Univerzitet u Sarajevu
Elektrotehnički fakultet Sarajevo
Odsjek za računarstvo i informatiku

Task 2: Inspekcija koda

e-Demokratija

ČLANOVI TIMA:

1. Dalila Kršlak – 18906 (vođa tima)
2. Nejra Adilović – 19061
3. Vildana Tabaković – 18968
4. Berina Zejnilović – 18805

ODGOVORNI NASTAVNIK:

R. prof. dr. Dženana Đonko dipl.el.ing

ODGOVORNI ASISTENT:

Mr. dipl. ing. Neira Novalić

Sadržaj

1. Organizacija inspekcije koda

Moderator inspekcije je Dalila Kršlak. Kod je dodijeljen na sljedeći način:

1. Nejra Adilović:
    - Klasa Program.cs
    - Klasa Kandidat.cs
2. Berina Zejnilović:
    - Klasa CSVMaker.cs
    - Klasa Glasac.cs
3. Vildana Tabaković:
    - Klasa Supervizor.cs
4. Dalila Kršlak:
    - Klasa Glas.cs

1.1. Checklists

ČLAN TIMA: Nejra Adilović

Structure

✓ Does the code completely and correctly implement the design?
❑ Does the code conform to any pertinent coding standards?
✓ Is the code well-structured, consistent in style, and consistently formatted?
❑ Are there any uncalled or unneeded procedures or any unreachable code?
❑ Are there any leftover stubs or test routines in the code?
❑ Can any code be replaced by calls to external reusable components or library functions?
❑ Are there any blocks of repeated code that could be condensed into a single procedure?
✓ Is storage use efficient?
✓ Are symbolics used rather than "magic number" constants or string constants?
❑ Are any modules excessively complex and should be restructured or split into multiple routines?

Documentation

❑ Is the code clearly and adequately documented with an easy-to-maintain commenting style?
❑ Are all comments consistent with the code?

Variables

✓ Are all variables properly defined with meaningful, consistent, and clear names?
✓ Do all assigned variables have proper type consistency or casting?
❑ Are there any redundant or unused variables?

Arithmetic Operations

✓ Does the code avoid comparing floating-point numbers for equality?
✓ Does the code systematically prevent rounding errors?
✓ Does the code avoid additions and subtractions on numbers with greatly different magnitudes?
❑ Are divisors tested for zero or noise?

Loops and Branches

✓ Are all loops, branches, and logic constructs complete, correct, and properly nested?
✓ Are the most common cases tested first in IF- -ELSEIF chains?
✓ Are all cases covered in an IF- -ELSEIF or CASE block, including ELSE or DEFAULT clauses?
❑ Does every case statement have a default?
✓ Are loop termination conditions obvious and invariably achievable?
✓ Are indexes or subscripts properly initialized, just prior to the loop?
❑ Can any statements that are enclosed within loops be placed outside the loops?
✓ Does the code in the loop avoid manipulating the index variable or using it upon exit from the loop?

Defensive Programming

✓ Are indexes, pointers, and subscripts tested against array, record, or file bounds?
✓ Are imported data and input arguments tested for validity and completeness?
✓ Are all output variables assigned?
✓ Are the correct data operated on in each statement?
✓ Is every memory allocation deallocated?
❑ Are timeouts or error traps used for external device accesses?
❑ Are files checked for existence before attempting to access them?
✓ Are all files and devices are left in the correct state upon program termination?

ČLAN TIMA: Berina Zejnilović

Structure

✓ Does the code completely and correctly implement the design?
❑ Does the code conform to any pertinent coding standards?
✓ Is the code well-structured, consistent in style, and consistently formatted?
❑ Are there any uncalled or unneeded procedures or any unreachable code?
❑ Are there any leftover stubs or test routines in the code?
❑ Can any code be replaced by calls to external reusable components or library functions?
❑ Are there any blocks of repeated code that could be condensed into a single procedure?
✓ Is storage use efficient?
✓ Are symbolics used rather than “magic number” constants or string constants?
❑ Are any modules excessively complex and should be restructured or split into multiple routines?

Documentation

✓ Is the code clearly and adequately documented with an easy-to-maintain commenting style?
❑ Are all comments consistent with the code?

Variables

✓ Are all variables properly defined with meaningful, consistent, and clear names?
✓ Do all assigned variables have proper type consistency or casting?
❑ Are there any redundant or unused variables?

Arithmetic Operations

✓ Does the code avoid comparing floating-point numbers for equality?
✓ Does the code systematically prevent rounding errors?
✓ Does the code avoid additions and subtractions on numbers with greatly different magnitudes?
❑ Are divisors tested for zero or noise?

Loops and Branches

✓ Are all loops, branches, and logic constructs complete, correct, and properly nested?
✓ Are the most common cases tested first in IF- -ELSEIF chains?
✓ Are all cases covered in an IF- -ELSEIF or CASE block, including ELSE or DEFAULT clauses?
❑ Does every case statement have a default?
✓ Are loop termination conditions obvious and invariably achievable?
✓ Are indexes or subscripts properly initialized, just prior to the loop?
❑ Can any statements that are enclosed within loops be placed outside the loops?
✓ Does the code in the loop avoid manipulating the index variable or using it upon exit from the loop?

Defensive Programming

✓ Are indexes, pointers, and subscripts tested against array, record, or file bounds?
✓ Are imported data and input arguments tested for validity and completeness?
✓ Are all output variables assigned?
✓ Are the correct data operated on in each statement?
✓ Is every memory allocation deallocated?
❑ Are timeouts or error traps used for external device accesses?
✓ Are files checked for existence before attempting to access them?
✓ Are all files and devices are left in the correct state upon program termination?

ČLAN TIMA: Vildana Tabaković

Structure

✓ Does the code completely and correctly implement the design?
❑ Does the code conform to any pertinent coding standards?
✓ Is the code well-structured, consistent in style, and consistently formatted?
❑ Are there any uncalled or unneeded procedures or any unreachable code?
❑ Are there any leftover stubs or test routines in the code?
❑ Can any code be replaced by calls to external reusable components or library functions?
❑ Are there any blocks of repeated code that could be condensed into a single procedure?
✓ Is storage use efficient?
❑ Are symbolics used rather than “magic number” constants or string constants?
❑ Are any modules excessively complex and should be restructured or split into multiple routines?

Documentation

❑ Is the code clearly and adequately documented with an easy-to-maintain commenting style?
❑ Are all comments consistent with the code?

