**UNIVERZITET U KRAGUJEVCU**

**FAKULTET TEHNIČKIH NAUKA ČAČAK**

****

**SEMINARSKI RAD**

Tema: Objektno orijentisano projektovanje

Analiza, dizajn i izrada softvera za manipulaciju jednostavnih poligona

**Studijski program:** Informacione tehnologije 2017

**Predmet:** Objektno orijentisano programiranje

**Student:** Veljko Lončarević

**Indeks:** 64/2019

**Profesor Asistent**

Prof. dr Vlade Urošević Katarina Mitrović

*Sadržaj*

[*Uvod 3*](#_Toc62116301)

[*Definicija poligona 4*](#_Toc62116302)

[*Planirane funkcionalnosti 5*](#_Toc62116303)

[*Unos podataka 5*](#_Toc62116304)

[*Validacija podataka 5*](#_Toc62116305)

[*Provera složenosti poligona 6*](#_Toc62116306)

[*Iscrtavanje poligona na ekranu 7*](#_Toc62116307)

[*Matematički proračuni – obim, površina 8*](#_Toc62116308)

[*Izmena i čuvanje podataka 10*](#_Toc62116309)

[*Use Case dijagram 11*](#_Toc62116310)

[*Dijagram aktivnosti 12*](#_Toc62116311)

[*Reference i tabela figura 13*](#_Toc62116312)

[*Reference/Literatura 13*](#_Toc62116313)

[*Tabela figura 13*](#_Toc62116314)

# Uvod

Mnogi veruju da je matematika temelj i osnova računarskih nauka. Takođe bi se moglo reći i da je fundamentalan intelektualni alat u računarstvu, ali i računarstvo se sve više koristi u rešavanju matematičkih problema. Diskretna matematika, bulova algebra, kombinatorika, logičke notacije, teorije redova, grafova, verovatnoće, brojeva, algebra, sve su to grane matematike koje su u potpunosti neophodne računarskim naukama.

Jedan od osnovnih delova računarskih nauka predstavljaju algoritmi. Nizovi instrukcija koji demonstriraju implementaciju računarskog programa. Međutim, naš prvi susret s algoritmima nije bio prilikom rada u programskom jeziku C, već mnogo ranije – kada smo tek učili da brojimo. Sabiranje, oduzimanje, množenje i deljenje prirodnih brojeva neki su od najjednostavnijih algoritama s kojima smo se susretali na samom početku našeg obrazovanja. Matematika je, dakle, neophodna za razvoj intelektualnih sposobnosti i osposobljavanje osoba za razumevanje i upotrebu algoritama koji će se kasnije koristiti u računarskim naukama, kao i za razvijanje analitičkih veština koje ćemo upotrebiti i u ovom seminarskom radu, prilikom analize objektno orijentisanog projekta.

Većina grana računarskih nauka kao što su operativni sistemi, baze podataka, veštačka inteligencija, mašinsko učenje, umrežavanje, ugrađeni sistemi i mikrokontroleri, računarska grafika, zahtevaju određen nivo poznavanja matematičkih koncepata kao što su linearna algebra, vektori, matrice, trigonometrija..

Kao jedan primer toga naveo bih korišćenje matematičkih koncepata tačke, linije i poligona u računarskoj grafici. Za crtanje dvodimenzionalnih objekata (geometrijskih figura) na ekranu moramo znati sledeće podatke o tom objektu:

1. Koliko sadrži temena (tačaka),
2. Koja od tih temena su povezana (linijama),
3. Da li će geometrijska figura sadržati dodatne atribute (Boja, providnost, nivo prikaza..)

Upravo iz tog razloga odlučio sam se za demonstraciju važnosti matematike u računarskim naukama, tako što će tema ovog seminarskog rada biti **Analiza, dizajn i izrada softvera za manipulaciju jednostavnih** **poligona**.

U daljem tekstu, kao i samom programu, obradićemo sledeće funkcije:

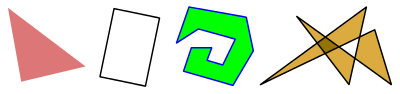
1. Učitavanje podataka (korisnički unos ili unos iz datoteke) o poligonu,
2. Provere ispravnosti podataka, složenosti poligona i kolinearnosti tačaka,
3. Iscrtavanje poligona na predviđenom prostoru u programu,
4. Matematički proračuni o poligonu (površina, obim, uglovi..),
5. Manipulacija i izmena podataka o poligonu,
6. Čuvanje podataka u vidu datoteke na memoriji računara.

