АКАДЕМИЈА ТЕХНИЧКО-ВАСПИТАЧКИХ СТРУКОВНИХ СТУДИЈА

ОДСЕК НИШ

PROGRAMSKI ALATI ZA RAZVOJ SOFTVERA

Vežba 7

Kreiranje i testiranje jednostavnih GUI aplikacija

(radi se dve nedelje)

Razvoj softverskih aplikacija sa grafičkim korisničkim interfejsom (GUI) omogućuje korisnicima intuitivnu i vizuelno privlačnu interakciju sa programima. U ovoj vežbi, bavićemo se razvojem aplikacije za upravljanje kontaktima koja koristi GUI za dodavanje, izmenu, brisanje i prikaz kontakata. Ova vežba pruža osnove rada sa grafikom u programskim jezicima Java i Python, rad sa događajima i jednostavnu integraciju tekstualnih fajlova kao načinom skladištenja podataka.

Ciljevi vežbe su:

1. Razumevanje osnovnih pojmova grafičkog korisničkog interfejsa.
2. Razvoj praktičnih veština rada sa Java Swing i Python Tkinter bibliotekama.
3. Pravilna implementacija klasa i metoda koje će pokrivati neophodne operacije.
4. Korišćenje tekstualnih fajlova za čuvanje podataka.
5. Testiranje aplikacija pomoću JUnit-a i pytest-a.

Aplikacija za kontakte u Javi

Java Swing je jedna od osnovnih biblioteka za kreiranje grafičkih korisničkih interfejsa u Javi. Swing je deo Jave već dugo i omogućuje razvoj desktop aplikacija sa komponentama kao što su prozori, dugmeta, tekstualna polja, tabele i meniji.

Osnovne karakteristike Swing biblioteke:

- **Platformska nezavisnost**: Swing aplikacije se ponašaju isto na različitim operativnim sistemima, tako da nema brige da vaša aplikacija neće raditi ako je neko drugi pokrene.
- **Bogata kolekcija grafičkih komponenti**: Swing nudi različite komponente za formiranje korisničkog interfejsa, gde su sve komponente prilično intuitivne (prefiks J je dodatak).
- **Event-driven arhitektura**: Swing koristi događaje za interakciju korisnika sa aplikacijom.
- **Podrška za MVC**: Model-View-Controller arhitektura pruža razdvajanje podataka, prikaza i logike, što vam omogućuje da kasnije razvijete i kompleksnije komercijalne projekte.

Koraci za izradu:

1. Kreiranje glavnog prozora

- Napravite JFrame sa naslovom "Upravljanje kontaktima".
- Postavite dimenzije prozora i dodajte opcije za zatvaranje aplikacije.
- Dodajte meni bar (JMenuBar) sa opcijama:
    - Fajl > Učitaj kontakte za učitavanje podataka iz fajla.
    - Fajl > Sačuvaj kontakte za čuvanje trenutnih kontakata.
    - Fajl > Izlaz za zatvaranje aplikacije.

2. Tabela za prikaz kontakata

- Koristite JTable za prikaz kontakata u tabelarnom formatu.
- Tabela treba da ima kolone za ime, prezime, telefon i email.
- Podaci u tabeli su povezani sa dinamičkom listom koja se ažurira prilikom dodavanja, izmene ili brisanja kontakata.

3. Forma za unos/izmenu kontakata

- Napravite JDialog za unos kontakata sa sledećim poljima: ime, prezime, telefon i email.
- Dodajte dugme za potvrdu unosa koje ažurira podatke u tabeli i listi.
- Uvedite osnovnu validaciju podataka, npr. proveru da li su polja popunjena i validaciju formata emaila.

4. Čuvanje i učitavanje podataka

- **Čuvanje podataka:**
    - Koristite FileWriter za zapisivanje podataka u tekstualni fajl.
    - Svaki kontakt se zapisuje u jednom redu sa separatorom (zarez u ovom slučaju).
- **Učitavanje podataka:**
    - Koristite FileReader za čitanje fajla.
    - Parsirajte podatke i učitajte ih u listu i tabelu.

5. Akcije korisnika

- Dodavanje kontakta: Klikom na dugme otvara se forma za unos novog kontakta, koji se dodaje u tabelu i listu.
- Izmena kontakta: Izaberite kontakt iz tabele i otvorite formu sa već popunjenim podacima za izmenu. Nakon potvrde, ažurirajte listu i tabelu.
- Brisanje kontakta: Izaberite red iz tabele i obrišite kontakt iz liste i tabele.

