaderi.get(animalKind);

Constructor<?> ctr = clazz.getConstructor(String.class);

animal = (Animal)ctr.newInstance(name);

} catch (ClassNotFoundException e) {

} catch (NoSuchMethodException e) {

} catch (SecurityException e) {

} catch (InstantiationException e) {

} catch (IllegalAccessException e) {

} catch (IllegalArgumentException e) {

} catch (InvocationTargetException e) {

} catch (MalformedURLException e) {

}

return animal;

}

}

----------Parrot.java----------

package hr.fer.zemris.ooup.lab3.model.plugins;

import hr.fer.zemris.ooup.lab3.model.Animal;

public class Parrot extends Animal {

String name;

public Parrot(String name) {

super();

this.name = name;

}

@Override

public String name() {

return name;

}

@Override

public String greet() {

return "pipi";

}

@Override

public String menu() {

// TODO Auto-generated method stub

return "jabuku";

}

}

------------------------------ 2 ZADATAK ------------------------------

------- TextEditorModel.java-------

package hr.fer.ooup.lab3.zad2;

import java.awt.Toolkit;

import java.awt.event.KeyEvent;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.Iterator;

import java.util.List;

import java.util.stream.Collectors;

public class TextEditorModel {

List<String> lines;

private Location cursorLocation;

LocationRange selectionRangle;

List<CursorObserver> cursorObservers = new ArrayList<>();

List<TextObserver> textObservers = new ArrayList<>();

boolean isSelected = false;

public TextEditorModel(String text) {

lines = new ArrayList<String>(Arrays.asList(text.split("\n")));

cursorLocation = new Location(0,0);

selectionRangle = new LocationRange(cursorLocation,cursorLocation);

}

public List<String> getLines() {

return lines;

}

public void setLines(List<String> lines) {

this.lines = lines;

}

public Location getCursorLocation() {

return cursorLocation;

}

public void setCursorLocation(Location cursorLocation) {

this.cursorLocation = cursorLocation;

this.notifyAllCursorObservers();

}

public Iterator<String> allLines() {

return lines.iterator();

}

public Iterator<String> linesRange(int index1,int index2) {

return lines.subList(index1, index2).iterator();

}

public void addTextObserver(TextObserver o) {

textObservers.add(o);

}

public void removeTextObserver(TextObserver o) {

textObservers.remove(o);

}

public void notifyAllTextObserver() {

for (TextObserver textObserver : textObservers) {

textObserver.text();

}

}

public void addCursorObserver(CursorObserver o) {

cursorObservers.add(o);

}

public void removeCursorObserver(CursorObserver o) {

cursorObservers.remove(o);

}

public void notifyAllCursorObservers() {

for (CursorObserver cursorObserver : cursorObservers) {

cursorObserver.updateCursorLocation(cursorLocation);

}

}

public void moveCursorLeft() {

if(cursorLocation.column == 0) {

if(cursorLocation.row == 0) return;

cursorLocation = new Location(cursorLocation.row - 1, lines.get(cursorLocation.row - 1).length());

notifyAllCursorObservers();

return;

}

cursorLocation = new Location(cursorLocation.row, cursorLocation.column - 1);

notifyAllCursorObservers();

}

public void moveCursorRight() {

if(cursorLocation.column == lines.get(cursorLocation.row).length()) {

if(cursorLocation.row + 1 == lines.size()) return;

cursorLocation = new Location(cursorLocation.row + 1, 0);

notifyAllCursorObservers();

return;

}

cursorLocation = new Location(cursorLocation.row, cursorLocation.column + 1);

notifyAllCursorObservers();

}

public void moveCursorUp() {

if(cursorLocation.row == 0) return;

if(cursorLocation.column > lines.get(cursorLocation.row - 1).length()) cursorLocation.column = lines.get(cursorLocation.row - 1).length();

cursorLocation = new Location(cursorLocation.row - 1, cursorLocation.column);

notifyAllCursorObservers();

}

public void moveCursorDown() {

if(cursorLocation.row + 1 == lines.size()) return;

if(cursorLocation.column > lines.get(cursorLocation.row + 1).length()) cursorLocation.column = lines.get(cursorLocation.row + 1).length();

cursorLocation = new Location(cursorLocation.row + 1, cursorLocation.column);

notifyAllCursorObservers();

}

public void deleteBefore() {

if(!selectionRangle.isStartEndSame()) {

deleteSelected();

return;

}

var prevLines = new ArrayList<String>(lines);

var prevCursorLocation = cursorLocation;

if(cursorLocation.column == 0) {

if(cursorLocation.row == 0) return;

int currentRow = cursorLocation.row;

moveCursorLeft();

lines.set(currentRow - 1, lines.get(currentRow - 1) + lines.get(currentRow));

lines.remove(lines.get(currentRow));

notifyAllTextObserver();

return;

}

String line = lines.get(cursorLocation.row);

lines.set(cursorLocation.row,line.substring(0,cursorLocation.column - 1) + line.substring(cursorLocation.column,line.length()));

moveCursorLeft();

notifyAllTextObserver();

var afterLines = new ArrayList<String>(lines);

var afterCursorLocation = cursorLocation;