Variables

❑ Are all variables properly defined with meaningful, consistent, and clear names?
✓ Do all assigned variables have proper type consistency or casting?
❑ Are there any redundant or unused variables?

Arithmetic Operations

❑ Does the code avoid comparing floating-point numbers for equality?
✓ Does the code systematically prevent rounding errors?
❑ Does the code avoid additions and subtractions on numbers with greatly different magnitudes?
❑ Are divisors tested for zero or noise?

Loops and Branches

✓ Are all loops, branches, and logic constructs complete, correct, and properly nested?
❑ Are the most common cases tested first in IF- -ELSEIF chains?
✓ Are all cases covered in an IF- -ELSEIF or CASE block, including ELSE or DEFAULT clauses?
❑ Does every case statement have a default?
❑ Are loop termination conditions obvious and invariably achievable?
✓ Are indexes or subscripts properly initialized, just prior to the loop?
❑ Can any statements that are enclosed within loops be placed outside the loops?
❑ Does the code in the loop avoid manipulating the index variable or using it upon exit from the loop?

Defensive Programming

❑ Are indexes, pointers, and subscripts tested against array, record, or file bounds?
❑ Are imported data and input arguments tested for validity and completeness?
✓ Are all output variables assigned?
❑ Are the correct data operated on in each statement?
✓ Is every memory allocation deallocated?
✓ Are timeouts or error traps used for external device accesses?
❑ Are files checked for existence before attempting to access them?
✓ Are all files and devices are left in the correct state upon program termination?

ČLAN TIMA: Dalila Kršlak

Structure

✓ Does the code completely and correctly implement the design?
❑ Does the code conform to any pertinent coding standards?
✓ Is the code well-structured, consistent in style, and consistently formatted?
❑ Are there any uncalled or unneeded procedures or any unreachable code?
❑ Are there any leftover stubs or test routines in the code?
❑ Can any code be replaced by calls to external reusable components or library functions?

❑ Are there any blocks of repeated code that could be condensed into a single procedure?
✓ Is storage use efficient?
❑ Are symbolics used rather than “magic number” constants or string constants?
❑ Are any modules excessively complex and should be restructured or split into multiple routines?

Documentation

❑ Is the code clearly and adequately documented with an easy-to-maintain commenting style?
❑ Are all comments consistent with the code?

Variables

✓ Are all variables properly defined with meaningful, consistent, and clear names?
✓ Do all assigned variables have proper type consistency or casting?
❑ Are there any redundant or unused variables?

Arithmetic Operations

✓ Does the code avoid comparing floating-point numbers for equality?
✓ Does the code systematically prevent rounding errors?
✓ Does the code avoid additions and subtractions on numbers with greatly different magnitudes?
❑ Are divisors tested for zero or noise?

Loops and Branches

✓ Are all loops, branches, and logic constructs complete, correct, and properly nested?
✓ Are the most common cases tested first in IF- -ELSEIF chains?
✓ Are all cases covered in an IF- -ELSEIF or CASE block, including ELSE or DEFAULT clauses?
❑ Does every case statement have a default?
✓ Are loop termination conditions obvious and invariably achievable?
✓ Are indexes or subscripts properly initialized, just prior to the loop?
❑ Can any statements that are enclosed within loops be placed outside the loops?
✓ Does the code in the loop avoid manipulating the index variable or using it upon exit from the loop?

Defensive Programming

❑ Are indexes, pointers, and subscripts tested against array, record, or file bounds?
✓ Are imported data and input arguments tested for validity and completeness?
✓ Are all output variables assigned?
✓ Are the correct data operated on in each statement?
✓ Is every memory allocation deallocated?
❑ Are timeouts or error traps used for external device accesses?
❑ Are files checked for existence before attempting to access them?
✓ Are all files and devices are left in the correct state upon program termination?

1.2. Izvještaji članova

Izvještaj člana:	Nejra Adilović
Datum sesije:	5.12.2023.
Ime projekta:	e-Demokratija
Inspekcijski tim:	Dalila Kršlak, Vildana Tabaković, Berina Zejnilović, Nejra Adilović
Verzija:	1

1. Lista grešaka

#	Tip greške	Priroda errora	Opis greške	Mjesto greške	Ozbiljnost greške
1	Logička	Wrong	Pogrešan uslov za kraj petlje	Program.cs – 87. linija	3
2	Logička	Wrong	Zamijenjeni uslovi	Program.cs – 136. i 159. linija	3
3	Sintaksna	Extra	Ponovljeni ispis	Program.cs – 191. linija	2
4	Logička	Wrong	Zamijenjen uslov	Kandidat.cs – 101. linija	3
5	Logička	Missing	Nedostatak uslova	Kandidat.cs – 69. linija	2
6	Sintaksna	Wrong	Pogrešan uslov	Kandidat.cs – 62. linija	3

2. Ispravke grešaka

a	Grešku će ispraviti:	Berina Zejnilović, Vildana Tabaković, Dalila Kršlak
b	Re-inspekcija preporučena:	Da

Izvještaj člana:	Berina Zejnilović
Datum sesije:	5.12.2023.
Ime projekta:	e-Demokratija
Inspekcijski tim:	Dalila Kršlak, Vildana Tabaković, Berina Zejnilović, Nejra Adilović
Verzija:	1

1. Lista grešaka

#	Tip greške	Priroda errora	Opis greške	Mjesto greške	Ozbiljnost greške
1	Logička	Wrong	Pogrešan uslov	CSVMaker.cs – 26. linija	4
2	Logička	Wrong	Pogrešna inicijalizacija	CSVMaker.cs – 123. linija	3
3	Sintaksna	Missing	Nedostatak poziva metode	Glasac.cs – 27. linija	4
4	Logička	Wrong	Pogrešna inicijalizacija stringa	Glasac.cs – 78. linija	2
5	Sintaksna	Missing	Nedostatak uslova	Glasac.cs – 129. linija	2
6	Logička	Wrong	Pogrešan uslov	Glasac.cs – 170. linija	2

2. Ispravke grešaka

a	Grešku će ispraviti:	Nejra Adilović, Vildana Tabaković
b	Re-inspekcija preporučena:	Da

Izvještaj člana:	Vildana Tabaković
Datum sesije:	5.12.2023.
Ime projekta:	e-Demokratija
Inspekcijski tim:	Dalila Kršlak, Vildana Tabaković, Berina Zejnilović, Nejra Adilović
Verzija:	1

