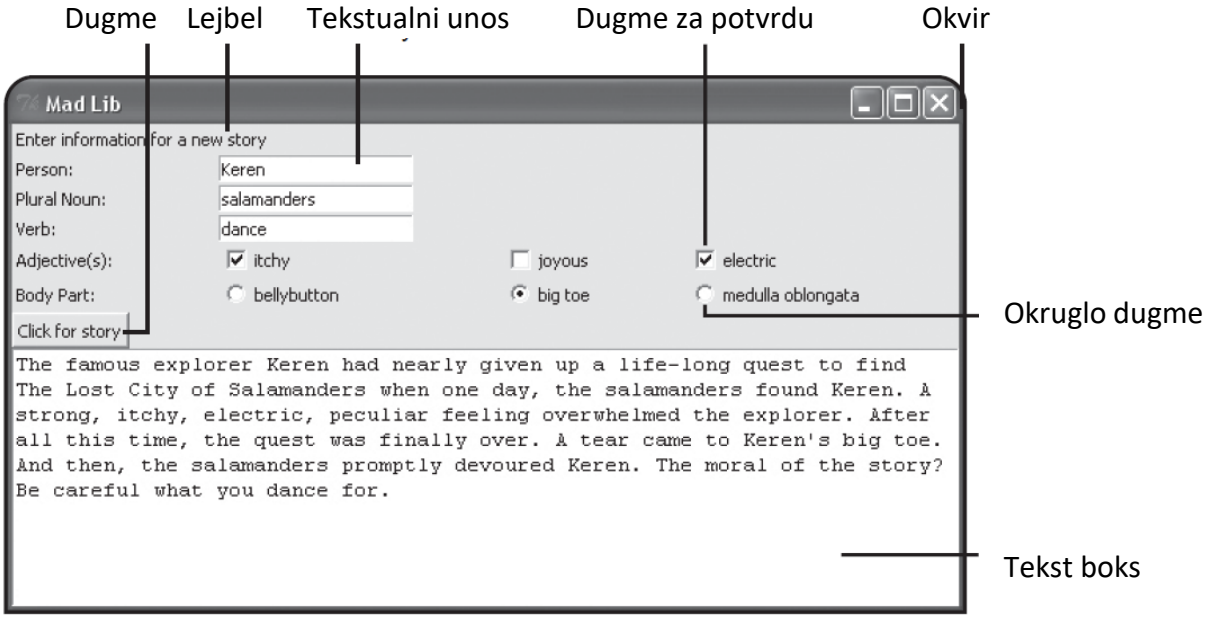
Kurs: Programiranje igara sa programskim jezikom Pajton
025 čas: Program Brojac_Klikova.py
Teme: tkinter modul, GUI alati, klase i GUI, dugmad

Razvoj GUI (Graphical User Interface – Korisnički grafički interfejs)

Do sada svi programi su koristili osnovni oblik teksta za interakciju sa korisnikom. Ali, postoje i sofisticiraniji načini predstavljanja i primanja informacija. GUI omogućava vizuelni način za korisnika da interaguje sa kompjuterom. Svi OS koriste GUI, čime je komunikacija sa korisnikom jednostavnija i jasnija. Ovde se prikazuje kreiranje korisničkog grafičkog interfejsa.

001 Delovi GUI

Ovde će se koristiti sledeći GUI elementi:



Da bi se mogao kreirati GUI, koriste se GUI alati (toolkit). Najpopularniji GUI alat je Tkinter, međuplatformski alat.

GUI elementi se kreiraju instancijacijom objekata iz klasa unutar tkinter modula, koji je deo Tkinter alata. Sledeća tabela opisuje svaki od GUI elemenata i odgovarajuću tkinter klasu.

Element	**tkinter**	**Opis klase**
Okvir	Frame	Sadrži druge GUI elemente
Lejbel	Label	Prikazuje nepromenjiv tekst ili ikone
Dugme	Button	Izvodi akciju koju korisnik aktivira
Tekst unos	Entry	Prihvata i prikazuje liniju teksta
Tekst boks	Text	Prihvata i prikazuje više linija teksta
Dugme za potvrdu	Checkbutton	Korisnik može ili nemora izabrati neku opciju
Okruglo dugme	Radiobutton	Izbor jedne opcije od više ponuđenih

002 Osnove Programiranja vođenog događajem (event-driven)

GUI programi su tradicionalno vođeni događajem, što znači da odgovaraju na akcije bez obzira na redosled kojim su postavljene. Programiranje vođeno događajem je donekle drugačiji način razmišljanja o kodovima.

