UNIVERZITET U NOVOM SADU
PRIRODNO-MATEMATIČKI
FAKULTET
DEPARTMAN ZA MATEMATIKU
I INFORMATIKU

Jovan Trkulja

Aplikacija za rezervisanje avionskih karata

- seminarski rad iz predmeta Skript jezici-

Novi Sad, 2019.

Sadržaj

1. Uvod

Phyton je programski jezik koji je platformski nezavisan, objektno-orijentisan, interpreterski i interaktivan. Spada u grupu softvera otvorenog koda sa dobrom podrškom, dokumentacijom i pratećim bibliotekama. Nastao je u akademskim uslovima, na univerzitetu Stičing u Holandiji 1989. godine. Mnoštvo projekata je zasnovano na Phytonu i ima primenu u organizacijama i firmama širom sveta: Google, Dropbox, Disney, NASA, Youtube... Pored programiranja konzolnih aplikacija, Python podržava I GUI programiranje.

Moj program – rezervacije avionskih karata biće podeljen na tri modula. Prvi modul će se zvati "sluzbenici.py" koji pokriva sve neophodne metode za logovanje i dodavanje novih službenika, kao i metode koje su neophodne ako neki službenik zaboravi korisničko ime i lozinku, da može bez problema i vratiti iste. Drugi modul će se zvati "avioKarte.py" koji pokriva sve neophodne metode za rezervisanje avio karti i štampanju reda letenja. U poslednjem modulu "Main.py" objedniće se navedena dva modula i testirati rad programa.

2. Opis programa

Na početku programa kreiran je projekat pod nazivom „Avionske karte" i unutar njega paket pod nazivom „Seminarski" u kojem su smešteni moduli i fajlovi neophodni za rad programa.

2.1 Tekstualni fajl "sluzbenici.txt"

Pomenuti tekstualni fajl sadrži podatke o službenicima koji rade za agenciju koja vrši uslugu rezervacije avio karata. Fajl sadrži podatke redom: (ID službenika, ime, prezime, korisničko ime, lozinka)

```
4722,Andreja,Lovric,aki1989,akiaki555
101,Ana,Savic,ana123,Beograd244
4034,Milan,Mazinjanin,mmiki225,MilanSanja883
510,Vladimir,Stevic,vlada123,dovlaeStevke785
671,Aleksa,Novkovic,aleksa987,Leksaee225
298,Tamara,Agovic,tamarce555,TamaraLuka123
200,Vladimir,Novakovic,dovla,dovla
9822,Dimitrije,Djukic,bule,bule
252,Andreja,Simic,aSimic,amdrija
2019,Jovan,Trkulja,JTrkulja,trki
```

2.2 Službenici (modul)

Pre bilo kakvog rada, moramo uvesti određene metode iz modula avioKarte.py naredbom:

```
From Seminarski.avioKarte import printFlights, loadFlights, reserveFlight, sortFlights
```

Modul „sluzbenici.py“ koristi gore opisan tekstualni fajl za realizaciju. Službenik je klasnog tipa i svaki ima autribute ID, ime, prezime, user, sifra. Unutar klase postoje dva metoda. Prvi metod toString() vraća opis službenika sa njegovim podacima, dok drugi metod getID() vraća ID službenika koji koristimo kasnije prilikom sortiranja službenika. ID svakog službenika je jedinstven pa s’tim ne mogu postojati 2 službenika sa istim ID-om u fajlu.

```
class Sluzbenik:

    def __init__(self, ide, ime, prez,user,sifra):
        self.id = ide
        self.ime = ime
        self.prez = prez
        self.user = user
        self.sifra = sifra

    def toString(self):
        return "Sluzbenik: ID -> "+str(self.id)+", "+self.ime+" "+self.prez

    def getId(self):
        return int(self.id)
```

Funkcija loadEmployee()

Službenici u programu su predstavljeni listom službenika pod nazivom “employee“čiji elementi su upravo objekti klase Sluzbenik. Na početku lista je prazna. Funkcija loadEmployee() učitava iz fajla “sluzbenici.txt“ (koji je otvoren u režimu za čitanje) službenike redom, od svakog službenika pravi objekat klase i nakon toga ga dodaje u listu. Nakon učitavanja fajl se zatvara. Pošto su podaci o službeniku razdvojeni zarezom neophodno je prvo pročitati celu liniju fajla, podeliti u pojedinačne podatke delimiterom zarez i tek nakon toga kreirati objekat pod nazivom “sluzbenik“. Funkcija strip() je ugrađena i ona se koristi da odseče bele znakove sa početka i kraja stringa (kao funkcija trim() u Java programskom jeziku).

```
employee = []

def loadEmployee():

    fajl = open("sluzbenici.txt", "r")
    for line in fajl:
        ide, ime, prez, user, sifra = line.split(",")
        sluzbenik=
Sluzbenik(ide.strip(),ime.strip(),prez.strip(),user.strip(),sifra.strip())
        employee.append(sluzbenik)
        fajl.close()
```