1. Lista grešaka

#	Tip greške	Priroda errora	Opis greške	Mjesto greške	Ozbiljnost greške
1	Logička	Wrong	Pogrešna inicijalizacija	Supervizor.cs – 14.linija	3
2	Logička	Wrong	Pogrešna deklaracija	Supervizor.cs – 11. linija	3
3	Sintaksna	Wrong	Neispravan uslov	Supervizor.cs – 77. linija	3
4	Logička	Wrong	Pogrešan pristup varijabli	Supervizor.cs – 185. linija	5
5	Logička	Wrong	Pogrešne granice u petlji	Supervizor.cs – 206. linija	2
6	Logička	Wrong	Neispravno grananje	Supervizor.cs – 77. linija	3
7	Logička	Wrong	Pogrešno proslijeđen parametar	Supervizor.cs – 41. linija	3
8	Logička	Wrong	Neispravno zatvoren blok	Supervizor.cs – 58. linija	2

2. Ispravke grešaka

a	Grešku će ispraviti:	Nejra Adilović, Berina Zejnilović
b	Re-inspekcija preporučena:	Da

Izvještaj člana:	Dalila Kršlak
Datum sesije:	5.12.2023.
Ime projekta:	e-Demokratija
Inspekcijski tim:	Dalila Kršlak, Vildana Tabaković, Berina Zejnilović, Nejra Adilović
Verzija:	1

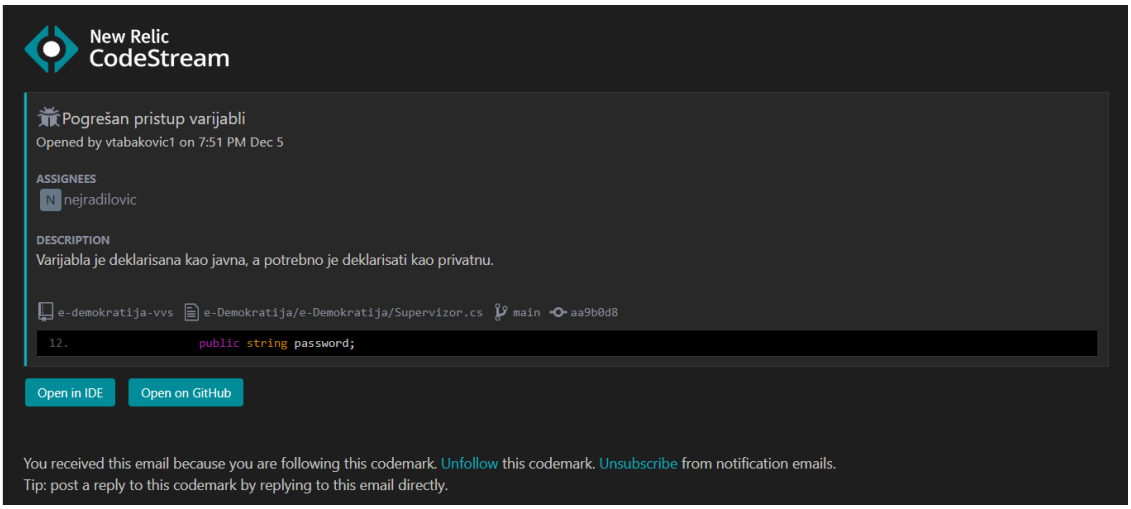
1. Lista grešaka

#	Tip greške	Priroda errora	Opis greške	Mjesto greške	Ozbiljnost greške
1	Logička	Wrong	Pogrešna povratna vrijednost	Glas.cs – 50. linija	3
2	Logička	Wrong	Nedostatak uslova	Glas.cs – 58. linija	2
3	Sintaksna	Wrong	Neispravan uslov	Glas.cs – 101. linija	1
4	Logička	Wrong	Pogrešna inicijalizacija brojača	Glas.cs – 108. linija	3
5	Logička	Wrong	Pogrešan if uslov	Glas.cs – 174. linija	3

2. Ispravke grešaka

a	Grešku će ispraviti:	Dalila Kršlak
b	Re-inspekcija preporučena:	Da

1.3. Korištenje CodeStream-a



2. Izvještaj moderatora

Datum sesije: 5.12.2023.

Naziv projekta: e-Demokratija

Verzija: 1

Inspekcioni tim: Dalila Kršlak, Nejra Adilović, Berina Zejnilović, Vildana Tabaković

1 Investirani resursi (sati rada)						
#	Ime člana	Sastanak	Priprema	Inspekcijska sesija	Ukupno (sati)	Komentar
1	Moderator Dalila	1	1	2.5	4.5	
2	Nejra	1	2	2	5	
3	Berina	1	2	2	5	
4	Vildana	1	2	2	5	
	Ukupno	**4**	**7**	**8.5**	**19.5**	

2 Greške u kodu							
Ozbiljnost	Priroda grešaka (W M E)*			Broj grešaka	Faktor ozbiljnosti	Broj grešaka (standardizovano)	Komentar
5-kritično	1			1	16	16	
4	1	1		2	8	16	
3	13			13	4	52	
2	5	2	1	8	2	16	
1-minorno	1			1	1	1	
Total	**21**	**3**	**1**	**25**	//	**101**	

3 Metrike defekata		
(1) Prosječno defekata po stranici 25/6 = 4.167		
(2) Prosječno defekata po stranici (standardizovano) 101/6 = 16.834		
(3) Efikasnost detektovanja defekata (sati po defektu) 19.5/25 = 0.78 h/def		
(4) Efikasnost detektovanja defekata standardizovano (sati po defektu) 19.5/101 = 0.193 h/def		
Pripremila: Dalila Kršlak	* W – pogrešno, M -nedostaje, E - viška	Datum: 5.12.2023.