Korišćenjem konzolnog pristupa, programer bi koristio niz input() funkcija za dobijanje odgovora od korisnika. Korisnik bi odgovarao po redosledu na pitanja kako bi se ona i

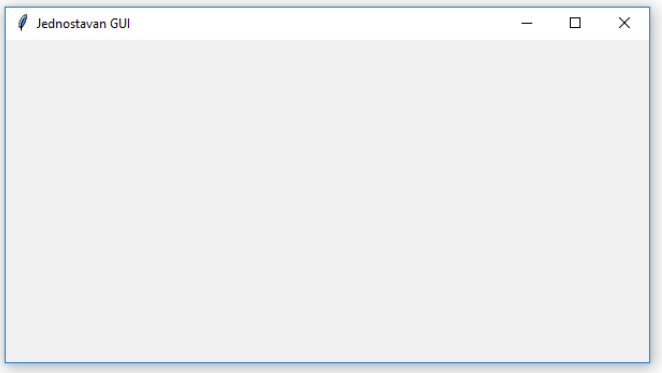
pojavljivala na konzoli. Ali, kada bi isti kod bio napisan u GUI, korisnik bi mogao unositi podatke po bilo kojem redosledu.
Kada se piše program vođen događajem, programer povezuje (bind) događaje (stvari koje se dese a uključuju objekte u programu) sa menadžerima događaja (event handlers – kod koji se startuje kada se desi nekakav događaj). Kada korisnik klikne na dugme (događaj), program podiže metod koji prikazuje tekst (menadžer događaja), a da bi se to i desilo mora se povezati klik na dugme sa metodom prikaza teksta.
Definisanjem svih objekata, događaja i menadžera događaja, programer definiše rad samog programa. Zatim se program startuje ulaskom u petlju događaja (event loop), gde program čeka da se desi događaj. Kada se desi bilo koji od definisanih događaja, program rukuje njima.

003 Korišćenje osnovnog (root) prozora

Osnova GUI programa je njegov osnovni prozor, na koji se dodaju svi ostali GUI elementi. Ako se GUI predstavi kao drvo, onda je osnovni prozor njegov koren.

Jednostavan GUI program

Rezultat ovog programa treba da bude:



Mogući su problemi ako se Tkinter startuje direktno iz IDLE, zato je možda bolje direktno startovati Tkinter program. Ali, ako iz bilo kog razloga ne funkcioniše Tkinter, brzo će se pojaviti i nestati konzolni prozor pre mogućnosti čitanja poruke. Sem prikazanog prozora Tkinter kreira i konzolni prozor. Konzolni prozor može da pruži značajne informacije ako se pojave greške u radu Tkintera, i zato ne treba zatvarati konzolu dok radi Tkinter prozor.

004 Importovanje tkinter modula

Prvo se importuje tkinter modul:

```
#Laki_GUI
#Demonstrira kreiranje prozora

from tkinter import *
```

Na ovaj način se importuje sve iz tkinter direktno u globalni doseg programa.
Obično se ovo ne radi; ipak, nekoliko modula poput tkinter je dizajnirano da se importuju na ovaj način.

005 Kreiranje osnovnog prozora

Da bi se kreirao osnovni prozor, instancira se objekat tkinter klase Tk:

```
#kreira osonovni prozor
osnovni = Tk()
```

Vidi se da nije bilo potrebe da se napiše ime modula kao prefiks, tkinter, na ime klase, Tk. Sada se može direktno prići bilo kojem delu tkinter modula, bez potrebe sa korišćenjem i imena

modula. Pošto većina tkinter programa uključuje mnoge reference prema klasama i konstantama u modulu, ovo olakšava kucanje i čini kod lakšim za čitanje.
Na ovaj način je napravljen osnovni (root) prozor u programu. U jednom programu može biti samo jedan osnovni prozor; kreiranje više osnovnih prozora bi učinilo da se oni međusobno bore za kontrolu nad događajima.

006 Modifikovanje osnovnog prozora

Modifikovanje osnovnog prozora pomoću nekoliko njegovih metoda:

```
osnovni.title("Jednostavan GUI")
osnovni.geometry("200x100")
```

Metoda title() postavlja naslov osnovnog prozora. Samo se preda naslov prikazan kao string. Naslov će se pojaviti kao tekst na naslovnoj traci prozora.
Metoda geometry() postavlja veličinu osnovnog prozora, u pikselima. Metoda uzima string (ne cele brojeve) i predstavlja širinu i visinu prozora, odvojenih sa “x“ karakterom. Ovde su oni postavljeni na 200 piksela širine i 100 piksela visine.