UndoManager.getInstance().push(new EditAction() {

@Override

public void execute\_undo() {

lines = prevLines;

cursorLocation = prevCursorLocation;

notifyAllTextObserver();

}

@Override

public void execute\_do() {

lines = afterLines;

cursorLocation = afterCursorLocation;

notifyAllTextObserver();

}

});

}

public void deleteAfter() {

if(!selectionRangle.isStartEndSame()) {

deleteSelected();

return;

}

var prevLines = new ArrayList<String>(lines);

var prevCursorLocation = cursorLocation;

if(cursorLocation.column == lines.get(cursorLocation.row).length() ) {

if(cursorLocation.row == lines.size() -1 ) return;

int currentRow = cursorLocation.row;

lines.set(currentRow , lines.get(currentRow ) + lines.get(currentRow + 1));

lines.remove(lines.get(currentRow + 1));

notifyAllTextObserver();

var afterLines = new ArrayList<String>(lines);

var afterCursorLocation = cursorLocation;

UndoManager.getInstance().push(new EditAction() {

@Override

public void execute\_undo() {

lines = prevLines;

cursorLocation = prevCursorLocation;

notifyAllTextObserver();

}

@Override

public void execute\_do() {

lines = afterLines;

cursorLocation = afterCursorLocation;

notifyAllTextObserver();

}

});

return;

}

String line = lines.get(cursorLocation.row);

lines.set(cursorLocation.row,line.substring(0,cursorLocation.column) + line.substring(cursorLocation.column + 1,line.length()));

notifyAllTextObserver();

}

public void deleteSelected() {

var prevLines = new ArrayList<String>(lines);

var prevCursorLocation = cursorLocation;

var prevSelectionRange = selectionRangle;

LocationRange lr = selectionRangle.sorted();

if(lr.start.row == lr.end.row) {

String line = lines.get(lr.start.row);

lines.set(cursorLocation.row,line.substring(0,lr.start.column) + line.substring(lr.end.column,line.length()));

}

else {

Iterator<String> i = linesRange(lr.start.row, lr.end.row +1);

String s;

int removeEnterIndex = lr.end.row;

for(int j = lr.start.row ;i.hasNext(); s = i.next(),j++) {

String line = lines.get(j);

int a = j == lr.start.row ? lr.start.column : 0;

int b = j == lr.end.row ? lr.end.column : lines.get(j).length();

lines.set(j, line.substring(0,a) + line.substring(b,line.length()));

}

for(int u = lr.start.row +1 ;u < lr.end.row;u++) {

lines.remove(lr.start.row +1);

removeEnterIndex--;

}

lines.set(removeEnterIndex - 1, lines.get(removeEnterIndex - 1) + lines.get(removeEnterIndex));

lines.remove(lines.get(removeEnterIndex));

}

cursorLocation = lr.start;

notifyAllTextObserver();

notifyAllCursorObservers();

var afterLines = new ArrayList<String>(lines);

var afterCursorLocation = cursorLocation;

var afterSelectionRange = selectionRangle;

UndoManager.getInstance().push(new EditAction() {

@Override

public void execute\_undo() {

lines = prevLines;

cursorLocation = prevCursorLocation;

selectionRangle = prevSelectionRange;

notifyAllTextObserver();

}

@Override

public void execute\_do() {

lines = afterLines;

cursorLocation = afterCursorLocation;

selectionRangle = afterSelectionRange;

notifyAllTextObserver();

}

});

}

LocationRange getSelectionRange() {

return selectionRangle;

}

void setSelectionRange(LocationRange range) {

this.selectionRangle = range;

}

void changeSelectionRange() {

if(!isSelected) setSelectionRange(new LocationRange(cursorLocation,cursorLocation));

selectionRangle.end = cursorLocation;

}

void insert(char c,boolean addUndo){

var prevLines = new ArrayList<String>(lines);

var prevCursorLocation = cursorLocation;

var prevSelectionRange = selectionRangle;

if(!selectionRangle.isStartEndSame()) {

deleteSelected();

}

String line = lines.get(cursorLocation.row);

if(c == '\n') {

lines.set(cursorLocation.row,line.substring(0,cursorLocation.column));

lines.add(cursorLocation.row +1, line.substring(cursorLocation.column,line.length()));

cursorLocation.column = 0;

cursorLocation.row = cursorLocation.row +1;

notifyAllTextObserver();

notifyAllCursorObservers();

return;

}

lines.set(cursorLocation.row,line.substring(0,cursorLocation.column) + c +line.substring(cursorLocation.column,line.length()));

moveCursorRight();

notifyAllTextObserver();

notifyAllCursorObservers();

var afterLines = new ArrayList<String>(lines);

var afterCursorLocation = cursorLocation;

var afterSelectionRange = selectionRangle;

if(addUndo) UndoManager.getInstance().push(new EditAction() {

@Override

public void execute\_undo() {

lines = prevLines;

cursorLocation = prevCursorLocation;

selectionRangle = prevSelectionRange;

notifyAllTextObserver();