Funkcije printEmployee() i sortEmployee()

Funkcija printEmployee štampa zaposlene koji se nalaze u listi employee pozivajući funkciju toString() iz klase Sluzbenik.

Funkcija sortEmployee sortira listu zaposlenih. Budući da elementi liste nisu primitivni tipovi već objekti neke klase, za ovu potrebu morali smo uvesti lambda funkciju k koja poziva metod getID() iz klase Sluzbenik i vrši sortiranje. Sortiramo isključivo rastućim redosledom.

```
def printEmployee():

    for line in employee:
        print(line.toString())

def sortEmployee():

     employee.sort(key = lambda k: k.getId(reverse = False)
```

Funkcija preglednik()

Ova funkcija ne radi nikakvu obradu, već samo na ekran ispisuje ponuđene opcije sa kojima službenik raspolaže prilikom rada.

```
def preglednik():
    print("=====================================\n")
    print("Izaberite opciju:")
    print("1. Uloguj se")
    print("2. Dodaj sluzbenika")
    print("3. Pregled (sortiranje) zaposlenih")
    print("4. Zaboravljeno korisnicko ime i lozinka")
    print("5. Kraj \n")
    print("=====================================\n")
```

Funkcija choice(n)

Funkcija choice(n) simulira “switch – case “ naredbu u Javi i zapravo obrađuje aktivnost koju je službenik odredio izborom jedne od ponuđenih 5 opcija u pregledniku. Parametar n predstavlja opciju koju korisnik unosi sa tastature. Izgled funkcije prikazan je na sledećoj stranici.

```
def choice(n):
    if(n == 1):
        user = input("Unesite korisnicko ime -> ")
        sifra = input("Unesite lozinku -> ")
        if login(user,sifra):
            print("Uspesno ulogovan sluzbenik: ")
            printWorker(user,sifra)
            nextJob()
        else:
            print("Pogresno korisnicko ime i lozinka. Da li ste zaboravili?")
            x = input("Unesite da ili ne -> ")
            if x == 'da':
                choice(4)
            else:
                choice(1)
    elif n == 2:
        addEmployee()
    elif n == 3:
        sortEmployee()
        printEmployee()
    elif n == 4:
        forgotUserOrPassword()
    else:
        print("Program je zavrsio sa radom")
        return
```

Ukoliko je korisnik uneo kao opciju 1, od njega će biti zahtevano da unese svoje korisničko ime i lozinku. Da li su podaci ispravno unešeni proverava funkcija login(user, sifra). Ukoliko funkcija vrati vrednost True, zaposleni postoji u listi i može pristupiti daljem radu.

Funkcija login(user,sifra)

Vrši se pretraživanje da li se uneti podaci slažu sa podacima u listi. Ukoliko se pronađu isti parametre vraća se vrednost True odnosno da takav radnik postoji i da može da se uloguje. Ukoliko vrati vrednost False onda takav radnik ne postoji i vrše se dalje opcije.

```
def login(user, sifra):

    for x in employee:
        if x.user == user and x.sifra == sifra:
            return True
    return False
```

Ukoliko funkcija vrati True ispisaće se podaci o radniku pozivom toString() metoda iz klase Sluzbenik. Ovu operaciju vrši funkcija printWorker koja kao parametre prima user i sifru ulogovanog radnika.

```
def printWorker(user, sifra):

    for x in employee:
        if x.user == user and x.sifra == sifra:
            print("**********************")
            print(x.toString())
            print("**********************")
```

Funkcije nextJob() i tempMeni()