007 Ulazak u petlju događaja (event loop) osnovnog prozora

Petlja događaja prozora se startuje pozivom metode mainloop():

```
#startuje petlju dogadjaja prozora
osnovni.mainloop()
```

Kao rezultat, prozor ostaje otvoren, čekajući da reaguje na događaje. Pošto nijedan događaj još uvek nije definisan, prozor ništa i ne radi. Ali je to potpuno korektan prozor kojem se može menjati veličina, koji se može minimizirati i zatvoriti.

008 Korišćenje lejbela

GUI elementi se nazivaju vidžeti (widgets – skraćeno od window gadgets). Najjednostavniji vidžet je lejbel vidžet, koji je tekst ili ikona koi se ne mogu editovati. Lejbel vidžet označava deo HUI. Često se koristi za označavanje drugih vidžeta. Za razliku od drugih vidžeta, lejbeli nisu interaktivni. Korisnik može kliknuti na lejbel ali neće biti nikakve reakcije. Ipak, lejbeli su važni i uvek se koriste u GUI.

program Lejbelic.py

Ovaj program kreira osnovni prozor i dodaje lejbel. Vidžet lejbel jednostavno deklariše da je on lejbel.

```
#Lejbelic
#Prikazuje rad sa lejbelima
from tkinter import *
#kreira osnovni prozor
osnovni = Tk()
osnovni.title("Lejbelic")
osnovni.geometry("300x150")
#kreira okvir u prozoru za smestanje drugih vidzeta
okvir = Frame(osnovni)
okvir.grid()
#kreira lejbel u okviru
lejbel = Label(okvir, text = "Ja sam lejbel!")
lejbel.grid()
#start petlje dogadjaja prozora
osnovni.mainloop()
```

009 Postavljanje programa

Prvo se postavi program Lejbelic importovanjem tkintera i kreiranjem osnovnog prozora:

```
#Lejbelic
#Prikazuje rad sa lejbelima
from tkinter import *
```

```
#kreira osnovni prozor
osnovni = Tk()
osnovni.title("Lejbelic")
osnovni.geometry("300x150")
```

010 Kreiranje okvira (frame)

Okvir je vidžet koji može sadržati druge vidžete (poput lejbel vidžeta). Okvir je poput plute na oglasnoj tabli, koristi se kao osnova na koju se stavljaju druge stvari. Ovako se kreira novi okvir:

```
#kreira okvir u prozoru za smestanje drugih vidzeta
okvir = Frame(osnovni)
```

Kada se kreira novi vidžet, mora se predati njegov master (stvar u kojoj će se nalaziti vidžet) konstruktoru novog objekta. Ovde, predat je osnovni konstruktoru Frame. Kao rezultat, novi okvir je smešten unutar osnovnog prozora.

Sledeće, poziva se grid() metoda novog objekta:

okvir.grid()

Metoda grid() je metoda koju svi vidžeti imaju. Ona je povezana sa menadžerom prikaza (layout manager) koji omogućava aranžiranje vidžeta.

011 Kreiranje lejbela

Vidžet lejbel se kreira instnciranjem objekta klase Label:

```
#kreira lejbel u okviru
lejbel = Label(okvir, text = "Ja sam lejbel!")
```

Predavanjem okvir u Label konstruktor objekta, kreira se okvir kojim okvir ukazuje na master od lejbel vidžeta. Kao rezultat, lejbel je smešten u okvir.

Vidžeti imaju opcije koje s mogu postavljati. Mnoge od ovih opcija utiču na to kako se vidžeti prikazuju. Predavanjem stringa `"Ja sam lejbel!"` parametru text, postavlja se opcija text vidžeta na taj string. Kao rezultat, tekst će se pojaviti kada se lejbel prikaže.

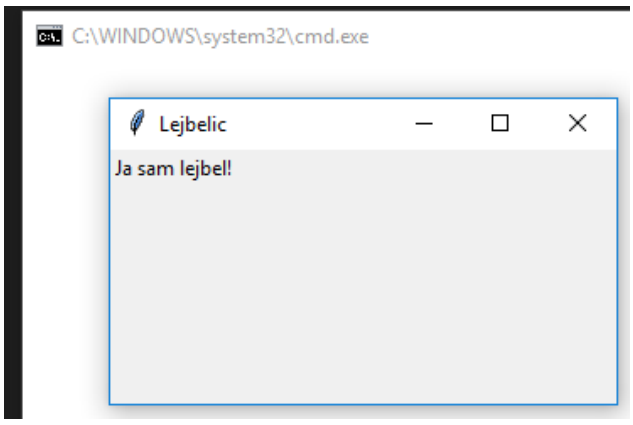
Dalje, poziva se grid() metoda objekta: lejbel.grid()

što omogućava da lejbel bude vidljiv.