}

@Override

public void execute\_do() {

lines = afterLines;

cursorLocation = afterCursorLocation;

selectionRangle = afterSelectionRange;

notifyAllTextObserver();

}

});

}

void insert(String text) {

var prevLines = new ArrayList<String>(lines);

var prevCursorLocation = cursorLocation;

var prevSelectionRange = selectionRangle;

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

insert(text.charAt(i),false);

}

var afterLines = new ArrayList<String>(lines);

var afterCursorLocation = cursorLocation;

var afterSelectionRange = selectionRangle;

UndoManager.getInstance().push(new EditAction() {

@Override

public void execute\_undo() {

lines = prevLines;

cursorLocation = prevCursorLocation;

selectionRangle = prevSelectionRange;

notifyAllTextObserver();

}

@Override

public void execute\_do() {

lines = afterLines;

cursorLocation = afterCursorLocation;

selectionRangle = afterSelectionRange;

notifyAllTextObserver();

}

});

}

public String getSelectedText() {

LocationRange lr = selectionRangle.sorted();

if(lr.start.row == lr.end.row) {

String line = lines.get(lr.start.row);

return line.substring(lr.start.column,lr.end.column);

}

else {

List<String> tmp = new ArrayList<String>();

Iterator<String> i = linesRange(lr.start.row, lr.end.row +1);

String s;

int removeEnterIndex = lr.end.row;

for(int j = lr.start.row ;i.hasNext(); s = i.next(),j++) {

String line = lines.get(j);

int a = j == lr.start.row ? lr.start.column : 0;

int b = j == lr.end.row ? lr.end.column : lines.get(j).length();

tmp.add(line.substring(a,b));

}

return tmp.stream().collect(Collectors.joining("\n"));

}

}

}

--------TextEditor.java-----

package hr.fer.ooup.lab3.zad2;

import java.awt.Color;

import java.awt.Dimension;

import java.awt.Font;

import java.awt.Graphics;

import java.awt.Graphics2D;

import java.awt.event.ComponentListener;

import java.util.Iterator;

import java.util.Timer;

import java.util.TimerTask;

import java.util.concurrent.Executors;

import java.util.concurrent.ScheduledExecutorService;

import java.util.concurrent.ScheduledFuture;

import java.util.concurrent.TimeUnit;

import javax.sound.sampled.Line;

import javax.swing.JComponent;

public class TextEditor extends JComponent{

private static final Object LOCK = new Object();

boolean hasUndo;

boolean hasRedo;

TextEditorModel model;

JComponent component;

boolean tiktak = true;

ScheduledExecutorService s = Executors.newSingleThreadScheduledExecutor();

ScheduledFuture<?> future;

static Timer timer;

ClipboardStack clipboardStack;

public TextEditor(String text) {

component = this;

model = new TextEditorModel(text);

model.addCursorObserver((l) -> {

resetTimer(timer);

model.changeSelectionRange();

component.repaint();

});

model.addTextObserver(() -> repaint());

clipboardStack = new ClipboardStack();

clipboardStack.addClipboardObserver(()-> {

if(clipboardStack.textOut!= null) model.insert(clipboardStack.textOut);

});

UndoManager.getInstance().addListener(()-> {

hasUndo = !UndoManager.getInstance().undoStack.isEmpty();

hasRedo = !UndoManager.getInstance().redoStack.isEmpty();

});

s = Executors.newSingleThreadScheduledExecutor();

future = s.scheduleWithFixedDelay(() -> {

tiktak = !tiktak;

component.repaint();

}, 700, 700, TimeUnit.MILLISECONDS);

}

private void resetTimer(Timer timer) {

future.cancel(true);

tiktak = true;

future = s.scheduleWithFixedDelay(() -> {

tiktak = !tiktak;

component.repaint();

}, 700, 700, TimeUnit.MILLISECONDS);

}

@Override

public Dimension getPreferredSize() {

return new Dimension(700,700);

}

@Override

public void paintComponent(Graphics g) {

Graphics2D g2 = (Graphics2D) g;

g2.setFont(new Font("monospaced", Font.PLAIN, g2.getFont().getSize()));

g2.setColor(Color.white);

g2.fillRect(0, 0, getWidth(), getHeight());

g2.setColor(Color.black);

int y = 0;

Iterator<String> i = model.allLines();

while(i.hasNext()) {

String line = i.next();

int topPadding = g2.getFontMetrics().getHeight() - g2.getFontMetrics().getAscent();

int index = y / g2.getFontMetrics().getHeight();

if(index == model.getCursorLocation().row && tiktak) {

int axisX = g2.getFontMetrics().stringWidth(line.substring(0, model.getCursorLocation().column));

g2.drawLine(axisX, y + topPadding, axisX, y + g2.getFontMetrics().getHeight() + topPadding);