# Definicija poligona

Poligon u geometriji predstavlja planarnu figuru koju opisuje konačan broj segmenata pravih linija koje su povezane i formiraju zatvorenu figuru koja se naziva poligonalna mreža (granica). Poligonalna mreža, zajedno s delom ravni unutar nje, čine jedan poligon.

Segmenti poligonalne mreže nazivaju se njene ivice ili strane (eng. *edge*), a tačke gde se dve ivice susreću su temena (eng. *vertex*). Unutrašnjost poligona nekada se naziva i njegovo telo (eng. *body*). N-ugao jeste poligon s *n* brojem stranica; na primer, trougao je poligon s 3 stranice.

Jednostavan poligon jeste onaj čije se ivice ne presecaju, dok su složeni oni čije se strane presecaju. Primer takvih poligona možete pronaći na slici ispod.



Slika 1:Razne vrste poligona

**Konveksni i nekonveksni poligoni**

Na osnovu svoje (ne)konveksnosti, poligoni se mogu grupisati na sledeće načine:

* **Konveksni** – Bilo koja linija nacrtana kroz poligon (i koja nije tangenta ivici ili temenu) seče poligonsku mrežu u tačno dve tačke. Svi unutrašnji uglovi su manji od 180°.
* **Nekonveksni** – Može se nacrtati takva linija koja seče poligonsku mrežu u više od dve tačke.
* **Jednostavni** – Ivice poligona se međusobno ne seku. Svi konveksni poligoni su analogno i jednaki.
* **Konkavni** – Nekonveksni, jednostavni poligoni. Postoji makar jedan unutrašnji ugao veći od 180°.
* **U obliku zvezde** – Cela unutrašnjost vidljiva je barem iz jedne tačke, bez prelaska preko neke od ivica. Poligon mora biti jednostavan, može biti konveksan ili konkavan. Svi konveksni poligoni su u obliku zvezde.
* **Kompleksni** – Neke od ivica poligona se međusobno seku.
* **Zvezde** – Poligon čije se ivice međusobno **pravilno** seku. Poligon ne može u isto vreme biti i zvezda i u obliku zvezde (Zvezdolik).

Takođe se mogu podeliti na osnovu simetričnosti i jednakosti uglova:

* **Jednakougaoni** – Poligon ima jednake uglove.
* **Jednakostranični** – Poligon ima jednake stranice. Ne mora biti konveksan.
* **Cikličan** – Sva temena leže na jednom istom opisanom krugu.
* **Izogonalni –** Sva temena leže na istoj orbiti simetrije. Poligon takođe mora biti cikličan i jednakostranični.
* **Tangentan –** Sve ivice su tangente opisanom krugu.
* **Izotokaslni –** Sva temena leže na istoj orbiti simetrije. Poligon takođe mora biti tangentan i jednakostranični.
* **Pravilan –** Poligon je i izogonalan i izotoksalan. To znači da je i cikličan i jednakostraničan, ili i jednakostraničan i jednakougaoni. Nekonveksan pravilan poligon naziva se i pravilan poligon zvezda.

U ovom seminarskom radu nećemo se baviti kompleksnim poligonima, zbog njihove matematičke kompleksnosti, već ćemo se fokusirati na jednostavne.

# Planirane funkcionalnosti

## Unos podataka

Kako bismo mogli manipulisati s određenim poligonom i analizirati ga, potrebno je prvo uneti podatke o njemu. U poglavlju „Definicija poligona“ naveli smo šta je sve potrebno od podataka za jedan poligon, ali mi ćemo pojednostaviti ovaj model podataka i radićemo samo s temenima. Ako obeležimo početno teme s A0, a poslednje teme s An-1, biće nam potreban niz tačaka (temena) poligona, koje će redom biti povezane, i na kraju će poslednje teme An-1 biti povezano s početnim A0.