Nakon odštampanog radnika poziva se metod nextJob() koji službeniku sada nudi konkretne opcije vezane za rezervaciju avionskih karata. Funkcija nextJob prvo poziva funkciju tempMeni koja na ekran ispisuje ponuđene opcije za rad. Nakon toga službenik unosi redni broj operacije. Vrši se kontrola unosa pa tako neće biti omogućen unos broja koji je van opsega operacija. Kada korisnik unese validan broj poziva se metod choiceFlights(x) i nastavlja se dalji rad.

```
def tempMeni():
    print("1. Stampaj red letenja (red letenja je sortiran) \n")
    print("2. Rezervisi let \n")
    print("3. Promeni sluzbenika \n")

def nextJob():
    print("Dobrodosli u meni. Izaberite opciju i nastavite dalji
rad. \n")
    tempMeni()
    x = eval(input("Unesite broj -> "))
    while x < 1 or x > 3:
        x = eval(input("Unesite broj -> "))
         choiceFlights(x)
```

Funkcija choiceFlights(x)

Funkcija kao parametar prima redni broj unete operacije koje su prikazane u funkciji tempMeni. Ova funkcija simulira “switch – case “ naredbu. Ukoliko korisnik unese 1, pozvaće se funkcije iz modula avioKarte kao što su loadFlights, sortFlights, printFlights. Ove funkcije bit će detaljno objašnjene kasnije.

```
def choiceFlights(x):
    if x == 1:
        loadFlights()
        sortFlights()
        printFlights()
        tempMeni()
        x = eval(input("Unesite opciju -> "))
        print("\n")
        choiceFlights(x)
    elif x == 2:
        loadFlights()
        reserveFlight()
    else:
        preglednik()
        n = eval(input("Unesite opciju -> "))
        choice(n)
```

Nakon odrađenih operacija službeniku se ponovo nudi izbor jedne od 3 opcije koje su prikazane funkcijom tempMeni(). Korisnik ponovo unosi opciju i rekurzivno se poziva metod choiceFlights ali sada za parametar x koji je unet ponovnim upitom.

Ukoliko službenik unese opciju 2, pozvaće se metode iz modula avioKarte koje će detaljno biti objašnjene kasnije.

Opcija 3 vraća službenika skroz na početak, odnosno vrši se logout službenika i nude se opcije za rad prikazane funkcijom preglednik().

Moramo se sada vratiti ponovo na funkciju choice. Ukoiliko je službenik pogrešio prilikom unosa nudi mu se varijanta da unese odgovor na pitanje sa da ili ne. Ukoliko je službenik zaboravio parametre za unos može ih se prisetiti tako što će se pozvati funkcija forgotUserOrPassword(). Pomenuta funkcija traži od službenika da unese svoj ID. Nakon unosa ID-a na ekran će se odštampati parametri za unos. Nakon toga poziva se metod choice sa parametrom 1 i vrši se ponovni upit korisnika za korisničko ime i sifru odnosno ponovo se pristupa login delu. Ukoliko je uneti ID pogrešan prekida se rad programa i ispisuje se prigodna poruka na ekran.

```
def forgotUserOrPassword():
    n = input("Unesite Vas ID -> ")
    for x in employee:
        if n == x.id:
            print("Vase korisnicko ime -> " , x.user)
            print("Vasa lozinka -> " , x.sifra)
            print("Vodite racuna o ovakvim stvarima! Sada se mozete ulogovati normalno\n")
            choice(1)
            return
    else:
        print("Uneti ID je pogresan. Obratite se Vasem sefu radi dobijanja novog ID-a \n")
         return
```

Ukoliko je kao odgovor uneto ’ne’, vrši se ponovo login sekcija.

U funkciji choice, ako je parametar 2, poziva se funkcija addEmployee koja dodaje novog službenika u listu i u fajl “službenici.txt“. Opis metoda nalazi se na sledećoj strani.

Funkcija addEmployee()

Kratak opis funkcije dat je na prethodnoj strani. Za pristup meniju dodavanja novog radnika neophodno je uneti administratorsku lozinku. Ukoliko je lozinka pogrešna ispisuje se poruka na ekran i prekida se rad programa. U suprotnom, administratoru se postavljaju upiti za novog radnika kao što su da unese ID novog radnika. Ako uneti ID već postoji kod nekog radnika u listi, ispisuje se poruka na ekran i prekida se rad programa. U suprotnom unose se ostali parametri kao što su ime, prezime, korisničko ime i lozinka novog radnika. Ako su svi parametri unešeni ispravno kreira se nov objekat klase Sluzbenik, dodaje se u fajl "sluzbenici.txt", potom u listu employee, sortiraju se radnici u listi i štampaju na ekran. Funkcija i baca izuzetak ukoliko se dogodi nešto neočekivano i poziva ponovo funkciju addEmployee i ponavlja se postupak. O tome da li postoji radnik sa unetim ID-om brine funkcija findWorker(ide) koja pretražuje listu radnika i proverava da li uneti ID već postoji u listi. Ako postoji vraća se vrednost True i onemogućava se dodavanje radnika sa unetim ID-om. U suprotnom slede koraci opisani gore. Kada se nov radnik upiše u fajl i u listu, poziva se metod za sortiranje radnika i na ekran se štampaju radnici sortirani po ID-u sa novim radnikom.

```
def findWorker(ide):
    for x in employee:
        if x.getId() == ide:
            return True
     return False
```