}

LocationRange lr = model.selectionRangle.sorted();

if(index >= lr.start.row && index <= lr.end.row) {

g2.setColor(new Color(51,204,255));

if(lr.start.row == lr.end.row) {

g2.fillRect(

(lr.start.column) \* g2.getFontMetrics().stringWidth("a"),

y+ topPadding,

(lr.end.column - lr.start.column) \* g2.getFontMetrics().stringWidth("a"),

g2.getFontMetrics().getHeight() +1

);

}

else if(index == lr.start.row) {

g2.fillRect(

(lr.start.column) \* g2.getFontMetrics().stringWidth("a"),

y+ topPadding,

(model.lines.get(index).length() - lr.start.column ) \* g2.getFontMetrics().stringWidth("a"),

g2.getFontMetrics().getHeight() +1

);

}

else if(index == lr.end.row) {

g2.fillRect(

0,

y+ topPadding,

(lr.end.column) \* g2.getFontMetrics().stringWidth("a"),

g2.getFontMetrics().getHeight() +1

);

}

else {

g2.fillRect(

0,

y+ topPadding,

g2.getFontMetrics().stringWidth(line),

g2.getFontMetrics().getHeight() +1

);

}

g2.setColor(Color.black);

}

g2.drawString(line, 0, y+= g2.getFontMetrics().getHeight());

}

}

void dispose(){

s.shutdownNow();

}

}

--------------Main.java--------------

package hr.fer.ooup.lab3.zad2;

import java.awt.BorderLayout;

import java.awt.Color;

import java.awt.Component;

import java.awt.Container;

import java.awt.event.ActionEvent;

import java.awt.event.ComponentEvent;

import java.awt.event.ComponentListener;

import java.awt.event.ContainerEvent;

import java.awt.event.ContainerListener;

import java.awt.event.KeyEvent;

import java.awt.event.KeyListener;

import java.awt.event.WindowAdapter;

import java.awt.event.WindowEvent;

import java.beans.PropertyChangeEvent;

import java.beans.PropertyChangeListener;

import java.io.File;

import java.io.IOException;

import java.lang.reflect.Constructor;

import java.lang.reflect.InvocationTargetException;

import java.net.MalformedURLException;

import java.net.URL;

import java.net.URLClassLoader;

import java.nio.charset.StandardCharsets;

import java.nio.file.Files;

import java.nio.file.Path;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.List;

import java.util.stream.Collectors;

import javax.swing.AbstractAction;

import javax.swing.Action;

import javax.swing.BorderFactory;

import javax.swing.Icon;

import javax.swing.JButton;

import javax.swing.JFileChooser;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JMenu;

import javax.swing.JMenuBar;

import javax.swing.JMenuItem;

import javax.swing.JOptionPane;

import javax.swing.JToolBar;

import javax.swing.SwingUtilities;

import javax.swing.WindowConstants;

import javax.swing.event.AncestorListener;

import hr.fer.ooup.lab3.zad2.plugins.Plugin;

import hr.fer.zemris.ooup.lab3.model.Animal;

public class Main extends JFrame {

String text = "Ovo je inicijalni test\nKoji se proteže u više redova\nOvo bi trebalo biti u 3 redu.\nOvo je jako dugačka rečenica te zbog toga se mora prelomiti automatski u dva retka nadam se da hoće.";

TextEditor textEditor;

Action openAction;

Action closeAction;

Action saveAction;

Action undoAction;

Action redoAction;

Action copyAction;

Action cutAction;

Action pasteAction;

Action pasteAndTakeAction;

Action deleteSelectionAction;

Action clearDocumentAction;

Action cursorStartAction;

Action cursorEndAction;

Path filePath;

List<Plugin> plugins;

private static final String PLUGINS\_PATH = "\\bin\\hr\\fer\\ooup\\lab3\\zad2\\plugins";

public Main() {

super();

setDefaultCloseOperation(WindowConstants.DISPOSE\_ON\_CLOSE);

setLocation(20, 20);

plugins = readPlugins();

initGUI();

pack();

this.setFocusable(true);

this.requestFocus();

addWindowListener(new WindowAdapter()

{

@Override

public void windowClosing(WindowEvent e)

{

textEditor.dispose();

e.getWindow().dispose();

}

});

}

private void initGUI() {

Container container = this.getContentPane();

setLayout(new BorderLayout());

MyComponent mc = new MyComponent();

add(mc, BorderLayout.WEST);

textEditor = new TextEditor(text);

createAction(this);

add(textEditor,BorderLayout.CENTER);

this.addKeyListener(new KeyListener() {

@Override

public void keyTyped(KeyEvent e) {

// TODO Auto-generated method stub

}

@Override

public void keyReleased(KeyEvent e) {

switch (e.getKeyCode()) {

case KeyEvent.VK\_SHIFT -> {

textEditor.model.isSelected = false;

}

}

}

@Override

public void keyPressed(KeyEvent e) {

char c = e.getKeyChar();

if(!Character.isIdentifierIgnorable(c) && (int)e.getKeyChar() != 65535 ) {

textEditor.model.isSelected = false;

textEditor.model.insert(c, true);

return;