Temena će se unositi sekvencijalno putem korisničkog interfejsa, i čuvati u vidu skupa tačaka u obliku (x,y). Skup unesenih tačaka biće sačuvan u posebnoj listi predviđenoj za tu namenu.

U grafičkom korisničkom interfejsu isplanirana su posebna polja za ručno unošenje podataka o tačkama, kao i polje za učitavanje datoteke koja će sadržati dotične podatke. Od dodatnih funkcionalnosti potrebno je dodati i mogućnost izmene već postojećih podataka.

Sve koordinate (x, y) biće zadate u vidu decimalnih brojeva koji predstavljaju koordinate u Dekartovom koordinatnom sistemu.

Obavezna je i provera integriteta i validnosti podataka (proizvoljna implementacija). Ona uključuje da li su unešeni podaci u decimalnom formatu, da li je datoteka pronađena/uspešno otvorena, da li fale određeni podaci, kao i da li je unešeni poligon jednostavan.

## Validacija podataka

Najrizičniji i najranjiviji deo svakog programa je zasigurno korisnički unos podataka. Ako se ne „steriliše“ svaki korisnički unos, može doći i do izvršavanja malicioznog koda, i raznih sigurnosnih propusta u najgorem slučaju, a do zaustavljanja rada programa u najboljem. Zato sam za potrebe ovog seminarskog rada isplanirao nov, poseban tip datoteke koji ćemo koristiti za čuvanje podataka o jednostavnim poligonima. Slede specifikacije .poligon datoteka:

.poligon datoteke će se sastojati iz dva dela:

Prvi deo biće tzv. **header** (srp. *zaglavlje*) koji će biti u sledećem formatu:

# - NazivDatoteke – BrojTemena – DaLiJeJednostavan

Gde NazivDatoteke predstavlja string s nazivom datoteke (bez .poligon ekstenzije), BrojTemena predstavlja prirodan broj temena poligona, a DaLiJeJednostavan nam govori da li je poligon jednostavan u formatu tačno/netačno.

Drugi deo će sadržati oznaku za teme ***t*** kao i realne brojeve koji će predstavljati njihov položaj u Dekartovom koordinatnom sistemu, i biće sledećeg formata:

t x1 y1

t x2 y2

...

t xn yn

Validacija tih podataka uključuje proveru tačnosti i konfornisanja s gorenavedenim standardom preko proizvoljnih algoritama i metoda ponuđenih u odabranom programskom jeziku i okruženju.

## Provera složenosti poligona

Kao što smo već gore ustanovili, većina algoritama i teorema koje važe za jednostavne poligone, ne funkcionišu niti se mogu primeniti na složenim poligonima. Iz tog razloga, doneo sam odluku da se fokusiram na jednostavne poligone u ovom seminarskom radu.

Iz toga proizilazi pitanje kako možemo znati da li radimo s jednostavnim ili složenim poligonom? Ispostavilo se da odgovor na ovo pitanje nije lako dati. Postoji nekoliko algoritama, od kojih svaki ima svoje „manjkavosti“. Algoritmi koji su brzi i efikasni, zauzimaju mnogo memorije i teško ih je implementirati, a za algoritme koji ne zauzimaju veliku količinu memorije, i lako je implementirati, potrebno je veliko vreme izvršavanja.

Za potrebe ovog seminarskog rada odlučio sam se za implementaciju “prostog” brute-force algoritma koji će proveravati svake nesusedne stranice za presecanja, i rezultat će vraćati u formatu tačno/netačno. Ovaj algoritam ima složenost O(n2), ali će nam ipak poslužiti budući da nećemo raditi s velikim brojem temena.

Pre nego što pređemo na analizu tog algoritma, moramo proveriti da li su svi skupovi tri uzastopnih tačaka **nekolinearni**, iz razloga što ako su tri uzastopne tačke kolinearne, one ne formiraju figuru, već pravu liniju.

Algoritam za proveru kolinearnosti tačaka je kranje jednostavan. Ako imamo tri uzastopne tačke T1(x1,y1), T2(x2,y2), T3(x3,y3), njihovu kolinearnost možemo proveriti preko sledeće formule:

Ako je rezultat determinante D jednak nuli, onda su te tri tačke kolinearne, i program mora obavestiti korisnika o tome. Proces ponoviti za sve uzastopne parove trotački u poligonu.