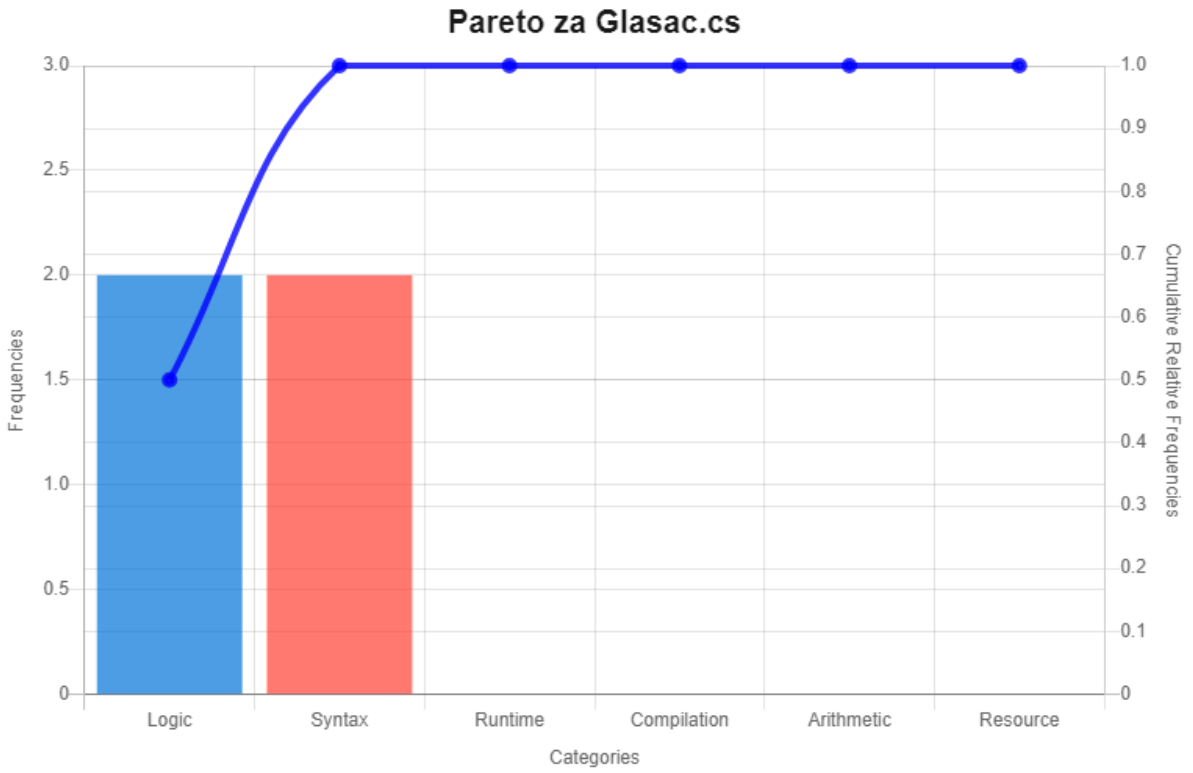
Opis odvijanja sesije:

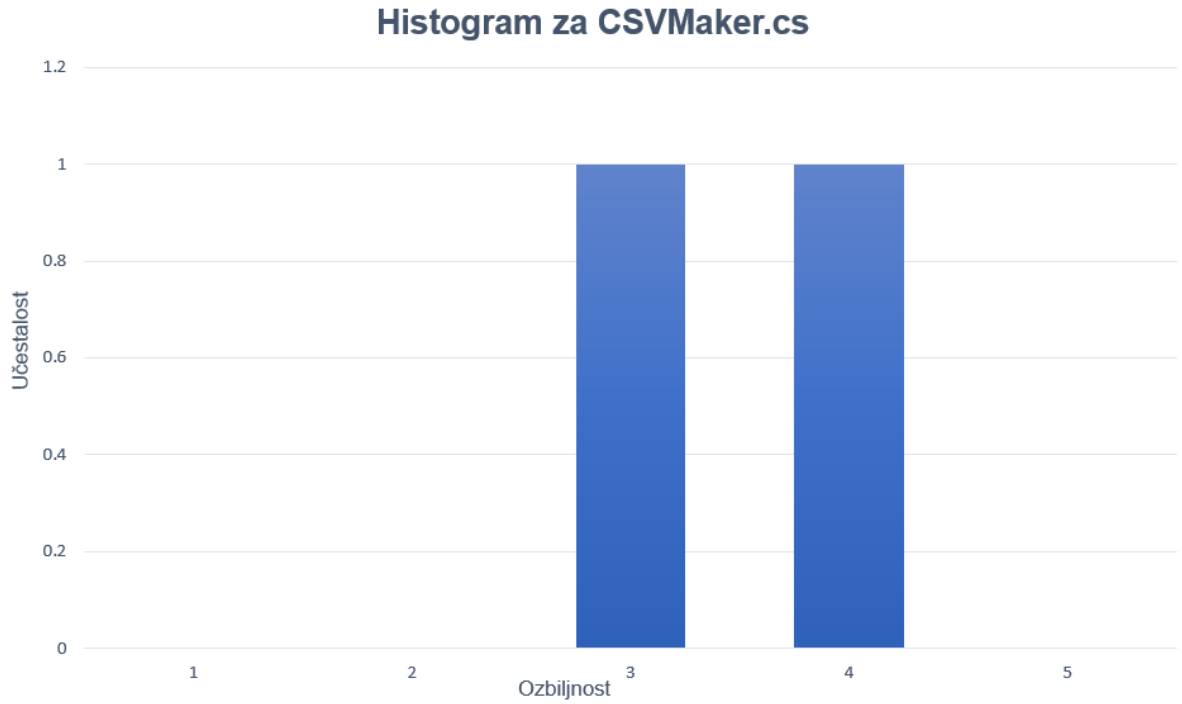
Prije same sesije održan je pripremni sastanak koji je trajao sat vremena. Tokom tog sastanka, članovi tima su zajednički odredili moderatora i dodijelili ostale uloge. Moderator je raspodijelio check liste i tabele ozbiljnosti među članovima tima. Svaki član dobio je određeni dio koda za pregled, a zatim su krenule pripreme.