}

System.out.println(e.getKeyCode());

switch (e.getKeyCode()) {

case KeyEvent.VK\_ENTER -> {

JFrame d = (JFrame)e.getSource();

d.dispose();

}

case KeyEvent.VK\_RIGHT -> textEditor.model.moveCursorRight();

case KeyEvent.VK\_LEFT -> textEditor.model.moveCursorLeft();

case KeyEvent.VK\_UP -> textEditor.model.moveCursorUp();

case KeyEvent.VK\_DOWN -> textEditor.model.moveCursorDown();

case KeyEvent.VK\_BACK\_SPACE -> textEditor.model.deleteBefore();

case KeyEvent.VK\_DELETE -> textEditor.model.deleteAfter();

case KeyEvent.VK\_SHIFT -> {

textEditor.model.isSelected = true;

}

case 67 -> textEditor.clipboardStack.push(textEditor.model.getSelectedText());

case 86 -> {

if(textEditor.model.isSelected == false) textEditor.clipboardStack.peek();

else {

textEditor.model.isSelected = false;

textEditor.clipboardStack.pop();

textEditor.model.isSelected = true;

}

}

case 88 -> {

textEditor.clipboardStack.push(textEditor.model.getSelectedText());

textEditor.model.deleteSelected();

}

case 90 -> {

if(textEditor.hasUndo) UndoManager.getInstance().undo();

}

case 89 -> {

if(textEditor.hasRedo)UndoManager.getInstance().redo();

}

}

}

});

createMenus();

createStatusBar();

}

private void createStatusBar() {

JLabel lengthLabel = new JLabel(textLabel(textEditor.model.lines.size(), 0, 0));

lengthLabel.setBorder(BorderFactory.createMatteBorder(

0, 1, 0, 0, Color.gray));

add(lengthLabel, BorderLayout.SOUTH);

textEditor.model.addCursorObserver((l)->

lengthLabel.setText(textLabel(

textEditor.model.lines.size(),

textEditor.model.getCursorLocation().row,

textEditor.model.getCursorLocation().column

))

);

}

String textLabel(int lines, int cursorRow, int cursorColumn) {

return " Broj linije: "+lines+", redak kursora: " + (cursorRow + 1) + ", stupac kursora:" + (cursorColumn +1);

}

private void createMenus() {

JMenuBar menuBar = new JMenuBar();

JMenu fileMenu = new JMenu("File");

menuBar.add(fileMenu);

fileMenu.add(new JMenuItem(openAction));

fileMenu.add(new JMenuItem(saveAction));

fileMenu.add(new JMenuItem(closeAction));

JMenu editMenu = new JMenu("Edit");

menuBar.add(editMenu);

editMenu.add(new JMenuItem(undoAction));

editMenu.add(new JMenuItem(redoAction));

editMenu.add(new JMenuItem(cutAction));

editMenu.add(new JMenuItem(copyAction));

editMenu.add(new JMenuItem(pasteAction));

editMenu.add(new JMenuItem(pasteAndTakeAction));

editMenu.add(new JMenuItem(deleteSelectionAction));

editMenu.add(new JMenuItem(clearDocumentAction));

JMenu moveMenu = new JMenu("Move");

menuBar.add(moveMenu);

moveMenu.add(new JMenuItem(cursorStartAction));

moveMenu.add(new JMenuItem(cursorEndAction));

JMenu pluginMenu = new JMenu("Plugins");

for (Plugin pl : plugins) {

pluginMenu.add(new JMenuItem(pluginAction(pl)));

}

menuBar.add(pluginMenu);

this.setJMenuBar(menuBar);

JToolBar t = new JToolBar();

t.add(new JButton(undoAction));

t.add(new JButton(redoAction));

t.add(new JButton(cutAction));

t.add(new JButton(copyAction));

t.add(new JButton(pasteAction));

this.add(t, BorderLayout.PAGE\_START);

}

private void createAction(Component parent) {

openAction = new AbstractAction() {

@Override

public void actionPerformed(ActionEvent e) {

Path path = fileChoice(parent);

if(path == null) return;

else {

if(!Files.isReadable(path)) throw new IllegalArgumentException("Datoteka: " + path.toAbsolutePath() + "ne postoji!");

filePath = path;

try {

byte[] bytes = Files.readAllBytes(path);

text = new String(bytes, StandardCharsets.UTF\_8);

textEditor.model.lines = new ArrayList<String>(Arrays.asList(text.split("\n")));

textEditor.model.setCursorLocation(new Location(0, 0));

textEditor.repaint();

} catch (IOException e1) {

throw new IllegalArgumentException("Pogreška pri učitavanju datoteke: " + path.toAbsolutePath());

}

}

}

};

openAction.putValue(Action.NAME,"Open");

saveAction = new AbstractAction() {

@Override

public void actionPerformed(ActionEvent e) {

if(filePath != null) {

byte[] podatci = textEditor.model.lines.stream().collect(Collectors.joining("\n")).getBytes(StandardCharsets.UTF\_8);

try {

Files.write(filePath, podatci);