Napomena:

- Validacija podataka je obavezna, da proverite da li su svi unosi ispravnog tipa i sadržaja. Korisničke poruke (pop-up prozori) dodatno pomažu u smanjenju grešaka.

Primer fajla:

```
Ana,Marković,0651234567,ana.markovic@gmail.com
Marko,Ilić,0629876543,marko.ilic@yahoo.com
Jovan,Petrović,0671249593,jovan.petrovic@icloud.com
```

Kao što možete da vidite, separator je zarez kod ovog tekst fajla. Možete koristiti i neki drugi simbol, ali vodite računa onda i kod da vam bude prilagođen tome.

Ideja za kod:

U nastavku je dat primer implementacije kako aplikacija za kontakte može izgledati. Vaš zadatak je da je implementirate sami i dodate bar još jednu funkcionalnost preko ove. Na primer, možete klikom na dugme sortirati kontakte po abecednom redu, u rastućem ili opadajućem poretku. Možete implementirati i funkcionalnost za pretragu kontakta po imenu, mejlu, ili broju telefona. Rezultat pretrage može izlaziti na ekranu kao pop-up prozor, ili da se ispiše u nekoj labeli.

```
import javax.swing.*;                          // Swing komponente
import javax.swing.table.DefaultTableModel;    // Model za tabelu
import java.awt.*;                             // Layout menadžeri
import java.awt.event.*;                       // Dogadjaji
import java.io.*;                              // Rad sa fajlovima
import java.util.ArrayList;                    // Lista za čuvanje podataka

// Glavna klasa aplikacije
public class KontaktMenadzer {

    // Glavne komponente aplikacije
    private JFrame frame;                      // Glavni prozor
    private JTable table;                      // Tabela za prikaz kontakata
    private DefaultTableModel tableModel;      // Model za tabelu
    private ArrayList<String[]> kontakti;      // Lista za čuvanje kontakata

    public KontaktMenadzer() {
        // Inicijalizacija liste kontakata
        kontakti = new ArrayList<>();

        // Kreiranje glavnog prozora
        frame = new JFrame("Upravljanje kontaktima");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  // Zatvaranje app
        frame.setSize(600, 400);                               // Dimenzije prozora
        frame.setLayout(new BorderLayout());                   // Stavljamo raspored
```

```
// Kreiranje menija
JMenuBar menuBar = new JMenuBar();           // Glavni meni
JMenu fileMenu = new JMenu("Fajl");          // "Fajl" opcija u meniju
JMenuItem loadMenuItem = new JMenuItem("Učitaj kontakte");
JMenuItem saveMenuItem = new JMenuItem("Sačuvaj kontakte");
JMenuItem exitMenuItem = new JMenuItem("Izlaz");

// Dodavanje funkcionalnosti meniju
loadMenuItem.addActionListener(e -> ucitajKontakte());
saveMenuItem.addActionListener(e -> sacuvajKontakte());
exitMenuItem.addActionListener(e -> System.exit(0));

// Dodavanje elemenata u meni
fileMenu.add(loadMenuItem);
fileMenu.add(saveMenuItem);
fileMenu.addSeparator();              // Separator
fileMenu.add(exitMenuItem);
menuBar.add(fileMenu);
frame.setJMenuBar(menuBar);           // Postavljanje menija u prozor

// Kreiranje tabele
String[] kolone = {"Ime", "Prezime", "Telefon", "Email"};
 // Model sa kolonama, bez redova
tableModel = new DefaultTableModel(kolone, 0);
 // Kreiramo tabelu sa modelom
table = new JTable(tableModel);
 // Dodajemo tabelu u centar prozora
frame.add(new JScrollPane(table), BorderLayout.CENTER);

// Kreiranje panela sa dugmetima
JPanel buttonPanel = new JPanel();
JButton addButton = new JButton("Dodaj kontakt");
JButton editButton = new JButton("Izmeni kontakt");
JButton deleteButton = new JButton("Obriši kontakt");

// Dodavanje funkcionalnosti dugmetima
addButton.addActionListener(e -> dodajKontakt());
```

```
    editButton.addActionListener(e -> izmeniKontakt());
    deleteButton.addActionListener(e -> obrisiKontakt());

    // Dodavanje dugmeta u panel
    buttonPanel.add(addButton);
    buttonPanel.add(editButton);
    buttonPanel.add(deleteButton);
    frame.add(buttonPanel, BorderLayout.SOUTH); // Postavljanje panela na dno

    // Prikaz prozora
    frame.setVisible(true);
}

private void dodajKontakt() {
    // Otvaranje dijaloga za unos kontakta
    KontaktForma forma = new KontaktForma(frame, null, kontakt -> {
        kontakti.add(kontakt);     // Dodajemo kontakt u listu