Ako algoritam ne pronađe nijedan kolinearan rezultat, onda se može nastaviti s izvršavanjem programa, i s proverom složenosti poligona. Provera složenosti se odvija po sledećim pravilima:

Zadate dve nesusedne ivice poligona AB(x1,y1; x2,y2) i CD(x3,y3; x4,y4) (gde su tačke temena obeležene s A, B, C, D) se međusobno **ne** seku ako je ispunjen sledeći uslov:

Gde funkcija *g(A, B, C)* predstavlja sledeću logičku proveru:

Proveru je potrebno ponoviti za svaki nesusedni par ivica u poligonu.

## Iscrtavanje poligona na ekranu

Kao što nam je već poznato, ove podatke pripremamo kako bismo ih na kraju iscrtali na mestu u programu koje je predviđeno za to. Ovde nećemo zalaziti u detalje implementacije programa za iscrtavanje, jer to sve zavisi od programskog jezika i okruženja koje koristimo. Međutim, možemo pričati o daljoj (rekao bih i konačnoj) pripremi podataka, nakon čega će oni biti spremni za crtanje.

Potrebno je diskutovati o temi iscrtavanja tačaka i linija na ekranu. Pošto znamo da je najmanja nedeljiva jedinica na samom ekranu piksel, uzećemo je za mernu jedinicu ovde. Pretpostavimo da imamo jedan deo grafičkog korisničkog interfejsa odvojen za iscrtavanje poligona, i da je taj deo kvadratnog oblika stranice *a.* U prethodnim poglavljima naveli smo kako su tačke koje unosimo realnog tipa, međutim, kako su pikseli najmanja **nedeljiva** jedinica, moraćemo nekako te realne brojeve da pretvorimo u cele brojeve. Takođe, moramo uskladiti naše koordinate s veličinom dela za iscrtavanje poligona (ne možemo iscrtati tačku s koordinatama (40000,40000) bez dalje obrade). Upravo iz tog razloga sam razvio posebnu proporciju:

Gde *x1* predstavlja našu novu koordinatu (koja je i dalje realnog tipa), *n* predstavlja stranicu *a* dela za iscrtavanje, *x* predstavlja početnu koordinatu, a max predstavlja **najveću apsolutnu vrednost** koordinate iz liste zadatih koordinata. Dakle, moramo izvršiti i proveru nad skupom dostupnih tačaka kako bismo pronašli najveću koordinatu. U obzir dolaze i *x* i *y* koordinate. Obezbediti da *max* nikada ne bude jednako nuli.

Nakon ponavljanja sledeće proporcije i dobijanja novog skupa tačaka, potrebno je izvršiti još jednu izmenu. Tačke se moraju zaokružiti i mora im se dodati određena celobrojna vrednost koja zavisi od stranice *a*, tj. od širine/dužine dela za iscrtavanje poligona, u zavisnosti od toga u kom kvadrantu se tačka nalazi. Zato sam razvio još jednu matematičku formulu za ovaj proces:

Ako se tačka nalazi u koordinatnom početku:

Ako je *x* pozitivno:

Ako je *x* negativno:

Ako je *y* pozitivno:

Ako je *y* negativno:

Dakle, ove matematičke formule su **u potpunosti nezavisne** od bilo kog programskog jezika, i mogu se implementirati u okruženju po želji.

## Matematički proračuni – obim, površina

Kod matematičkih proračuna kao što su obim i površina, već postoje jednostavni i efikasni algoritmi koji se mogu s lakoćom prevesti iz matematike. Napomena: Već smo više puta utvrdili da se ovi algoritmi ne mogu primeniti na složenim poligonima, te ako se utvrdi da je unesen složen poligon, program mora onemogućiti izvršavanje ovih algoritama. Za izračunavanje obima poligona koristićemo sledeći algoritam:

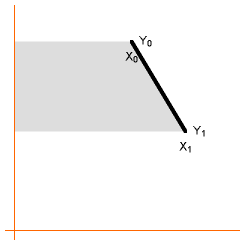
Za jednostavan poligon P koji se sastoji iz stranica A1, A2, ..., An obim se može izračunati na sledeći način:

Sada se postavlja pitanje, kako izračunati dužine stranica? Sledi formula za izračunavanje dužine stranice *L* u dvodimenzionalnom Dekartovom koordinatnom sistemu za stranicu zadatu tačkama T1(x1,y1) i T2(x2,y2):

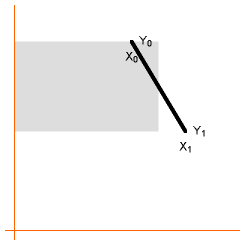
Površinu *P* jednostavnog poligona možemo izračunati tako što ćemo sabrati sve površine ograničene datim ivicama i Y koordinatnom osom, preko sledeće formule:

Rezultate je potrebno prikazati na prostoru posebno odvojenom za tu namenu.

Kako funkcioniše ovaj algoritam za računanje površine jednostavnog poligona? Na sledeći način; Ako uzmemo za posmatranje određenu stranicu poligona ograničenu tačkama *{ (x0, y0) (x1, y1) }* i uzmemo površinu s leve strane, sve do Y koordinatne ose (osenčeno sivom bojom na slici 2), ta površina će biti jednaka pravougaoniku na slici 3.

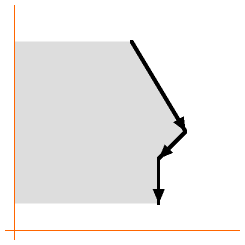


Slika : Stranica poligona



Slika : Pravougaonik površine

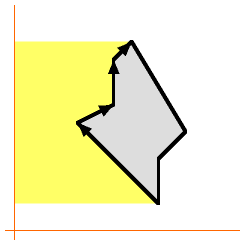
Ovu površinu je trivijalno izračunati preko formule . Ovu delimičnu površinu dodajemo ukupnoj površini poligona, kao što možete videti na slici 4.



Slika : Sabiranje površina

Nakon što pređemo na drugu stranu poligona, oduzimamo sve žute površine (prikazane na slici 5), iz razloga što kada je ivica usmerena na gore, y0-y1 je negativan broj. Površina koja nije oduzeta na kraju je površina poligona

Nije obavezno početi na vrhu poligona. Možete početi bilo gde, i brojevi će svakako dati isti rezultat. Takođe, nije bitno da li su neki od poligonovih uglova ili stranica u drugim kvadrantima (negativno x ili y), rezultat će biti uvek isti.



Slika : Površina jednostavnog poligona

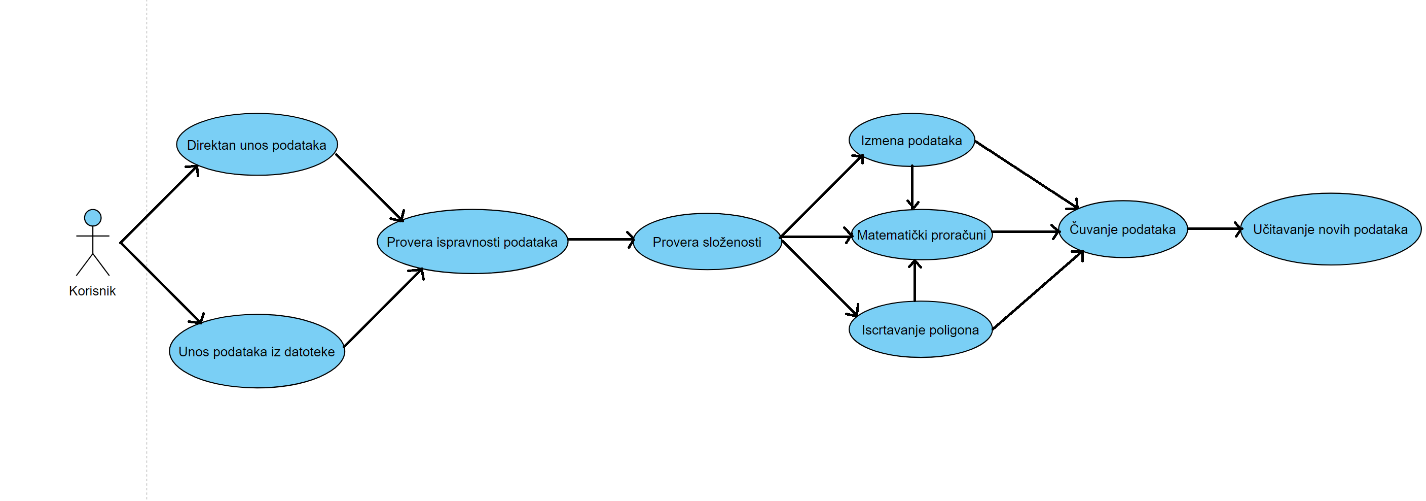
## Izmena i čuvanje podataka

Program **mora** obezbediti sledeće funkcionalnosti:

* Indeksirani, tabelarni i pre svega editabilni prikaz podataka o temenima poligona.
* Sposobnost čuvanja podataka u vidu .poligon fajla prema standardu navedenom u poglavlju “Unos podataka” i preko nativnog interfejsa operativnog sistema.

Na ovom poglavlju nećemo se više zadržavati, iz razloga što ono zavisi od same implementacije, kao i od programskog jezika i okruženja, a na početku smo rekli da se time nećemo baviti u ovom seminarskom radu.

# Use Case dijagram



Slika : Use Case dijagram

**Slučaj korišćenja #1: Direktan unos podataka**

Akter: Korisnik

Opis: Akter unosi podatke o poligonu u polju specijalno predviđenom za tu namenu.

**Slučaj korišćenja #2: Unos podataka iz datoteke**

Akter: Korisnik

Opis: Akter bira i učitava datoteku iz koje će program pročitati podatke o poligonu.

**Slučaj korišćenja #3: Provera ispravnosti podataka**

Akter: Korisnik

Opis: Akteru se prikazuje statusna poruka nakon rada funkcija za proveru ispravnosti podataka.

**Slučaj korišćenja #4: Provera složenosti**

Akter: Korisnik

Opis: Akteru se prikazuje rezultat funkcija za proveru složenosti poligona. Ukoliko je poligon složen, aktivnost #6 postaje nedostupna.

**Slučaj korišćenja #5: Izmena podataka**

Akter: Korisnik

Opis: Akter menja podatke o poligonu po želji.

**Slučaj korišćenja #6: Matematički proračuni**

Akter: Korisnik

Opis: Akter pokreće funkcije za matematičke proračune nad poligonom, kao što su površina i obim.

**Slučaj korišćenja #7: Iscrtavanje poligona**

Akter: Korisnik

Opis: Akter pokreće funkciju za iscrtavanje poligona na ekranu.