} catch (IOException e1) {

throw new IllegalArgumentException("Nije moguće spremiti datoteku na putanju: " + filePath);

}

} else {

Path path = fileChoice(parent);

if(path == null) return;

try {

byte[] podatci = textEditor.model.lines.stream().collect(Collectors.joining("\n")).getBytes(StandardCharsets.UTF\_8);

try {

Files.write(path, podatci);

filePath = path;

} catch (IOException e1) {

throw new IllegalArgumentException("Nije moguće spremiti datoteku na putanju: " + path);

}

} catch (IllegalStateException e1) {

JOptionPane.showMessageDialog(parent,

"Datoteka na toj putanji je već otvorena",

"Warning",

JOptionPane.WARNING\_MESSAGE );

}

}

}

};

saveAction.putValue(Action.NAME,"Save");

closeAction = new AbstractAction() {

@Override

public void actionPerformed(ActionEvent e) {

JFrame d = (JFrame)parent;

d.dispose();

}

};

closeAction.putValue(Action.NAME,"Close");

undoAction = new AbstractAction() {

@Override

public void actionPerformed(ActionEvent e) {

if(textEditor.hasUndo) UndoManager.getInstance().undo();

}

};

undoAction.setEnabled(false);

UndoManager.getInstance().addListener(()-> {

undoAction.setEnabled(!UndoManager.getInstance().undoStack.isEmpty());

});

undoAction.putValue(Action.NAME,"Undo");

redoAction = new AbstractAction() {

@Override

public void actionPerformed(ActionEvent e) {

if(textEditor.hasRedo) UndoManager.getInstance().redo();

parent.requestFocus();

}

};

UndoManager.getInstance().addListener(()-> {

redoAction.setEnabled(!UndoManager.getInstance().redoStack.isEmpty());

});

redoAction.setEnabled(false);

redoAction.putValue(Action.NAME,"Redo");

cutAction = new AbstractAction() {

@Override

public void actionPerformed(ActionEvent e) {

textEditor.clipboardStack.push(textEditor.model.getSelectedText());

textEditor.model.deleteSelected();

parent.requestFocus();

}

};

cutAction.setEnabled(false);

textEditor.model.addCursorObserver((l) -> cutAction.setEnabled(!textEditor.model.selectionRangle.isStartEndSame()));

cutAction.putValue(Action.NAME,"Cut");

copyAction = new AbstractAction() {

@Override

public void actionPerformed(ActionEvent e) {

textEditor.clipboardStack.peek();

parent.requestFocus();

}

};

copyAction.setEnabled(false);

textEditor.model.addCursorObserver((l) -> copyAction.setEnabled(!textEditor.model.selectionRangle.isStartEndSame()));

copyAction.putValue(Action.NAME,"Copy");

pasteAction = new AbstractAction() {

@Override

public void actionPerformed(ActionEvent e) {

if(textEditor.model.isSelected == false) textEditor.clipboardStack.peek();

parent.requestFocus();

}

};

pasteAction.setEnabled(false);

textEditor.clipboardStack.addClipboardObserver(() -> pasteAction.setEnabled(!textEditor.clipboardStack.isEmpty()));

pasteAction.putValue(Action.NAME,"Paste");

pasteAndTakeAction = new AbstractAction() {

@Override

public void actionPerformed(ActionEvent e) {

if(textEditor.model.isSelected == false) textEditor.clipboardStack.pop();

parent.requestFocus();

}

};

pasteAndTakeAction.setEnabled(false);

textEditor.clipboardStack.addClipboardObserver(() -> pasteAndTakeAction.setEnabled(!textEditor.clipboardStack.isEmpty()));

pasteAndTakeAction.putValue(Action.NAME,"Past & Take");

deleteSelectionAction = new AbstractAction() {

@Override

public void actionPerformed(ActionEvent e) {

textEditor.model.deleteSelected();

parent.requestFocus();

}

};

deleteSelectionAction.setEnabled(false);

textEditor.model.addCursorObserver((l) -> deleteSelectionAction.setEnabled(!textEditor.model.selectionRangle.isStartEndSame()));

deleteSelectionAction.putValue(Action.NAME,"Delete Section");

clearDocumentAction = new AbstractAction() {

@Override

public void actionPerformed(ActionEvent e) {

textEditor.model.lines = new ArrayList<String>();

textEditor.model.lines.add("");

textEditor.model.setCursorLocation(new Location(0, 0));

textEditor.repaint();

parent.requestFocus();

}

};

clearDocumentAction.putValue(Action.NAME, "Clear document");

cursorStartAction = new AbstractAction("Cursor to document start") {

@Override

public void actionPerformed(ActionEvent e) {

textEditor.model.setCursorLocation(new Location(0, 0));

textEditor.repaint();

parent.requestFocus();

}

};

cursorEndAction = new AbstractAction("Cursor to document end") {

@Override

public void actionPerformed(ActionEvent e) {

textEditor.model.setCursorLocation(new Location(textEditor.model.lines.size() -1, textEditor.model.lines.get(textEditor.model.lines.size() -1).length()));

textEditor.repaint();

parent.requestFocus();