        tableModel.addRow(kontakt); // Dodajemo kontakt u tabelu
    });
    forma.setVisible(true);
}

private void izmeniKontakt() {
    // Hvatanje selektovanog reda
    int selektovaniRed = table.getSelectedRow();
    if (selektovaniRed >= 0) {
        String[] kontakt = kontakti.get(selektovaniRed);
        KontaktForma forma = new KontaktForma(frame, kontakt, izmenjenKontakt
         -> {
            kontakti.set(selektovaniRed, izmenjenKontakt);
            // Izmena u listi
            for (int i = 0; i < izmenjenKontakt.length; i++) {
                tableModel.setValueAt(izmenjenKontakt[i], selektovaniRed, i);
                // Izmena u tabeli
            }
        });
        forma.setVisible(true);
```

```
    } else {
        JOptionPane.showMessageDialog(frame, "Izaberite kontakt za izmenu.",
                                      "Upozorenje", JOptionPane.WARNING_MESSAGE);
    }
}

private void obrisiKontakt() {
    // Brisanje selektovanog reda
    int selektovaniRed = table.getSelectedRow();
    if (selektovaniRed >= 0) {
        kontakti.remove(selektovaniRed);         // Uklanjamo iz liste
        tableModel.removeRow(selektovaniRed);    // Uklanjamo iz tabele
    } else {
        JOptionPane.showMessageDialog(frame, "Izaberite kontakt za brisanje.",
                                      "Upozorenje", JOptionPane.WARNING_MESSAGE);
    }
}

private void ucitajKontakte() {
    // Učitavanje kontakata iz fajla
    try (BufferedReader reader =
                          new BufferedReader(new FileReader("kontakti.txt"))) {
        kontakti.clear();
        tableModel.setRowCount(0); // Brišemo sve redove iz tabele
        String linija;
        while ((linija = reader.readLine()) != null) {
            String[] kontakt = linija.split(",");
            kontakti.add(kontakt);
            tableModel.addRow(kontakt);
        }
    } catch (IOException e) {
        JOptionPane.showMessageDialog(frame, "Greška pri čitanju fajla.",
                                      "Greška", JOptionPane.ERROR_MESSAGE);
    }
}
```

```
    private void sacuvajKontakte() {
        // Čuvanje kontakata u fajl
        try (BufferedWriter writer =
                            new BufferedWriter(new FileWriter("kontakti.txt"))) {
            for (String[] kontakt : kontakti) {
                writer.write(String.join(",", kontakt));
                writer.newLine();
            }
        } catch (IOException e) {
            JOptionPane.showMessageDialog(frame, "Greška pri pisanju fajla.",
                                        "Greška", JOptionPane.ERROR_MESSAGE);
        }
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(KontaktMenadzer::new);
                    // Pokrećemo aplikaciju na Swing niti (threads)
    }
}

// Klasa za unos/izmenu kontakta
class KontaktForma extends JDialog {
    public KontaktForma(JFrame parent, String[] kontakt, KontaktCallback callback)
    {
        super(parent, true);
        setTitle(kontakt == null ? "Dodaj kontakt" : "Izmeni kontakt");
        setSize(300, 200);
        setLayout(new GridLayout(5, 2));

        JTextField imeField = new JTextField(kontakt != null ? kontakt[0] :"");
        JTextField prezimeField = new JTextField(kontakt != null ? kontakt[1] :"");
        JTextField telefonField = new JTextField(kontakt != null ? kontakt[2] :"");
        JTextField emailField = new JTextField(kontakt != null ? kontakt[3] :"");

        add(new JLabel("Ime:"));
        add(imeField);
        add(new JLabel("Prezime:"));
```

```
        add(prezimeField);
        add(new JLabel("Telefon:"));
        add(telefonField);
        add(new JLabel("Email:"));
        add(emailField);

        JButton sacuvajButton = new JButton("Sačuvaj");
        sacuvajButton.addActionListener(e -> {
            String[] noviKontakt = {imeField.getText(), prezimeField.getText(),
                                    telefonField.getText(), emailField.getText()};
            callback.onKontaktSačuvan(noviKontakt);
            dispose();
        });

        add(sacuvajButton);
    }

    // Funkcionalni interfejs za prosleđivanje kontakta
    interface KontaktCallback {
        void onKontaktSačuvan(String[] kontakt);
    }
}
```