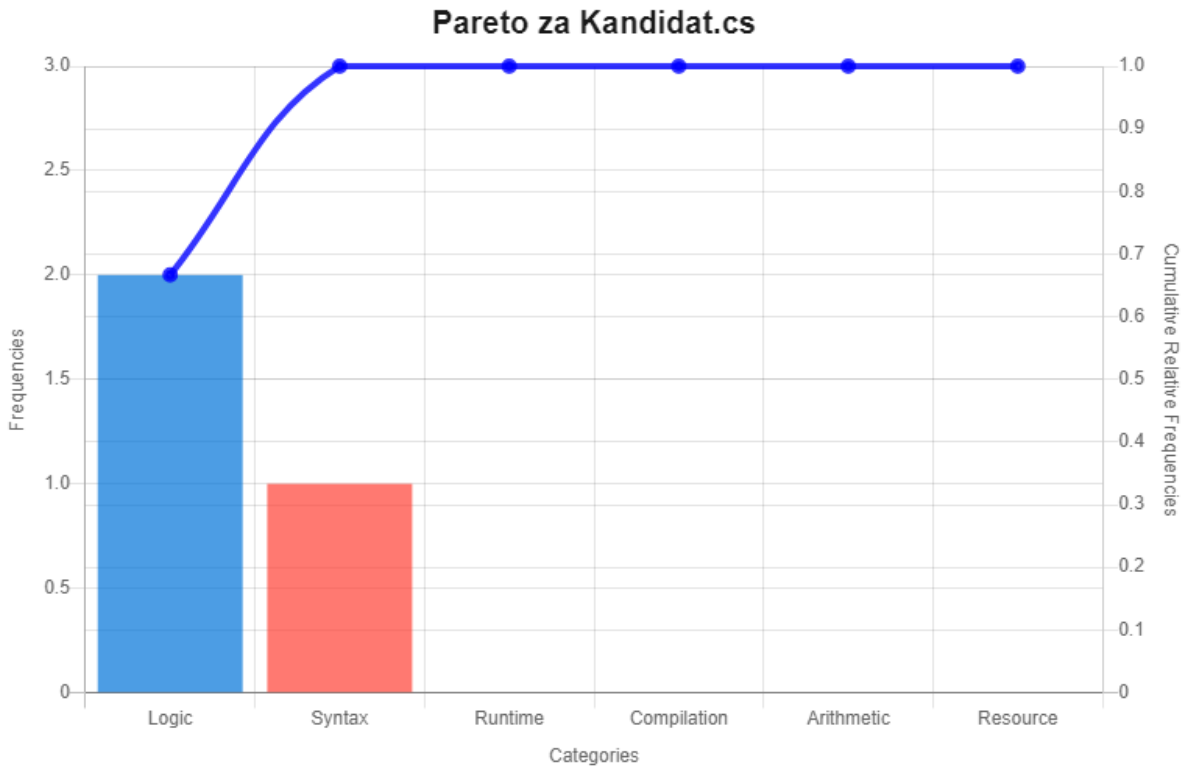
Recezenti i moderator su zajednički popunili dokumente koji su bili dostupni prije same sesije, a zatim su analizirali i pripremili kod za inspekciju. Na narednom sastanku, koji je trajao otprilike 2 sata, recezenti su zabilježili identificirane greške, a moderator/zapisničar je te greške sistematizirao kako bi se izbjeglo zaboravljanje.

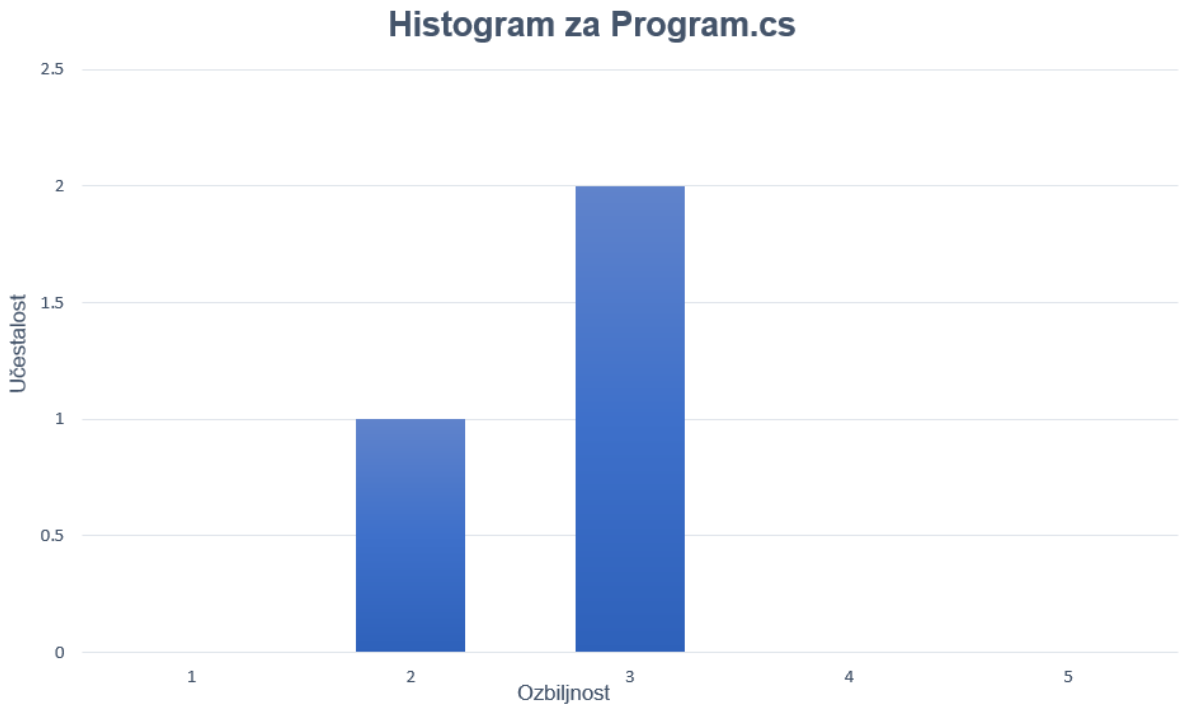
Nakon završetka sesije, sastavljen je sumarni izvještaj koji je služio kao temelj za daljnje korake. Autor programa je potom odredio ko će izvršiti potrebne ispravke, te je izvršen ponovni pregled. Ovaj iterativni proces kontrola i povratnih informacija osigurao je kvalitetu rada i ispravnost koda prije konačne evaluacije.