Da bi se dodani radnik upisao i u fajl poziva se funkcija addInFile sa parametrom "novi" koji predstavlja kreiran objekat klase Sluzbenik. Otvaramo fajl "sluzbenici.txt" u r+ režimu odnosno da na kraj fajla dodajemo dalje podatke. To postižemo pozivom funkcije read(). Najpre, moramo objekat „novi" prebaciti u string opis pozivom funkcije toString(novi). Potom, funkciji write kao parametre prosleđujemo string opis novog službenika. Nakon upisa zatvaramo fajl.

```
def addInFile(novi):
    fajl = open("sluzbenici.txt", "r+")
    fajl.read()
    fajl.write("\n" + toString(novi))
    fajl.close()
```

```
def toString(novi):
 return str(novi.id)+","+novi.ime+","+novi.prez+","+novi.user+","+novi.sifra
```

```
def addEmployee():
    print("Usli ste u Meni za dodavanje novog radnika. ")
    print("Za dalji rad Vam je neophodna sifra administratora sistema. \n")
    try:
        a = input("Unesite administratorsku lozniku -> ")
        if a == 'administratorAvioKarte9822':
            print("Uspesno ulogovan administrator. Sledite dalje korake.\n")
            ide = eval(input("Dodelite id novom radniku -> "))
            if findWorker(ide) == True:
                print("Zaposleni sa unetim id-om vec postoji!")
            else:
                ime = input("Unesite ime novog radnika -> ")
                prez = input("Unesite prezime novog radnika -> ")
                user = input("Dodelite mu korisnicko ime -> ")
                sifra = input("Dodelite mu lozinku -> ")
                novi = Sluzbenik(ide,ime,prez,user,sifra)
                addInFile(novi)
                employee.append(novi)
                print("Uspesno upisan nov radnik. Srecan pocetak posla. \n")
                sortEmployee()
                printEmployee()

        else:
            print("Greska u prijavi administratora.")
    except:
        print("Nesto je krenulo po zlu")
            addEmployee()
```

U funkciji choice opcija 3 rezervisana je za sortiranje i štampu radnika iz liste. Pomenute metode su već opisane.

Opcija 4 poziva već opisan metod forgotUserOrPassword, dok opcija 5 prekida rad programa.

*** KRAJ MODULA ***

2.3 Tekstualni fajl „letovi.txt“

Pomenuti fajl se koristi u drugom modulu pod nazivom “avioKarte.py“. Ovaj fajl sadrži informacije o letovima koje su u mogućnosti rezervacije. Za svaki let se pamti ID leta, polazište – dolazište i udaljenost ta 2 grada vazdušnom linijom izraženu u kilometrima.

```
JU501, Beograd - New York JFK, 7260
JU701, Beograd - Paris CDG, 1427
JU412, Beograd - Podgorica, 286
JU780, Beograd - Tivat, 312
JU500, Beograd - London Heathrow, 2123
JU100, Beograd - Moskva Sheremetyevo, 1406
JU250, Beograd - Berlin Tegel, 1252
JU632, Beograd - Roma Fiumicino, 795
JU465, Beograd - Milan Malpensa, 1063
JU341, Beograd - Atina, 683
```

2.4 Avio karte (modul)

Za potrebe ovog modula prvo je neophodno uvesti funkciju randint iz paketa random koji će nam kasnije biti neophodan.

```
From random import randint
```