Grafičke komponente korišćene u aplikaciji:

- **JFrame**: Glavni prozor aplikacije, predstavlja osnovni kontejner.
- **JTable**: Komponenta za prikaz tabele sa kontaktima, koristi model DefaultTableModel za rad sa podacima.
- **JMenuBar i JMenu**: Koristi se za navigaciju kroz aplikaciju, pružajući opcije kao što su učitavanje i čuvanje kontakata.
- **JButton**: Dugmeta za dodavanje, izmenu i brisanje kontakta.
- **JPanel**: Panel za organizaciju dugmadi na dnu prozora.
- **JDialog**: Koristi se za unos ili izmenu podataka o kontaktima (klasa KontaktForma).

Raspored komponenti:

- **BorderLayout**: Glavni raspored prozora, gde su meni i tabela u centralnom delu, dok su dugmeta smeštena na dnu.
- **GridLayout**: Koristi se unutar dijaloga za unos i izmenu kontakta, raspoređujući labele i tekstualna polja u mrežu.

Rešavanje klik događaja:

- Koriste se ActionListener objekti za osluškivanje događaja na dugmetima i stavkama menija.
- Primer: Klik na dugme "Dodaj kontakt" otvara dijalog pomoću metode dodajKontakt(), gde korisnik unosi podatke, a zatim se novi kontakt dodaje u tabelu i listu.

Da bismo bolje iskoristili **OOP** principe, možemo kreirati i klasu **Kontakt** koja čuva podatke o jednom kontaktu. Na taj način, umesto korišćenja niza stringova, koristili bismo listu objekata tipa Kontakt.

```
public class Kontakt {
    private String ime;
    private String prezime;
    private String telefon;
    private String email;

    public Kontakt(String ime, String prezime, String telefon, String email) {
        this.ime = ime;
        this.prezime = prezime;
        this.telefon = telefon;
        this.email = email;
    }

    // Getteri i setteri
    public String getIme() { return ime; }
    public void setIme(String ime) { this.ime = ime; }
    public String getPrezime() { return prezime; }
    public void setPrezime(String prezime) { this.prezime = prezime; }
    public String getTelefon() { return telefon; }
    public void setTelefon(String telefon) { this.telefon = telefon; }
    public String getEmail() { return email; }
    public void setEmail(String email) { this.email = email; }

    @Override
    public String toString() {
        return ime + "," + prezime + "," + telefon + "," + email;
    }
}
```

Potrebne izmene u kodu za korišćenje klase Kontakt:

1. Umesto ArrayList<String[]>, koristi se ArrayList<Kontakt>.
2. Metode za dodavanje, izmenu i brisanje treba da budu prilagođene za rad sa objektima tipa Kontakt. Primer za dodavanje novog kontakta je dat u nastavku:

```
private void dodajKontakt(String ime, String prezi, String tel, String mail)
{
    KontaktForma forma = new KontaktForma(frame, null, noviKontakt -> {
        kontakti.add(noviKontakt);  // Dodavanje objekta tipa Kontakt u listu
        tableModel.addRow(new String[] {
            noviKontakt.getIme(),
            noviKontakt.getPrezime(),
            noviKontakt.getTelefon(),
            noviKontakt.getEmail()
        });  // Dodavanje u tabelu, tu i dalje imamo stringove po kolonama
    });
    forma.setVisible(true);
}
```

Prilikom čuvanja, koristi se toString() metoda klase Kontakt za formatiranje podataka. Učitavanje zahteva kreiranje objekata Kontakt na osnovu pročitanih podataka.

Ovaj pristup poboljšava čitljivost i održivost koda, jer se koristi jasna struktura podataka i objekata umesto nizova stringova. Naravno, za svoja rešenja možete koristiti oba prisutpa, ili pak neki treći ako imate ideju, bićete ocenjeni na osnovu tačnosti koda i kvaliteta celokupne implementacije.