**Slučaj korišćenja #8: Čuvanje podataka**

Akter: Korisnik

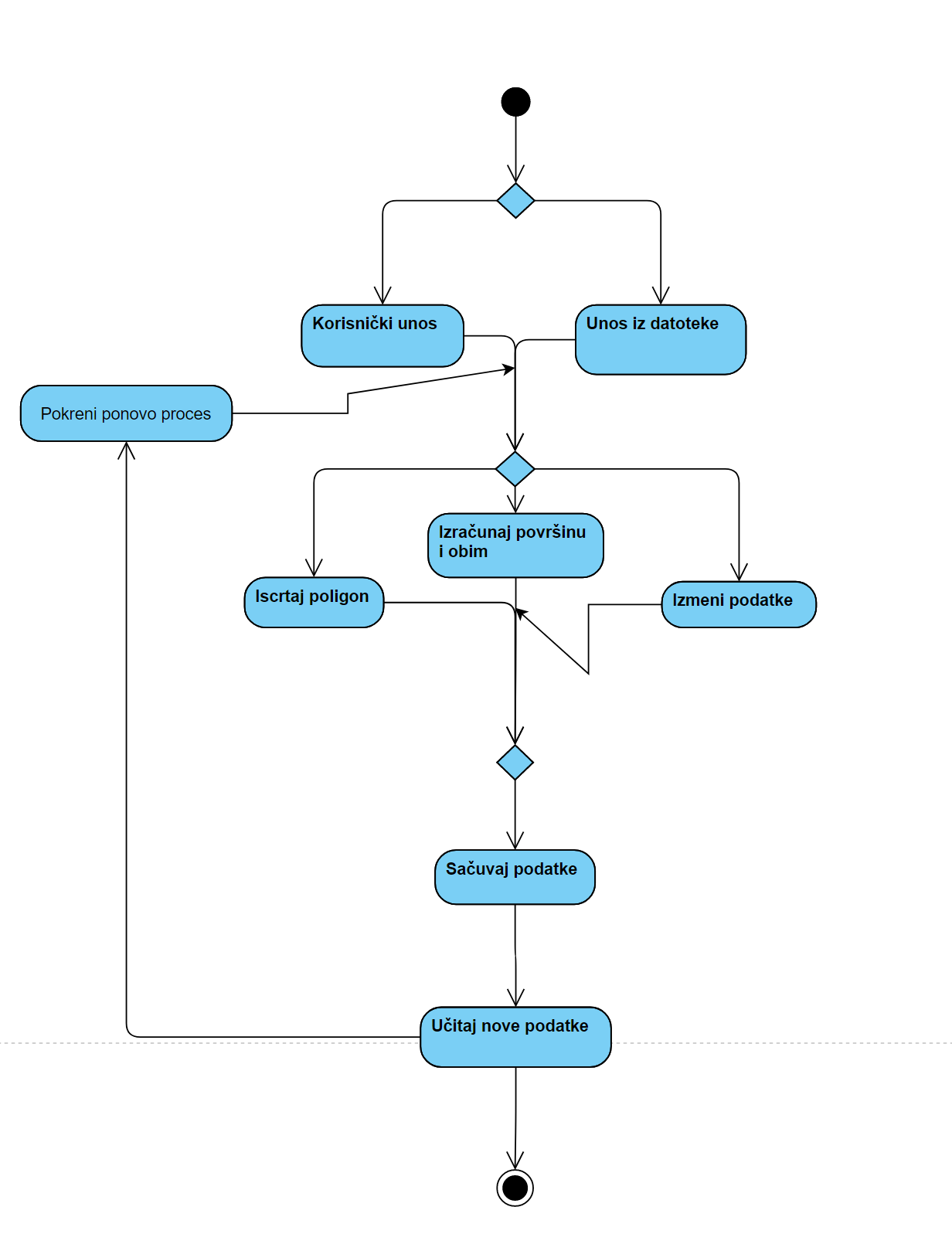
Opis: Akter čuva podatke o poligonu u datoteku s ekstenzijom .poligon