Na samom početku modula nalazi se klasa Flight koja opisuje jedan let sa atributima istim kao i u tekstualnom fajlu (ID leta, polazište, dolazište, udaljenost). Svi atributi klase osim udaljenosti su predstavljeni kao string.

```
class Flight:

	def __init__(self, idL, start, end, distance):
		self.idL = idL
		self.start = start
		self.end = end
		self.distance = distance

	def toString(self):
		return "Let " + self.idL + ", relacija: " + self.start + " - " + self.end
		+ ", udaljenost: " + str(self.distance

	def toStringReverse(self):
		return "Let " + self.idL + ", relacija: " + self.end + " - " + self.start
		+ ", udaljenost: " + str(self.distance)

	def toStringJustDestRev(self):
		return self.end + " - " + self.start
```

Funkcija toString vraća opis leta od polazišta ka dolazištu. Funkcija toStringReverse vraća opis leta od dolazišta ka polazištu. Poslednja funkcija vraća destinaciju leta bez ID leta i udaljenosti.

U u klasi Flight se nalaze i još neke dodatne funkcije neophodne za rad.

```
def getIDL(self):
      return self.idL

def toStringJustDest(self):
      return self.start + " - " + self.end

def returnPriceReverseBussines(self):
      return self.distance * 0.50

def returnPriceReverseEconomic(self):
      return self.distance * 0.25

def returnPriceOneWayBussines(self):
      return self.distance * 0.6

def returnPriceOneWayEconomic(self):
      return self.distance * 0.28
```

Funkcija getIDL vraća ID leta. Ova funkcija će nam kasnije biti neophodna prilikom sortiranja letova.

Funkcija toStringJustDest vraća opis leta od polazišta ka dolazištu bez ID i udaljenosti.

Ostale funkcije se koriste prilikom računanja cene avio karte. Za cenu se uzima udaljenost polazišta i dolazišta i taj iznos se množi određenim koeficijentom. Za biznis klasu u povratnom pravcu udaljenost se množi sa 0.50, u jednom pravcu sa 0.6. Za ekonomsku klasu u povratnom pravcu udaljenost se množi sa 0.25, dok u jednom pravcu se množi sa 0.28.

Funkcija loadFlights()

Pre učitavanja letova pravimo praznu listu letova – flights = [],

praznu listu podataka data = [] i cenu avio karte (price) stavljamo na 0. Otvaramo fajl "letovi.txt" u režimu za čitanje I učitavamo letove slično kao i službenike. Prvo po delimiteru "zarez" učitavamo prvu i drugu liniju a potom i udaljenost. Budući da nam druga linija predstavlja relaciju koja je u fajlu upisana kao polazište – dolazište, neophodno je i drugu liniju podeliti po delimiteru "ravna crta" i dobiti 2 podatka. Tek nakon toga možemo pristupiti kreiranju objekta klase Flight. Nakon kreiranja objekta, let dodajemo u listu. Kada sve letove iz fajla dodamo u listu zatvaramo fajl i prekidamo učitavanje. Izgled funkcije dat je na sledećoj strani.

```
flights = []
data = []
price = 0

def loadFlights():

    fajl = open("letovi.txt", "r")
    for line in fajl:
        idl, druga, distance= line.split(",")
        start, end = druga.split("-")
       let = Flight(idl.strip(),start.strip(),end.strip(),
    int(distance.strip()))
         flights.append(let)
        fajl.close()
```

Funkcije sortFlights() i printFlights()

Slično kao što smo sortirali službenike tako ćemo i sortirati letove. Uvodimo lambda funkciju k koja sortira letove po ID-u. ID leta je jedinstven. Letove štampamo tako što prolazimo kroz listu letova i pozivamo metod toString(). Uvek prvo sortiramo letove pa ih tek onda štampamo.

```
def sortFlights():
    flights.sort(key = lambda k: k.getIDL(), reverse=False)

def printFlights():

    for line in flights:
        print(line.toString())
         print("*****udaljenost je izrazena u km***** \n")
```

Funkcija findFlight(ide)

Funkcija findFlight pretražuje i vraća let u listi koji ima isti ID sa unetim. Ukoliko ne postoji let sa unetim id-om vraća se NULL referenca.

```
def findFlight(ide):

    for line in flights:
        if str(line.idL) == str(ide):
            return line
      return NULL
```

Funkcija reserveFlight()