012 Startovanje petlje događaja osnovnog prozora

Poziva se petlja događaja osnovnog prozora da pokrene GUI:

```
#start petlje dogadjaja prozora
osnovni.mainloop()
```



Upotreba dugmeta

Vidžet dugme se može aktivirati od strane korisnika da bi se proizvela neka akcija.

program Dugmici_Lencuge.py

U ovom programu su kreirani nekoliko dugmadi koji ništa ne rade kada se aktiviraju.

```
#Dugmici_Lencuge
#Prikazuje kreiranje dugmica
from tkinter import *
#kreiranje osnovnog prozora
osnovni = Tk()
osnovni.title("Dugmici Lencuge")
osnovni.geometry("300x150")
#kreira okvir u prozoru za drzanje drugih vidzeta
okvir = Frame(osnovni)
okvir.grid()
#kreira dugme u okviru
dugme1 = Button(okvir, text = "Ja ne radim nista!")
dugme1.grid()
#kreira drugo dugme u okviru
dugme2 = Button(okvir)
dugme2.grid()
dugme2.configure(text = "Ni ja!")
#kreira trece dugme u okviru
dugme3 = Button(okvir)
dugme3.grid()
dugme3["text"] = "Kao ni ja!"
#petlja dogadjaja prozora
osnovni.mainloop()
```

013 Postavka programa

Program se postavlja importovanjem tkinter i kreiranjem osnovnog prozora i okvira:

```
#Dugmici_Lencuge
#Prikazuje kreiranje dugmica
from tkinter import *
#kreiranje osnovnog prozora
osnovni = Tk()
osnovni.title("Dugmici Lencuge")
osnovni.geometry("300x150")
#kreira okvir u prozoru za drzanje drugih vidzeta
okvir = Frame(osnovni)
okvir.grid()
```

014 Kreiranje dugmeta

Kreira se vidžet dugme instancijacijom objekta klase Button:

```
#kreira dugme u okviru
dugme1 = Button(okvir, text = "Ja ne radim nista!")
dugme1.grid()
```

Ovaj kod kreira novo dugme sa tekstom na njemu. Master dugmeta je okvir ranije kreiran, što znači da je dugme smešteno u okvir.

Modul tkinter omogućava fleksibilnost kada se govori o kreiranju, definisanju i izmeni vidžeta. Može se kreirati vidžet i postaviti sve njegove opcije u jednoj liniji ili se može kreirati vidžet i postaviti ili izmeniti njegove opcije kasnije.

Kreira se novo dugme:

```
#kreira drugo dugme u okviru
dugme2 = Button(okvir)
dugme2.grid()
```

Primetiti da jedina vrednost koja se predaje konstruktoru objekta je okvir, master dugmeta. Na ovaj način se dodaje prazno dugme na okvir. Ipak, može se modifikovati vidžet posle kreiranja korišćenjem metode configure():

```
dugme2.configure(text = "Ni ja!")
```

Ova linija postavlja text opciju dugmeta na string `"Ni ja!"` , koji ga smešta na dugme.

Može se koristiti metoda configure() za bilo koju vidžet opciju (i bilo koji tip vidžeta). Čak se može koristiti ova metoda i za promenu opcije koja se prethodno postavila.
Kreira se i treće dugme:

```
#kreira trece dugme u okviru
dugme3 = Button(okvir)
dugme3.grid()
```

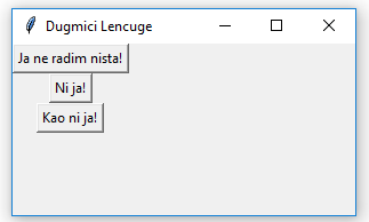
Zatim, postavi se text opcija, korišćenjem različitog interfejsa:

```
dugme3["text"] = "Kao ni ja!"
```

Pristupa se text opciji kroz interfejs poput rečnika. Postavlja se text opcija na string `"Kao ni ja!"` , koji smešta taj tekst na dugme. Kada se postavi vrednost opcije korišćenjem ovog tipa pristupa nalik rečniku, ključ za opciju je ime opcije poput stringa.