Primer JUnit testova:

```
import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;
import java.util.List;

public class KontaktMenadzerTest {

    @Test
    void testDodajKontakt() {
        // Kreiramo menadžer i kontakt za testiranje
        KontaktMenadzer menadzer = new KontaktMenadzer();
        String[] kontakt = {"Marko", "Marković", "0651234567",
                            "marko@example.com"};
```

```
    // Dodajemo kontakt
    menadzer.dodajKontakt(kontakt);

    // Proveravamo da li je kontakt uspešno dodat
    assertEquals(1, menadzer.getKontakti().size(),
                "Broj kontakata treba biti 1 nakon dodavanja.");
    assertArrayEquals(kontakt, menadzer.getKontakti().get(0),
                      "Kontakt nije pravilno dodat.");
}

@Test
void testUcitajKontakte() {
    // Kreiramo menadžer i učitavamo kontakte iz test fajla
    KontaktMenadzer menadzer = new KontaktMenadzer();
    menadzer.ucitajKontakte("test_kontakti.txt");

    // Proveravamo da li su svi kontakti učitani
    List<String[]> kontakti = menadzer.getKontakti();
    assertEquals(2, kontakti.size(), "Broj učitanih kontakata treba biti 2.");

    // Proveravamo tačnost podataka za prvi kontakt
    String[] prvi = {"Marko", "Marković", "0651234567", "marko@example.com"
    assertArrayEquals(prviKontakt, kontakti.get(0), "Prvi kontakt nije pravilno
                                                              učitan.");}
@Test
void testIzmeniKontakt() {
    // Kreiramo menadžer i dodajemo kontakt
    KontaktMenadzer menadzer = new KontaktMenadzer();
    String[] kontakt = {"Marko", "Marić", "0651234567", "marko@example.com"};
    menadzer.dodajKontakt(kontakt);

    // Izmenimo kontakt
    String[] izmKont = {"Marko", "Marić", "0657654321", "marko@izmenjen.com"};
    menadzer.izmeniKontakt(0, izmenjeniKontakt);

    // Proveravamo da li je kontakt pravilno izmenjen
    assertEquals(1, menadzer.getKontakti().size(), "Broj kontakata treba biti
                                                                    1");
```

```
        assertArrayEquals(izmenjeniKontakt, menadzer.getKontakti().get(0),
                                            "Kontakt nije pravilno izmenjen.");
    }

    @Test
    void testObrisiKontakt() {
        // Kreiramo menadžer i dodajemo kontakte
        KontaktMenadzer menadzer = new KontaktMenadzer();
        String[] kontakt1 = {"Marko", "Marić", "0651234567", "marko@example.com"};
        String[] kontakt2 = {"Ana", "Antić", "0619876543", "ana@example.com"};
        menadzer.dodajKontakt(kontakt1);
        menadzer.dodajKontakt(kontakt2);

        // Brišemo prvi kontakt
        menadzer.obrisiKontakt(0);

        // Proveravamo da li je kontakt obrisan
        assertEquals(1, menadzer.getKontakti().size(), "Broj kontakata treba biti
                                                        1 nakon brisanja.");
        assertArrayEquals(kontakt2, menadzer.getKontakti().get(0),
                                              "Preostali kontakt nije ispravan.");
    }
}
```

Aplikacija za kontakte u Pythonu

Python Tkinter je ugrađena biblioteka za razvoj GUI aplikacija u Pythonu. Tkinter je jednostavna, ali moćna biblioteka koja omogućava brzo kreiranje grafičkog interfejsa koristeći Python sintaksu.

Osnovne karakteristike Tkinter-a:

- **Jednostavan za učenje i upotrebu**: Tkinter omogućuje početnicima da brzo započnu rad sa grafikom, prvenstveno jer su sve komande intuitivne.
- **Standardna Python biblioteka**: Nema komplikovanih instalacija, jer Tkinter je dostupan u svim standardnim Python distribucijama.
- **Podrška za različite UI elemente**: Tkinter nudi komponente poput okvira, tekstualnih polja, dugmeta, check boxova, padajućih menija i tabela sa podacima.
- **Podrška za događaje**: Tkinter koristi event-driven model za reagovanje na korisničke akcije, tako da se sve aktivnosti koje se dešavaju klikom na komponentu, ili recimo promenom indeksa u padajućoj listi, realizuju preko događaja i funkcija koje ih prate.