Ova funkcija zapravo predstavlja suštinu projekta. Od službenika se na početku zahteva da unese ID leta za koji želi rezervisati klijentu kartu. Nakon toga poziva se funkcija findFlight koja će za uneti ID pronaći odgovarajući let. Ukoliko funkcija vrati NULL od službenika će se očekivati da ponovo unese ID leta. U suprotnom, u listu podataka "data" na 0 poziciju upisuje se nađeni let. Zatim se službeniku na ekran ispisuju podaci o letu za koji se želi rezervisati karta. Nakon toga se unosi datum polaska i službenik je u obavezi da pita klijenta da li želi rezervisati kartu u jednom pravcu ili povratnu. Ukoliko korisnik želi povratnu kartu biće neophodno uneti i datum povratka. Da bismo sprečili službenika da unese datum povratka pre datuma polaska, moramo tokenizirati unete datume i izvršiti proveru da li je godina polaska manja od 2018. godine i da li je godina povratka manja od 2019. godine. Ukoliko je pogrešno unešen datum, svi uneti do sad podaci se anuliraju, lista data se anulira i službenik mora ponovo započeti proces rezervacije karte. U suprotnom u listu data dodajemo datum polaska i datum povratka. Sledeći upit je da li korisnik želi kartu za biznis ili ekonomsku klasu. Što god da izabere korisnik, izračunaće se cena karte pozivom odgovarajuće funkcije iz klase Flight i u listu data će se upisati odgovarajući string. Ukoliko službenik pogreši prilikom unosa, svi do sad uneti podaci se anuliraju i ponavlja se postupak rezervacije. Nakon odabira klase, na redu je i upit za koliko osoba se karta rezerviše. Službenik unosi broj a program proverava da li je uneti broj manji od nule ili veći od 10. Jedna osoba može rezervisati maksimalno deset karata. Ako je unos ispravan, cena karte se množi sa unetim brojem osoba, podaci se dodaju u listu data i poziva se metod dataReturn ili dataOneWay u zavisnosti od toga da li je putovanje povratno ili jednosmerno. U slučaju da je korisnik odabrao kartu u jednom pravcu postupak je isti samo se upit za datum povratka izostavlja i poziva se metod dataOneWay().

Oba metoda rade istu stvar ali zbog čitljivosti koda razdvojeni su u dva. Od službenika se zahteva unos ličnih podataka od korisnika kao što su ime i prezime. Nakon toga rezervacija će dobiti svoj jedinstveni trocifreni broj. Za to se brine funkcija randint iz paketa random. Funkcija dataReturn je vezana za povratna putovanja i nakon svih obrađenih podataka u ovom slučaju lista data mora imati veličinu 9. Ako je jednosmerno putovanje u pitanu dužina liste data mora biti 8.

Nakon unosa podataka i generisanja ličnog broja rezervacije, na ekran se ispisuju svi uneti podaci o rezervaciji kao što su tip putovanja, datum polaska, povratka, ime vlasnika itd... Odnosno čita se sve što piše u listi data. O ovome brine funkcija readData. Ako službenik potvrdi rezervaciju karte, poziva se funkcija confrim a zatim i prepareToWriting, nakon čega će se rezervacija upisati u fajl pod nazivom "rezervacije.txt". Ako se odustane od rezervacije prekida se rad programa. Na narednim stranama biće prikazane slike svih opisanih metoda.

```
def reserveFlight():

    print("Usli ste u meni za rezervaciju avio karte.")
    print("Vodite racuna o tacnosti podataka koje uzimate od klijenata.")
    ide = input("Unesite ID leta -> ")
    flight = findFlight(ide
    if flight != NULL:
        data.insert(0, flight)
        print("Zelite rezervisati kartu za let " + ide + ", "+flight.toStringJustDest()+"\n")
        print("*datume unosite u formatu DD-M-GG*")
        datP = input("Unesite datum polaska -> ")
        data.insert(1, datP)
        way = eval(input("Povratno(1) / jednosmerno(2) putovanje -> "))
        if way == 1:
            data.insert(2, "Povratno putovanje")
            datPov = input("Unesite datum povratka -> ")
            d, m, g = datP.split("-")
            d1, m1, g1 = datPov.split("-")
            if int(g) < 2018 or int(g1) < 2019:
                print("Neispravan format datuma. Unesite ponovo!!! \n")
                reserveFlight()
            data.insert(3, datPov)
            klasa = input("Klasa putovanja (b)Bussines, (e)Economic -> ")
            if klasa == "b":
                price = flight.returnPriceReverseBussines()
                data.insert(4, "Biznis klasa")
            elif klasa == "e":
                price = flight.returnPriceReverseEconomic()
                data.insert(4, "Ekonomska klasa")
            else:
                print("Pogresan unos!!! \n")
                reserveFlight()

            kol = eval(input("Koliko osoba putuje -> "))
            if kol < 0 or kol > 10:
                print("Broj osoba nije ispravno unesen \n")
                reserveFlight()
            else:
                data.insert(5, kol)
                price *= kol
                data.insert(6, price)
                dataReturn()

        else:
            klasa = input("Klasa putovanja (b)Bussines, (e)Economic -> ")
            data.insert(2, "Jednosmerno putovanje")
            if klasa == "b":
                price = flight.returnPriceOneWayBussines()
                data.insert(3, "Biznis klasa")
            elif klasa == "e":
                price = flight.returnPriceOneWayEconomic()
                data.insert(3, "Ekonomska klasa")
            else:
                print("Pogresan unos \n")
                reserveFlight()

            kol = eval(input("Koliko osoba putuje -> "))
            if kol < 0 or kol > 10:
                print("Broj osoba nije ispravno unesen \n")
                reserveFlight()
            else:
                data.insert(4, kol)
                price *= kol
                data.insert(5, price)
                dataOneWay()
    else:
        print("Unesen pogresan broj leta \n")
              reserveFlight()
```