015 Petlja događaja

```
osnovni.mainloop()
```



Kreiranje GUI korišćenjem klase

Organizovanje koda u klase pomaže programiranje. Često je korisno pisati veće GUI programe definisanjem sopstvenih klasa.

program Dugmici_Lencuge_2.py

Ovo je isti kod kao prethodni samo što se koriste klase:

```
#Dugmici_Lencuge_2
#Prikazuje kreiranje dugmica sa klasama
from tkinter import *
class Aplikacija(Frame):
    """ GUI aplikacija sa tri dugmeta. """
    def __init__(self, master):
        """ Inicijalizuje Frame. """
        super(Aplikacija, self).__init__(master)
        self.grid()
        self.kreira_vidzete()

    def kreira_vidzete(self):
        """ Kreira tri dugmeta koja ne rade nista. """
        #kreira prvo dugme
        self.dugme1 = Button(self, text = "Ja ne radim nista! ")
        self.dugme1.grid()
        #kreira drugo dugme
        self.dugme2 = Button(self)
        self.dugme2.grid()
        self.dugme2.configure(text = "Ni ja!")
        #kreira trece dugme
        self.dugme3 = Button(self)
        self.dugme3.grid()
        self.dugme3["text"] = "Ja takodje! "

#main
osnovni = Tk()
osnovni.title("Dugmici Lencuge 2")
osnovni.geometry("300x150")
okvir = Aplikacija(osnovni)
osnovni.mainloop()
```

016 Definisanje klase aplikacije

Kreiranje nove klase, Aplikacija, bazirane na Frame:

```
class Aplikacija(Frame):
    """ GUI aplikacija sa tri dugmeta. """
```

Umesto da se instancira objekat Frame, instancijacija se vrši nad objektom Aplikacija koji će sadržati sve dugmiće. Ovo će funkcionisati pošto je objekat Aplikacija samo specijalan tip objekta Frame.

017 Definisanje metoda konstruktora

Definiše se konstruktor od Aplikacija:

```
def __init__(self, master):
        """ Inicijalizuje Frame. """
        super(Aplikacija, self).__init__(master)
        self.grid()
        self.kreira_vidzete()
```

Prvo se poziva konstruktor superklase. Dodaje se master objekta Aplikacija, pa se pravilno postavlja kao master. Na kraju se podiže metoda kreira_vidzete() objekta Aplikacija koja se sledeća definiše.

018 Definisanje metode za kreiranje vidžeta

Definiše se metoda koja kreira sva tri dugmeta, kreira_vidzete():

```
def kreira_vidzete(self):
        """ Kreira tri dugmeta koja ne rade nista. """
        #kreira prvo dugme
        self.dugme1 = Button(self, text = "Ja ne radim nista! ")
        self.dugme1.grid()
        #kreira drugo dugme
        self.dugme2 = Button(self)
        self.dugme2.grid()
        self.dugme2.configure(text = "Ni ja!")
        #kreira trece dugme
        self.dugme3 = Button(self)
        self.dugme3.grid()
        self.dugme3["text"] = "Ja takodje! "
```

Kod je sličan kodu kada se kreiraju dugmad u prvom kodu. Bitna razlika je to što dugme1, dugme2 i dugme3 su atributi objekta Aplikacija. Druga razlika je što se koristi self kao master za dugmad tako da je objekat Aplikacija njihov master.

019 Kreiranje objekta Aplikacija

U glavnom delu koda, kreira se osnovni prozor, daje mu se naslov i veličina:

```
#main
osnovni = Tk()
osnovni.title("Dugmici Lencuge 2")
osnovni.geometry("300x150")
```

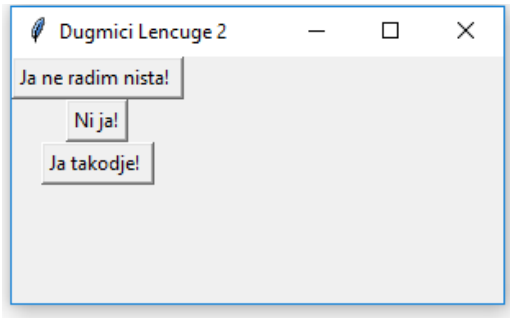
Zatim se instancira objekat Aplikacija sa osnovnim prozorom kao njegovim masterom:

```
okvir = Aplikacija(osnovni)
```

Kod kreira objekat Aplikacija sa osnovnim prozorom kao njegovim masterom. Konstruktor objekta Aplikacija uzdiže objektovu metodu kreira_vidzete(). Ova metoda zatim kreira tri dugmeta, sa objektom Aplikacija kao njihovim masterom.