Sledi ideja za kod, koristićemo istu strukturu

tekstualnog fajla kao kod Java implementacije, samo što će sve sada da bude prilagođeno Python TKinter-u. Vaš zadatak je i ovde da odradite svoju verziju kontakt aplikacije, i dodate još jednu funkcionalnost po želji.

```
import tkinter as tk                    # Modul za kreiranje GUI aplikacija
from tkinter import ttk, messagebox     # ttk za napredne GUI komponente
import os                               # Modul za rad sa fajlovima i folderima

# Klasa koja implementira glavnu logiku aplikacije
class KontaktManager:
    def __init__(self, root):
        """
        Konstruktor klase za inicijalizaciju glavnih GUI elemenata i logike.
        """
        self.root = root  # Glavni prozor aplikacije
        self.root.title("Upravljanje kontaktima")  # Naslov prozora
        self.kontakti = []  # Lista za čuvanje kontakata u memoriji

        # Glavni meni aplikacije
        menu = tk.Menu(self.root)                       # Kreiramo meni
        self.root.config(menu=menu)                     # Dodajemo meni u prozor
        file_menu = tk.Menu(menu, tearoff=0)            # Podmeni za opcije
        menu.add_cascade(label="Fajl", menu=file_menu)  # Dodajemo "Fajl" opciju
        file_menu.add_command(label="Učitaj kontakte",
                              command=self.ucitaj_kontakte)  # Opcija za učitavanje
        file_menu.add_command(label="Sačuvaj kontakte",
                              command=self.sacuvaj_kontakte) # Opcija za čuvanje
        file_menu.add_separator()                           # Separator u meniju
        file_menu.add_command(label="Izlaz", command=self.root.quit) # Izlazak

        # Tabela za prikaz kontakata
        self.tree = ttk.Treeview(
            self.root,  # Glavni prozor kao roditelj
            columns=("Ime", "Prezime", "Telefon", "Email"),  # Kolone u tabeli
            show="headings"  # Prikaz bez dodatnog praznog prvog stupca
        )
```

```
    # Naslovi kolona
    self.tree.heading("Ime", text="Ime")
    self.tree.heading("Prezime", text="Prezime")
    self.tree.heading("Telefon", text="Telefon")
    self.tree.heading("Email", text="Email")
    self.tree.pack(fill=tk.BOTH, expand=True)   # Popunjavanje prostora

    # Dugmeta za manipulaciju podacima
    button_frame = tk.Frame(self.root)          # Okvir za dugmeta
    button_frame.pack(fill=tk.X)  # Raspored na horizontalnoj osi
    tk.Button(button_frame, text="Dodaj kontakt",
              command=self.dodaj_kontakt).pack(side=tk.LEFT, padx=5, pady=5)
    tk.Button(button_frame, text="Izmeni kontakt",
               command=self.izmeni_kontakt).pack(side=tk.LEFT, padx=5, pady=5)
    tk.Button(button_frame, text="Obriši kontakt",
               command=self.obrisi_kontakt).pack(side=tk.LEFT, padx=5, pady=5)

def dodaj_kontakt(self):
    """Otvara formu za dodavanje novog kontakta."""
    self.otvori_formu(izmena=False)  # Izmena = False jer dodajemo novi kontakt

def izmeni_kontakt(self):
    """
    Otvara formu za izmenu postojećeg kontakta.
    Prvo proverava da li je selektovan kontakt.
    """
    selektovan = self.tree.focus()  # Uzima ID trenutno selektovanog reda
    if selektovan:
        indeks = self.tree.index(selektovan)  # Pozicija reda u listi
       # Otvaramo formu za izmenu
        self.otvori_formu(izmena=True, indeks=indeks)
    else:
         # Poruka ako nema odabira
        messagebox.showwarning("Upozorenje", "Izaberite kontakt za izmenu.")
def obrisi_kontakt(self):
```

```
    """
    Briše selektovani kontakt iz tabele i liste.
    """
    selektovan = self.tree.focus()              # Hvata ID selektovanog reda
    if selektovan:
        indeks = self.tree.index(selektovan)    # Pozicija u listi
        self.kontakti.pop(indeks)               # Uklanjamo iz liste
        self.tree.delete(selektovan)            # Uklanjamo iz tabele
    else:
       # Poruka ako nema selekcije
        messagebox.showwarning("Upozorenje", "Izaberite kontakt za brisanje.")

def otvori_formu(self, izmena, indeks=None):
    """
    Otvara novu formu za dodavanje ili izmenu kontakta.
    """
    form = tk.Toplevel(self.root)  # Kreiramo novi prozor
    form.title("Izmena kontakta" if izmena else "Dodavanje kontakta")  # Naslov
    form.geometry("300x200")  # Dimenzije prozora

    # Polja za unos podataka
    tk.Label(form, text="Ime").grid(row=0, column=0, pady=5, padx=5)
    ime_entry = tk.Entry(form)  # Polje za ime
    ime_entry.grid(row=0, column=1, pady=5, padx=5)

    tk.Label(form, text="Prezime").grid(row=1, column=0, pady=5, padx=5)
    prezime_entry = tk.Entry(form)  # Polje za prezime
    prezime_entry.grid(row=1, column=1, pady=5, padx=5)

    tk.Label(form, text="Telefon").grid(row=2, column=0, pady=5, padx=5)
    telefon_entry = tk.Entry(form)  # Polje za telefon
    telefon_entry.grid(row=2, column=1, pady=5, padx=5)

    tk.Label(form, text="Email").grid(row=3, column=0, pady=5, padx=5)
```

```
    email_entry = tk.Entry(form)  # Polje za email
    email_entry.grid(row=3, column=1, pady=5, padx=5)

    if izmena:  # Ako radimo izmenu, popunimo polja postojećim podacima
        kontakt = self.kontakti[indeks]
        ime_entry.insert(0, kontakt[0])         # Unos imena
        prezime_entry.insert(0, kontakt[1])     # Unos prezimena
        telefon_entry.insert(0, kontakt[2])     # Unos telefona
        email_entry.insert(0, kontakt[3])       # Unos emaila

    def sacuvaj():
        """
        Validacija unosa i čuvanje podataka u listi i tabeli.
        """
        ime = ime_entry.get().strip()
        prezime = prezime_entry.get().strip()
        telefon = telefon_entry.get().strip()
        email = email_entry.get().strip()

        if not ime or not prezime or not telefon or not email:
            # Provera praznih polja
            messagebox.showwarning("Upozorenje", "Sva polja su obavezna.")
            return

        if izmena:    # Ažuriranje postojećeg kontakta
            self.kontakti[indeks] = (ime, prezime, telefon, email)
            self.tree.item(self.tree.get_children()[indeks],
                           values=(ime, prezime, telefon, email))
        else:         # Dodavanje novog kontakta
            self.kontakti.append((ime, prezime, telefon, email))
            self.tree.insert("", tk.END, values=(ime, prezime, telefon, email))
        form.destroy()  # Zatvaramo formu

    tk.Button(form, text="Sačuvaj", command=sacuvaj).grid(row=4, column=0,
                                                          columnspan=2, pady=10)
def ucitaj_kontakte(self):
```

```
        """
        Učitavanje kontakata iz tekstualnog fajla.
        """
        # Proveravamo da li fajl postoji
        if os.path.exists("kontakti.txt"):
            with open("kontakti.txt", "r") as f:
                # Čitamo podatke iz fajla
                self.kontakti = [line.strip().split(",") for line in f]
                # Dodajemo u tabelu
                for kontakt in self.kontakti:
                    self.tree.insert("", tk.END, values=kontakt)
        else:
            # Poruka ako fajl ne postoji
            messagebox.showinfo("Informacija", "Fajl ne postoji.")

    def sacuvaj_kontakte(self):
        """
        Čuvanje kontakata u tekstualni fajl.
        """
        with open("kontakti.txt", "w") as f:  # Otvaramo fajl za pisanje
            for kontakt in self.kontakti:     # Upisujemo svaki kontakt
                f.write(",".join(kontakt) + "\n")

# Glavna funkcija za pokretanje aplikacije
if __name__ == "__main__":
    root = tk.Tk()  # Kreiramo glavni prozor
    app = KontaktManager(root)  # Inicijalizujemo aplikaciju
    root.mainloop()  # Pokrećemo glavni događajni petlju
```