**Slučaj korišćenja #9: Učitavanje novih podataka**

Akter: Korisnik

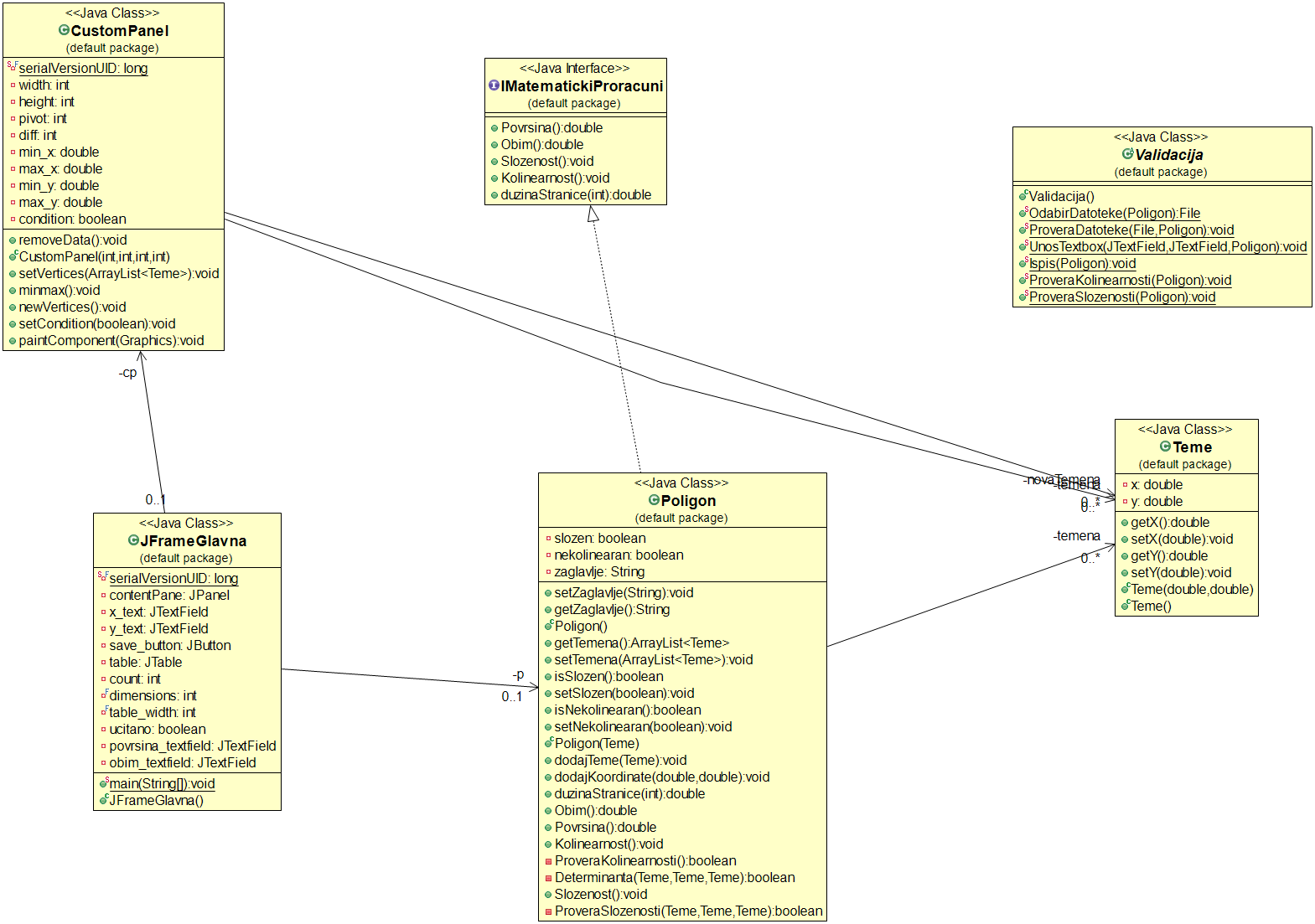
Opis: Akter po želji učitava nove podatke i započinje ceo proces iznova.

# Dijagram aktivnosti



Slika 7: Dijagram aktivnosti

# Dijagram klasa



# Reference i tabela figura

## Reference/Literatura

1. [*https://cp-algorithms.com/geometry/area-of-simple-polygon.html*](https://cp-algorithms.com/geometry/area-of-simple-polygon.html)
2. [*https://www.geeksforgeeks.org/program-check-three-points-collinear/*](https://www.geeksforgeeks.org/program-check-three-points-collinear/)
3. [*https://stackoverflow.com/questions/3838329/how-can-i-check-if-two-segments-intersect*](https://stackoverflow.com/questions/3838329/how-can-i-check-if-two-segments-intersect)
4. [*https://www.mathopenref.com/coordpolygonarea2.html*](https://www.mathopenref.com/coordpolygonarea2.html)
5. [*https://www.geeksforgeeks.org/what-is-the-importance-of-mathematics-in-computer-science/*](https://www.geeksforgeeks.org/what-is-the-importance-of-mathematics-in-computer-science/)
6. [*https://en.wikipedia.org/wiki/Simple\_polygon*](https://en.wikipedia.org/wiki/Simple_polygon)
7. [*https://mathworld.wolfram.com/SimplePolygon.html*](https://mathworld.wolfram.com/SimplePolygon.html)
8. [*http://www.faqs.org/faqs/graphics/algorithms-faq/*](http://www.faqs.org/faqs/graphics/algorithms-faq/)
9. *В. Лазаревић, А. Шебековић, Математика 2 - решени примери, Технички факултет, Чачак, 2007, ИСБН 978-86-7776-041-0.*

## Tabela figura

[*Slika 1:Razne vrste poligona 4*](#_Toc62116239)

[*Slika 2: Stranica poligona 9*](#_Toc62116240)

[*Slika 3: Pravougaonik površine 9*](#_Toc62116241)

[*Slika 4: Sabiranje površina 9*](#_Toc62116242)

[*Slika 5: Površina jednostavnog poligona 9*](#_Toc62116243)

[*Slika 6: Use Case dijagram 11*](#_Toc62116244)

[*Slika 7: Dijagram aktivnosti 12*](#_Toc62116245)