Funkcije dataOneWay() i dataReturn()

Funkcije su već opisane, slede njihove slike:

```
def dataOneWay():
    ime = input("Ime vlasnika karte -> ")
    prez = input("Prezime vlasnika karte -> ")
    broj = randint(100, 1000)
    print("Jedinstveni broj karte -> " + str(broj))
    data.insert(6, ime)
    data.insert(7, prez)
    data.insert(8, broj)
    readData()

def dataReturn():
    ime = input("Ime vlasnika karte -> ")
    prez = input("Prezime vlasnika karte -> ")
    broj = randint(100, 1000)
    print("Jedinstveni broj karte -> " + str(broj))
    data.insert(7, ime)
    data.insert(8, prez)
    data.insert(9, broj)
    readData()
```

Funkcija readData()

Funkcija čita podatke iz liste data i ispisuje ih na ekran radi provere. Nakon ispisa poziva se funkcija confrim().

```
def readData():
    print("Proverite da li ste uneli sve podatke ispravno: \n")
    if data != NULL:
        if data[2] == "Povratno putovanje":
            print(data[0].toString())
            print(data[0].toStringReverse() + "\n")
            print("Tip putovanja -> Povratno putovanje")
            print("Datum polaska -> " + data[1])
            print("Datum povratka -> " + data[3])
            print("Klasa -> " + data[4])
            print("Putnika -> " + str(data[5]))
            print("Ukupna cena -> " + str(data[6]) + " $")
            print("Karta se rezervise na klijenta -> " + data[7]+" "+data[8])
            print("Jedinstveni broj karte -> " + str(data[9]) + "\n")
            confrim()
        else:
            print(data[0].toString() + "\n")
            print("Tip putovanja -> Jednosmerno putovanje")
            print("Datum polaska -> " + data[1])
            print("Klasa -> " + data[3])
            print("Putnika -> " + str(data[4]))
            print("Ukupna cena -> " + str(data[5]) + " $")
            print("Karta se rezervise na klijenta -> " + data[6]+" " + data[7])
            print("Jedinstveni broj karte -> " + str(data[8]) + "\n")
            confrim()
```

Funkcija confrim()

Ovde se od službenika očekuje da unese odgovor da ili ne, odnosno da li želi rezervisati kartu ili ne. Unosom da poziva se funkcija printInFile, unosom ne prekida se rad programa.

```
def confrim():

    n = input("Da li potvrdjujete rezervaciju karte -> ")
    if n == 'da':
      print("Uspesno rezervisana karta. Rezervaciju proverite u fajlu
      'rezervacije.txt' \n")
      printInFile(data)
    else:
        print("Kraj programa")
              return
```

Funkcije prepareToWriting(data) i printInFile(data)