Osnovni elementi aplikacije:

- **Glavni prozor**: Prikazuje meni, tabelu za kontakte i dugmeta za manipulaciju podacima.
- **Meni**: Opcije za učitavanje, čuvanje kontakata i izlazak iz aplikacije.
- **Tabela**: Prikazuje podatke o kontaktima sa kolonama za ime, prezime, telefon i email.
- **Dugmeta**: Omogućuju dodavanje, izmenu i brisanje kontakata.

Funkcionalnosti:

1. **Dodavanje kontakta (dodaj_kontakt)**:
    - Otvara formu za unos novih podataka.
    - Poziva metodu otvori_formu sa parametrom izmena=False.
2. **Izmena kontakta (izmeni_kontakt)**:
    - Proverava da li je selektovan red u tabeli.
    - Ako jeste, otvara formu sa postojećim podacima za izmenu.
3. **Brisanje kontakta (obrisi_kontakt)**:
    - Selektuje red u tabeli i uklanja ga iz liste i tabele.
    - Prikazuje poruku upozorenja ako nije selektovan kontakt.
4. **Otvaranje forme (otvori_formu)**:
    - Kreira novi prozor sa poljima za unos podataka.
    - Ako je izmena, popunjava polja postojećim podacima.
    - Metoda sacuvaj: Validira unos i dodaje ili ažurira kontakte.
5. **Učitavanje kontakata (ucitaj_kontakte)**:
    - Proverava postojanje fajla kontakti.txt.
    - Učitava podatke iz fajla i prikazuje ih u tabeli.
    - Ako fajl ne postoji, prikazuje informativnu poruku.
6. **Čuvanje kontakata (sacuvaj_kontakte)**:
    - Upisuje podatke iz liste self.kontakti u fajl kontakti.txt.