}

};

}

private Action pluginAction(Plugin pl) {

Action action = new AbstractAction(pl.getName()) {

@Override

public void actionPerformed(ActionEvent e) {

pl.execute(textEditor.model, UndoManager.getInstance(), textEditor.clipboardStack);

textEditor.repaint();

}

};

action.putValue(Action.SHORT\_DESCRIPTION, pl.getDescription());

return action;

}

public static void main(String[] args) {

SwingUtilities.invokeLater(()->{

new Main().setVisible(true);

});

}

private static Path fileChoice(Component parent) {

JFileChooser fc = new JFileChooser();

fc.setDialogTitle("Open file");

if(fc.showOpenDialog(parent)!=JFileChooser.APPROVE\_OPTION) {

return null;

}

File fileName = fc.getSelectedFile();

Path filePath = fileName.toPath();

return filePath;

}

private List<Plugin> readPlugins() {

List<Plugin> plugins = new ArrayList<Plugin>();

String fullPluginsPath = new File("").getAbsolutePath() + PLUGINS\_PATH;

File folder = new File(fullPluginsPath);

URLClassLoader newClassLoader;

try {

newClassLoader = new URLClassLoader(

new URL[] {

folder.toURI().toURL()

});

for(File f : folder.listFiles()) {

String name = f.getName();

if(name.endsWith(".class")) {

String className = "hr.fer.ooup.lab3.zad2.plugins." +name.substring(0, name.length() - 6);

Class<?> clasa = Class.forName(className ,true, newClassLoader);

if(clasa.isInterface()) continue;

try{

Class<Plugin> classPlugin = (Class<Plugin>) clasa;

Constructor<?> ctr = classPlugin.getConstructor();

plugins.add((Plugin)ctr.newInstance());

}catch (ClassCastException e) {

continue;

} catch (NoSuchMethodException e) {

} catch (SecurityException e) {

} catch (InstantiationException e) {

} catch (IllegalAccessException e) {

} catch (IllegalArgumentException e) {

} catch (InvocationTargetException e) {

}

}

}

} catch (MalformedURLException e) {;

} catch (ClassNotFoundException e) {

}

return plugins;

}

}

------------ ClipboardObserver------------

package hr.fer.ooup.lab3.zad2;

public interface ClipboardObserver {

void updateClipboard();

}

------------- ClipboardStack-------------

package hr.fer.ooup.lab3.zad2;

import java.util.ArrayList;

import java.util.List;

import java.util.Stack;

public class ClipboardStack extends Stack<String>{

String textOut;

List<ClipboardObserver > clipboardObservers = new ArrayList<>();

public void addClipboardObserver(ClipboardObserver o) {

clipboardObservers.add(o);

}

public void removeClipboardObserver(ClipboardObserver o) {

clipboardObservers.remove(o);

}

public void notifyAllClipboardObserver() {

for (ClipboardObserver clipboardObserver : clipboardObservers) {

clipboardObserver.updateClipboard();

}

}

public ClipboardStack() {

super();

}

@Override

public String push(String item) {

String pushIntem = super.push(item);

textOut = null;

notifyAllClipboardObserver();

return pushIntem;

}

@Override

public synchronized String pop() {

if(this.isEmpty()) return null;

String popItem = super.pop();

textOut = null;

notifyAllClipboardObserver();

return popItem;

}

@Override

public synchronized String peek() {

if(this.isEmpty()) return null;

textOut = super.peek();

notifyAllClipboardObserver();

return textOut;

}

}

---------EditAction-----------

package hr.fer.ooup.lab3.zad2;

public interface EditAction {

void execute\_do();

void execute\_undo();

}

----------Location------------

package hr.fer.ooup.lab3.zad2;

public class Location implements Comparable<Location>{

int row;

int column;

public Location(int row, int column){

super();

this.row = row;

this.column = column;

}

public int getRow() {

return row;

}

public void setRow(int row) {

this.row = row;

}

public int getColumn() {

return column;

}

public void setColumn(int column) {

this.column = column;

}

@Override

public int compareTo(Location o) {

int rowcmp = Integer.valueOf(this.row).compareTo(Integer.valueOf(o.row));

if(rowcmp != 0) return rowcmp;

return Integer.valueOf(this.column).compareTo(Integer.valueOf(o.column));

}

}

-----------LocationRange---------

package hr.fer.ooup.lab3.zad2;

public class LocationRange {

Location start;

Location end;

public LocationRange(Location start, Location end) {

super();

if(start.compareTo(end) > 0) {

Location t = start;

start = end;

end = t;

}

this.start = start;

this.end = end;

}

public Location getStart() {

return start;

}

public void reset(Location point) {

if(this.end.compareTo(point) >= 0) this.start = point;

else this.end = point;

if(start.compareTo(end) > 0) {

Location t = start;

start = end;

end = t;

}

}

public LocationRange sorted() {

LocationRange lr = new LocationRange(start, end);

if(lr.start.compareTo(lr.end) > 0) {

Location t = lr.start;

lr.start = lr.end;

lr.end = t;