Na kraju, uzdiže se petlja događaja osnovnog prozora koja startuje GUI:

```
osnovni.mainloop()
```



Povezivanje vidžeta i menadžera događaja

Do sada, GUI programi nisu ništa suviše korisno radili. To je zato što nije postojao kod koji bi bio pridružen sa aktiviranjem njihovih vidžeta; što će se sada i uraditi.

program Brojac_Klikova.py

Ovaj program ima dugme koje ispisuje broj puta koliko je korisnik kliknuo na dugme. Tehnički, menadžer događaja vodi računa o updejtovanju brojanja klikova i menja tekst na dugmetu:

```
#Brojac_Klikova
#prikazuje povezivanje dogadjaja sa menadzerom dogadjaja
from tkinter import *
class Aplikacija(Frame):
    """ GUI aplikacija koja broji klikove na dugmetu. """
    def __init__(self, master):
        """ Inicijalizacija okvira """
        super(Aplikacija, self).__init__(master)
        self.grid()
        self.dugme_klikovi = 0 #broj klikova na dugme
        self.kreira_vidzete()

    def kreira_vidzete(self):
        """ Kreira dugme koje prikazuje broj klikova. """
        self.dugme = Button(self)
        self.dugme["text"]= "Ukupno klikova: 0"
        self.dugme["command"] = self.updejt_brojaca
        self.dugme.grid()

    def updejt_brojaca(self):
        """ Poveca vrednost brojaca klika i prikaze je. """
        self.dugme_klikovi += 1
        self.dugme["text"] = "Ukupno klikova: " + str(self.dugme_klikovi)

# main
osnovni = Tk()
osnovni.title("Brojac Klikova")
osnovni.geometry("300x150")
okvir = Aplikacija(osnovni)
osnovni.mainloop()
```

020 Postavka programa

Importovanje GUI modula:

```
#Brojac_Klikova
#prikazuje povezivanje dogadjaja sa menadzerom dogadjaja
from tkinter import *
```

Sledi definicija klase Aplikacija:

```
class Aplikacija(Frame):
    """ GUI aplikacija koja broji klikove na dugmetu. """
    def __init__(self, master):
        """ Inicijalizacija okvira """
        super(Aplikacija, self).__init__(master)
        self.grid()
```

```
        self.dugme_klikovi = 0 #broj klikova na dugme
        self.kreira_vidzete()
```

Ovde je nova linija `self.dugme_klikovi = 0` kojom se kreira atribut objekta koji vodi evidenciju o broju klika na dugme.

021 Povezivanje menadžera događaja

U kreira_vidzete() metodi, kreira se jedno dugme:

```
def kreira_vidzete(self):
        """ Kreira dugme koje prikazuje broj klikova. """
        self.dugme = Button(self)
        self.dugme["text"]= "Ukupno klikova: 0"
        self.dugme["command"] = self.updejt_brojaca()
        self.dugme.grid()
```

Postavljena je opcija command, Button vidžeta na metodu updejt_brojaca(). Kao rezultat, kada korisnik klikne na dugme, poziva se metod. Tehnički, ono što je urađeno je povezivanje događaja (klik Button vidžeta) sa menadžerom događaja (metoda updejt_brojaca()).

Uopšteno, postavlja se vidžet opcija command za povezivanje aktivacije vidžeta sa menadžerom događaja.

022 Kreiranje menadžera događaja

Zatim, piše se metoda updejt_brojaca(), koja upravlja događajem klika na dugme:

```
def updejt_brojaca(self):
        """ Poveca vrednost brojaca klika i prikaze je. """
        self.dugme_klikovi += 1
        self.dugme["text"] = "Ukupno klikova: " + str(self.dugme_klikovi)
```

Metod inkrementira ukupan broj klikova na dugme a zatim menja tekst na dugmetu kojim se prikazuje nova vrednost.

Ostatak koda:

```
# main
osnovni = Tk()
osnovni.title("Brojac Klikova")
osnovni.geometry("300x150")
okvir = Aplikacija(osnovni)
osnovni.mainloop()
```

Kreiran je osnovni prozor i postavljen njegov naslov i dimenzije. Zatim je instanciran nov objekat Aplikacija sa osnovnim prozorm kao njegovim masterom.