Pomoćne funkcionalnosti

- **Validacija unosa**:
    - Sva polja (ime, prezime, telefon, email) su obavezna.
    - Koristi messagebox za prikaz grešaka.
- **Dijaloge sa porukama**:
    - show warning: Upozorenje o praznim poljima ili nedostatku selekcije.
    - show info: Informacija o neuspelim operacijama (npr. nedostatak fajla).

Glavna funkcija (main)

- Kreira glavni prozor aplikacije (Tk).
- Inicijalizuje klasu KontaktManager.
- Pokreće glavnu petlju aplikacije (mainloop)

Primer Pytestova:

```
import os
import pytest
from kontakt_menadzer import KontaktMenadzer

# Fixture za kreiranje instance menadžera
@pytest.fixture
def menadzer():
    return KontaktMenadzer()

def test_dodaj_kontakt(menadzer):
    # Dodajemo kontakt u menadžer
    kontakt = ("Marko", "Marković", "0651234567", "marko@example.com")
    menadzer.dodaj_kontakt(kontakt)

    # Provera da li je kontakt dodat
    assert len(menadzer.kontakti) == 1, "Broj kontakata treba biti 1 nakon
                                                                dodavanja."
    assert menadzer.kontakti[0] == kontakt, "Kontakt nije pravilno dodat."

def test_ucitaj_kontakte(menadzer):
    # Kreiramo test fajl sa kontaktima
    test_fajl = "test_kontakti.txt"
    with open(test_fajl, "w") as f:
        f.write("Marko,Marković,0651234567,marko@example.com\n")
        f.write("Ana,Antić,0619876543,ana@example.com\n")

    # Učitavamo kontakte
    menadzer.ucitaj_kontakte(test_fajl)

    # Provere učitanih kontakata
    assert len(menadzer.kontakti) == 2, "Broj učitanih kontakata treba biti 2."
    assert menadzer.kontakti[0] == ("Marko", "Marković", "0651234567",
                                "marko@example.com"), "Prvi kontakt nije ispravan."
```

```
    assert menadzer.kontakti[1] == ("Ana", "Antić", "0619876543",
                                    "ana@example.com"), "Drugi kontakt nije ispravan."

    # Brisanje test fajla
    os.remove(test_fajl)

def test_izmeni_kontakt(menadzer):
    # Dodajemo kontakt
    kontakt = ("Marko", "Marković", "0651234567", "marko@example.com")
    menadzer.dodaj_kontakt(kontakt)

    # Menjamo kontakt
    izmenjeni_kontakt = ("Marko", "Marković", "0657654321", "marko@izmenjen.com")
    menadzer.izmeni_kontakt(0, izmenjeni_kontakt)

    # Provera izmena
    assert menadzer.kontakti[0] == izmenjeni_kontakt, "Kontakt nije pravilno
                                                                    izmenjen."

def test_obrisi_kontakt(menadzer):
    # Dodajemo kontakte
    kontakt1 = ("Marko", "Marković", "0651234567", "marko@example.com")
    kontakt2 = ("Ana", "Antić", "0619876543", "ana@example.com")
    menadzer.dodaj_kontakt(kontakt1)
    menadzer.dodaj_kontakt(kontakt2)

    # Brišemo prvi kontakt
    menadzer.obrisi_kontakt(0)

    # Provera brisanja
    assert len(menadzer.kontakti) == 1, "Broj kontakata treba biti 1 nakon
                                                                brisanja."
    assert menadzer.kontakti[0] == kontakt2, "Preostali kontakt nije ispravan."
```

Korisni resursi

https://www.geeksforgeeks.org/introduction-to-java-swing/

https://www.javatpoint.com/java-swing

https://www.guru99.com/java-swing-gui.html

https://www.youtube.com/watch?app=desktop&v=Kmgo00avvEw&t=0s&ab_channel=BroCode

https://www.youtube.com/playlist?list=PLlGZc17KPrVCGRKtgbdvnGshN8AePlqpd

https://www.geeksforgeeks.org/python-gui-tkinter/

https://www.tutorialspoint.com/python/tk_canvas.htm

https://tkinterpython.top/drawing/

https://www.youtube.com/watch?v=mop6g-c5HEY&ab_channel=ClearCode

https://www.youtube.com/@TkinterPython