}

return lr;

}

public boolean isStartEndSame() {

return start.compareTo(end) == 0;

}

}

----------MyCommponent-------------

package hr.fer.ooup.lab3.zad2;

import java.awt.Color;

import java.awt.Dimension;

import java.awt.Graphics;

import javax.swing.JComponent;

public class MyComponent extends JComponent {

@Override

public Dimension getPreferredSize() {

return new Dimension(200,100);

}

@Override

public void paint(Graphics g) {

g.setColor(Color.red);

g.drawLine(20, 20, 80, 20);

g.drawLine(20, 40, 20, 100);

g.setColor(Color.black);

g.drawString("Ovo je prvi redak teksta", 40, 40);

g.drawString("Ovo je drugi redak teksta", 40, 40 + g.getFontMetrics().getHeight());

}

}

------------TextObserver-----------

package hr.fer.ooup.lab3.zad2;

public interface TextObserver {

void text();

}

--------------UndoManager----------------

package hr.fer.ooup.lab3.zad2;

import java.util.ArrayList;

import java.util.List;

import java.util.Stack;

public class UndoManager {

Stack<EditAction> undoStack;

Stack<EditAction> redoStack;

List<Runnable> liseners = new ArrayList<>();

void addListener(Runnable l) {

liseners.add(l);

}

void notifyAllListners() {

for (Runnable l : liseners) {

l.run();

}

}

private static final UndoManager instance = new UndoManager();

public static UndoManager getInstance() {

return instance;

}

private UndoManager() {

undoStack = new Stack<EditAction>();

redoStack = new Stack<EditAction>();

}

void undo() {

EditAction action = undoStack.pop();

action.execute\_undo();

redoStack.push(action);

notifyAllListners();

}

void redo() {

EditAction action = redoStack.pop();

action.execute\_do();

undoStack.push(action);

notifyAllListners();

}

public void push(EditAction c) {

redoStack.clear();

undoStack.push(c);

notifyAllListners();

}

}

--------------Plugin-------------

package hr.fer.ooup.lab3.zad2.plugins;

import hr.fer.ooup.lab3.zad2.ClipboardStack;

import hr.fer.ooup.lab3.zad2.TextEditorModel;

import hr.fer.ooup.lab3.zad2.UndoManager;

public interface Plugin {

String getName(); // ime plugina (za izbornicku stavku)

String getDescription(); // kratki opis

void execute(TextEditorModel model, UndoManager undoManager, ClipboardStack clipboardStack);

}

-----------------Statistika--------------

package hr.fer.ooup.lab3.zad2.plugins;

import javax.swing.JOptionPane;

import hr.fer.ooup.lab3.zad2.ClipboardStack;

import hr.fer.ooup.lab3.zad2.TextEditorModel;

import hr.fer.ooup.lab3.zad2.UndoManager;

public class Statistika implements Plugin {

@Override

public String getName() {

return "Statistika";

}

@Override

public String getDescription() {

return "plugin koji broji koliko ima redaka, riječi i slova u dokumentu i to prikazuje korisniku u dijalogu.";

}

@Override

public void execute(TextEditorModel model, UndoManager undoManager, ClipboardStack clipboardStack) {

int line = model.getLines().size();

int words = model.getLines().stream().mapToInt((e) -> e.split(" ").length).sum();

int letters = model.getLines().stream().mapToInt((e) -> e.length()).sum();

JOptionPane.showMessageDialog(null, "Dokumnet sadrži "+line+" linija, "+words+" riječi i "+letters+" slova");

}

}

-----------------VelikoSlovo------------------

package hr.fer.ooup.lab3.zad2.plugins;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.stream.Collectors;

import hr.fer.ooup.lab3.zad2.ClipboardStack;

import hr.fer.ooup.lab3.zad2.EditAction;

import hr.fer.ooup.lab3.zad2.TextEditorModel;

import hr.fer.ooup.lab3.zad2.UndoManager;

public class VelikoSlovo implements Plugin {

public VelikoSlovo() {

super();

}

@Override

public String getName() {

return "Veliko slovo";

}

@Override

public String getDescription() {

return "prolazi kroz dokument i svako prvo slovo riječi mijenja u veliko";

}

@Override

public void execute(TextEditorModel model, UndoManager undoManager, ClipboardStack clipboardStack) {

var prevLines = new ArrayList(model.getLines());

model.setLines(

model.getLines().stream().map(

(e)-> Arrays.stream(e.split(" ")).map(

(e1) -> e1.substring(0, 1).toUpperCase() + e1.substring(1,e1.length())

).

collect(Collectors.joining(" "))

).toList()

);

var afterLines = new ArrayList<String>(model.getLines());

undoManager.getInstance().push(new EditAction() {

@Override

public void execute\_undo() {

model.setLines(prevLines);

model.notifyAllTextObserver();

}

@Override

public void execute\_do() {

model.setLines(afterLines);

model.notifyAllTextObserver();

}

});

}

}