Ova funkcija zadužena je za upis rezervacije u tekstualni fajl “rezervacije.txt“, ali pre upisa sve podatke iz liste data moramo pretvoriti u nekakav string. Za to nam je potrebna funkcija prepareToWritting koja kao rezultat vraća string formiran od podataka iz fajla. Funkcija je jedinstvena i radi i za povratna putovanja i za jednosmerna putovanja (ovo obezbeđujemo prostim upitom). Svaki podatak je upisan u novom redu u fajlu. Na kraj se dodaju ******* da bi označile kraj jedne rezervacije.

```
def prepareToWriting(data):
    string = "Relacija: "
    if data[2] == "Povratno putovanje":
        string += "Jedinstveni broj karte -> " + str(data[9]) + "\n"
        string += "Ime i prezime klijenta -> "+data[7]+" "+data[8]+"\n"
        string += data[0].toStringJustDest() + " < - > "
        string += data[0].toStringJustDestRev() + "\n"
        string += "Tip putovanja -> Povratno putovanje \n"
        string += "Datum polaska -> " + data[1] + "\n"
        string += "Datum povratka -> " + data[3] + "\n"
        string += "Klasa -> " + data[4] + "\n"
        string += "Karta se rezervise za -> "+str(data[5])+" putnika \n"
        string += "Ukupna cena -> " + str(data[6]) + " $ \n"
        string += "********************************** \n"
        return string
    else:
        string += "Jedinstveni broj karte -> " + str(data[8]) + "\n"
        string += "Ime i prezime klijenta -> "+data[6]+" "+data[7]+ "\n"
        string += data[0].toStringJustDest() + "\n"
        string += "Tip putovanja -> Jednosmerno putovanje \n"
        string += "Datum polaska -> " + data[1] + "\n"
        string += "Klasa -> " + data[3] + "\n"
        string += "Karta se rezervise za -> "+str(data[4])+" putnika \n"
        string += "Ukupna cena -> " + str(data[5]) + " $ \n"
        string += "********************************** \n"
        return string
```

```
def printInFile(data):

    fajl = open("rezervacije.txt", "r+")
    fajl.read()
    stri = prepareToWriting(data)
    fajl.write(stri)
    fajl.close()
```

***** KRAJ MODULA ****

2.6 Tekstualni fajl "rezervacije.txt"

```
Rezervacije:
Relacija: Jedinstveni broj karte -> 239
Ime i prezime klijenta -> andreja simic
Beograd - Tivat < - > Tivat - Beograd
Tip putovanja -> Povratno putovanje
Datum polaska -> 30-12-2018
Datum povratka -> 7-1-2019
Klasa -> Biznis klasa
Karta se rezervise za -> 3 putnika
Ukupna cena -> 468.0 $
**********************************
Relacija: Jedinstveni broj karte -> 840
Ime i prezime klijenta -> dimitrije djukic
Beograd - London Heathrow < - > London Heathrow - Beograd
Tip putovanja -> Povratno putovanje
Datum polaska -> 2-1-2019
Datum povratka -> 2-3-2019
Klasa -> Ekonomska klasa
Karta se rezervise za -> 3 putnika
Ukupna cena -> 1592.25 $
**********************************
```

2.7 Main modul

Na početku iz paketa Seminarski modula sluzbenici uvozimo sledeće funkcije:

```
from Seminarski.sluzbenici import loadEmployee, preglednik,
                                  choice
```

Nakon toga u main funkciji prvo se štampa poruka da smo u sistemu rezervacije avio karata. Dalje, pozivaju se metode za učitavanje službenika i preglednik. Očekuje se unos opcije u rasponu od 1 do 5. Ako se unese pogrešan broj postupak će se ponavljati. Main funkcija može baciti izuzetak ako službenik unese neko slovo umesto cifre. Posle ispravno unešenog broja poziva se funkcija choice iz modula službenici i dalje se program odvija. Izgled modula dat je na sledećoj strani.

```
def main():
    print("=============================================")
    print("=       REZERVACIJE AVIO KARATA             =")
    loadEmployee()
    preglednik()
    try:
        n = eval(input("Unesite opciju -> "))
        while n < 1 or n > 5:
            n = eval(input("Unesite opciju -> "))
        choice(n)
    except:
        print("Uneli ste slovo umesto cifre")


if __name__ == '__main__':
        main()
```

3. Zaključak

Ovim projektom omogućili smo rezervisanje avio karti, dodavanje novih radnika i njihovo sortiranje, pregled. Sve neophodne funkcije i sadržaj opisano je u drugom poglavlju.

Prikazan i opisan program se može dalje koristiti i unapređivati raznim potrebama agencija koje se bave rezervacijom avionskih karata.

4. Literatura

1. Materijal sa predavanja predmeta "Script jezici" profesorica Jovana Vidaković
2. https://sr.wikipedia.org/sr-ec/ Пајтон_(програмски_језик)
3. http://mef-lab.com/osnove-2016/book/index.html