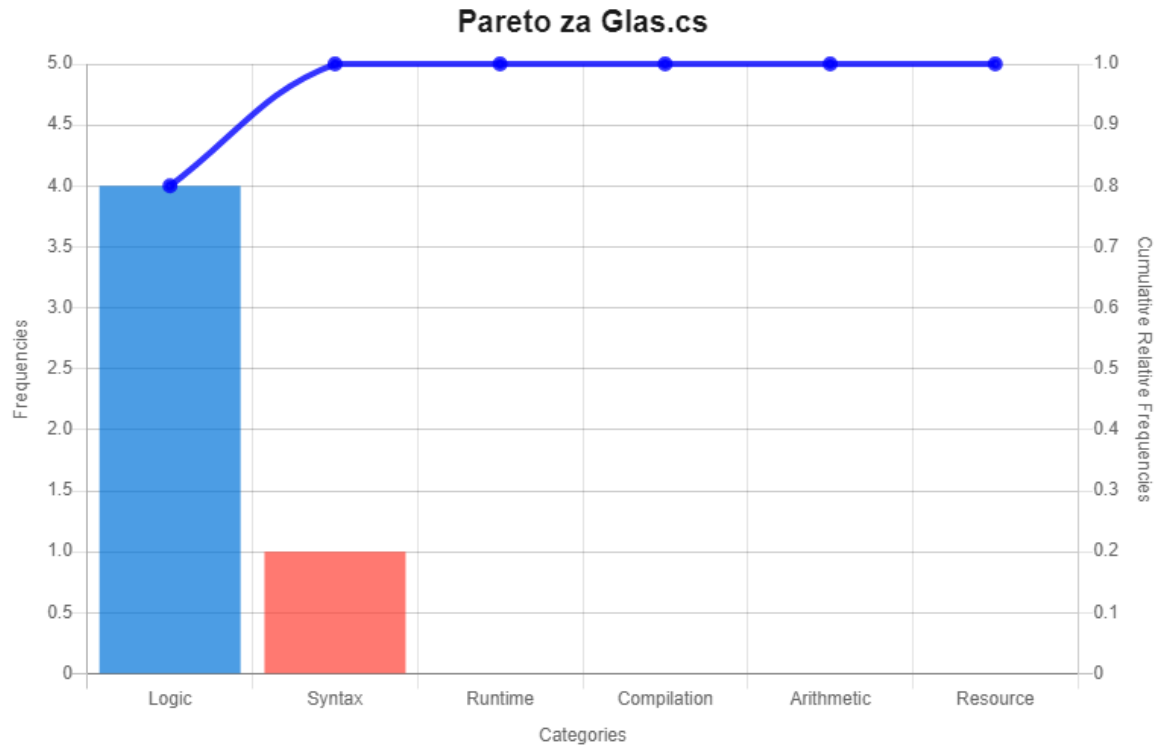
3. Statistički alati

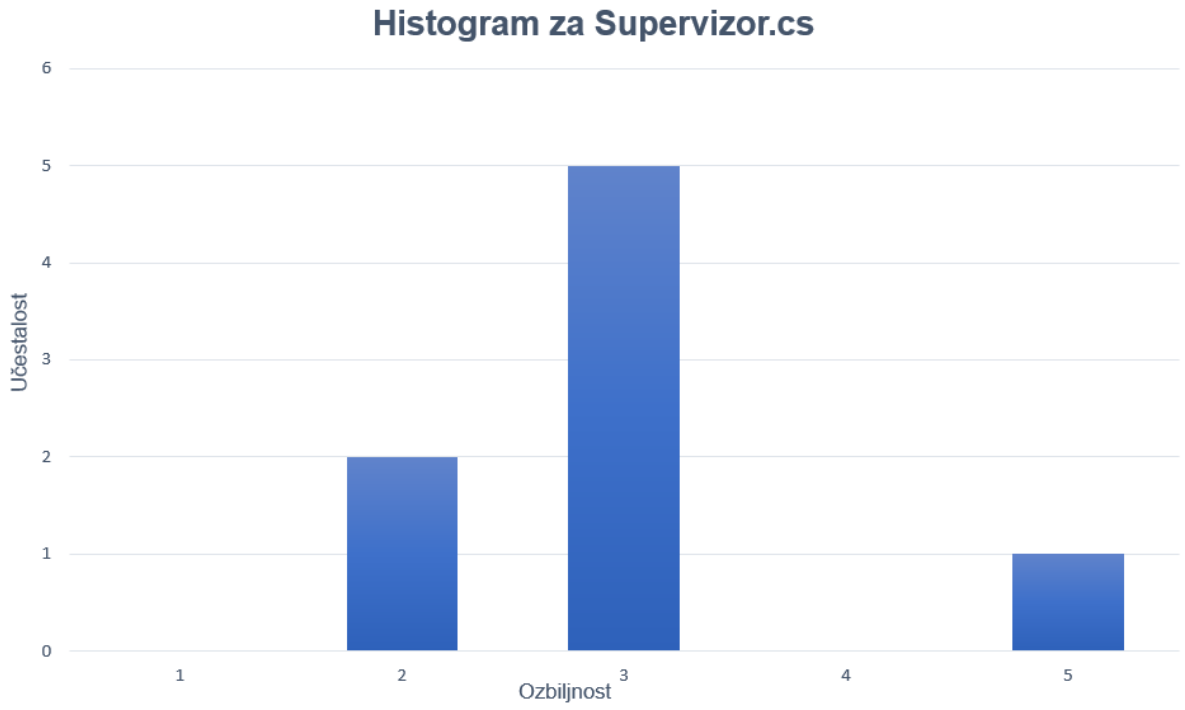





Pareto za Kandidat.cs
Frequencies
Cumulative Relative Frequencies
Logic
Syntax
Runtime
Compilation
Arithmetic
Resource
Categories


Histogram za Program.cs
Učestalost
Ozbiljnost


Pareto za Glas.cs
Frequencies
Cumulative Relative Frequencies
Logic
Syntax
Runtime
Compilation
Arithmetic
Resource
Categories


Histogram za Supervizor.cs
Učestalost
Ozbiljnost

4. Spisak korektivnih akcija

Za svakog člana je navedeno koji dio koda je ispravljao.

Nejra Adilović

Stavke vezane za klasu Supervizor:

- Pogrešan pristup varijabli password, umjesto da bude deklarisana kao javna, potrebno je deklarisati kao privatnu
- Pogrešno deklarisan atribut datumRodjenjaKandidata, tj. nije deklarisan u skladu sa parametrima klase DateTime umjesto DateTime(dan, mjesec, godina) potrebno je ispraviti na DateTime(godina, mjesec, dan)
- Potrebno je dodati naredbu za prekid petlje pri iteriranju kroz Stranke tj. dodati break; naredbu da bi se prekinulo traganje za strankom jer je već pronađena
- Neispravan uslov, potrebno je promijeniti da uslov bude negiran na 119. liniji
- Pogrešno postavljene granice for-petlje, petlja treba da kreće od 0 a ne od 1 kako bi se pristupilo svakoj stranci

Stavke vezane za klasu CSVMarker:

- Pogrešan uslov, potrebno je promijeniti da uslov bude negiran tj. UkolikoFile.Exists(putanjaGlasovi) ne postoji da se ista kreira

Stavke vezane za klasu Glasac:

- Pogrešna inicijalizacija stringa, potrebno je postaviti ga na prazan string
- Nedostatak uslova na 161. liniji je potrebno također provjeriti da li je brojCrtica veći od 1

Vildana Tabaković

Stavke vezane za klasu Program:

- Popravljen uslov u while petlji, potrebno prepraviti da uslov bude negiran
- Izbačena linija koda koja je ponavljala ispis koji je već bio odrađen prije petlje

Stavke vezane za klasu CSVMarker:

- Prepravljena inicijalizacija

Stavke vezane za klasu Glasac:

- Popravljen uslov u if-u, potrebno da bude datum bude veći od DateTime.now
- Potrebno pozvati metodu FormirajKodGlasaca() prije završetka konstruktora

Berina Zejnilović

Stavke vezane za klasu Program:

- Pravilno postavljeni uslovi za dodavanje i brisanje kandidata

Stavke vezane za klasu Supervizor:

- Popravljena dodjela passworda
- Popravljen odnos između uslova za pozicije gradonačenika, načelnika i vijećnika
- Popravljeno brisanje traženog kandidata, a ne njegovog prethodnika u metodi IzbrisiKandidata()

Dalila Kršlak

Stavke vezane za klasu Kandidat:

- Pravilno postavljen uslov za provjeru pozicije prilikom ispisa vijećnika
- Dodan uslov vijećnika u provjeri ispravnosti pozicije
- Ispravljen uslov u metodi za validaciju opisa

Stavke vezane za klasu Glas:

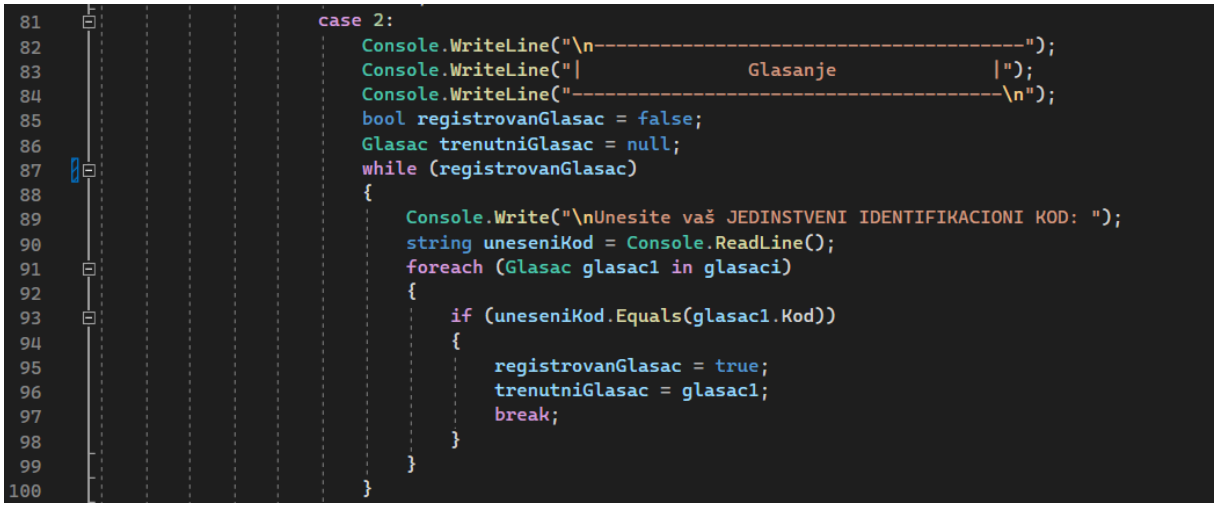
- Ispravljena povratna vrijednost metode DaLiJeGlasanjePocelo()
- Dodan uslov vijećnika u provjeri ispravnosti glasanja
- Ispravljen uslov u metodi za ispis stanja gradonačelnika
- brojacZaNacelnike postavljen na 0
- Ispravljen uslov za provjeru da li je glasanje počelo

4.1. Korektivne akcije prije i poslije

4.1.1. Klasa Program.cs

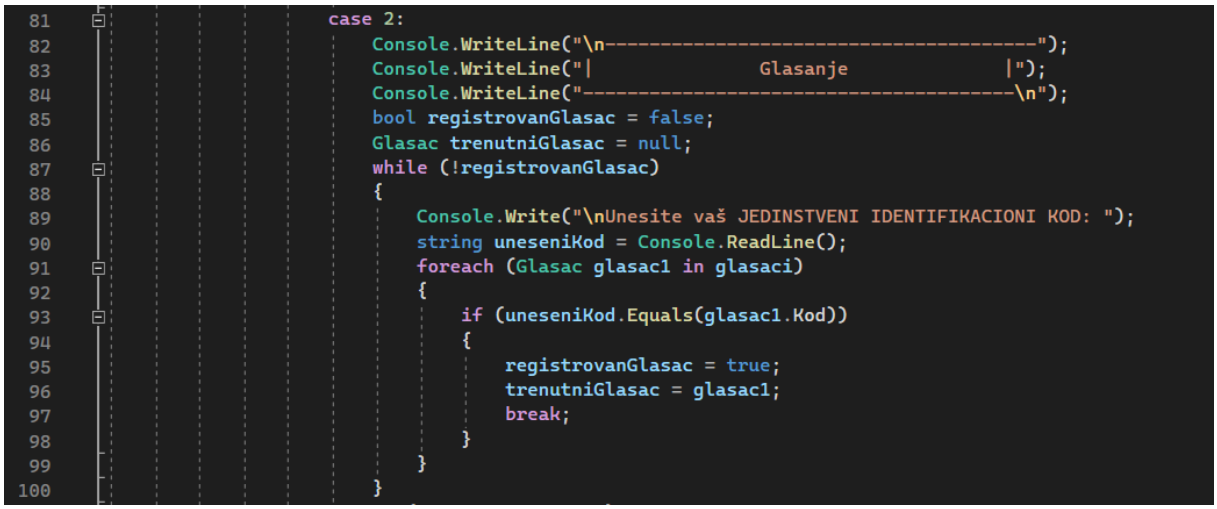
GREŠKA 1:

Kod prije ispravke:



```
81          case 2:
82              Console.WriteLine("\n---------------------------------------");
83              Console.WriteLine("|              Glasanje              |");
84              Console.WriteLine("---------------------------------------\n");
85              bool registrovanGlasac = false;
86              Glasac trenutniGlasac = null;
87              while (registrovanGlasac)
88              {
89                  Console.Write("\nUnesite vaš JEDINSTVENI IDENTIFIKACIONI KOD: ");
90                  string uneseniKod = Console.ReadLine();
91                  foreach (Glasac glasac1 in glasaci)
92                  {
93                      if (uneseniKod.Equals(glasac1.Kod))
94                      {
95                          registrovanGlasac = true;
96                          trenutniGlasac = glasac1;
97                          break;
98                      }
99                  }
100             }
```

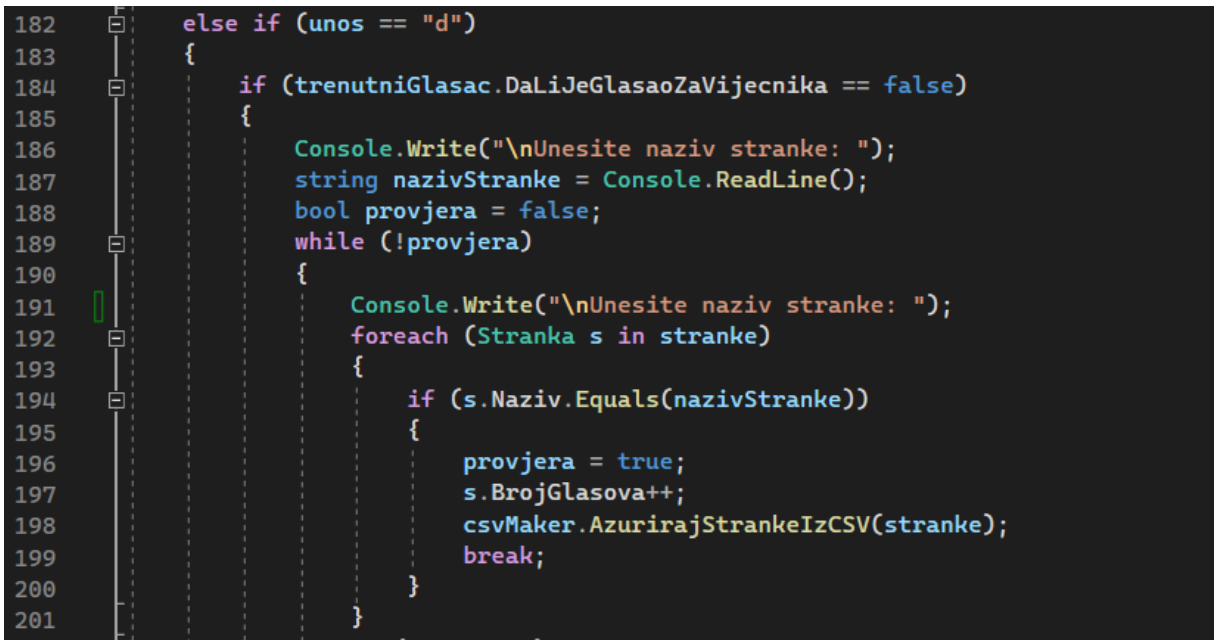
Kod poslije ispravke:



```
81          case 2:
82              Console.WriteLine("\n---------------------------------------");
83              Console.WriteLine("|              Glasanje              |");
84              Console.WriteLine("---------------------------------------\n");
85              bool registrovanGlasac = false;
86              Glasac trenutniGlasac = null;
87              while (!registrovanGlasac)
88              {
89                  Console.Write("\nUnesite vaš JEDINSTVENI IDENTIFIKACIONI KOD: ");
90                  string uneseniKod = Console.ReadLine();
91                  foreach (Glasac glasac1 in glasaci)
92                  {
93                      if (uneseniKod.Equals(glasac1.Kod))
94                      {
95                          registrovanGlasac = true;
96                          trenutniGlasac = glasac1;
97                          break;
98                      }
99                  }
100             }
```

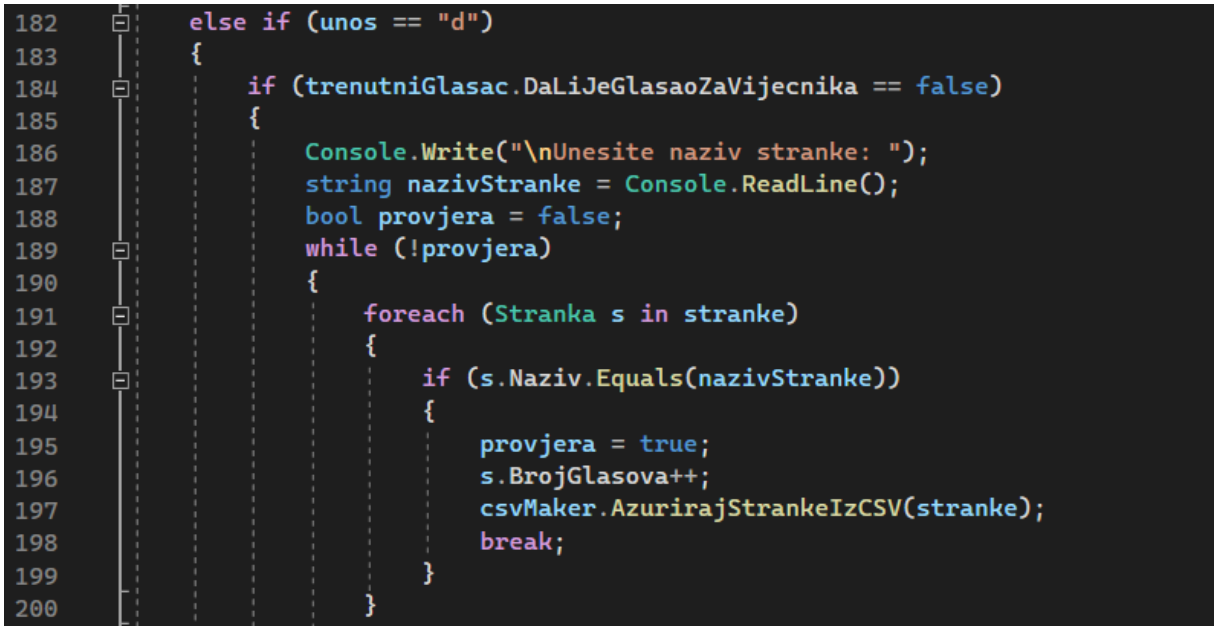
GREŠKA 2:

Kod prije ispravke:



```
182     else if (unos == "d")
183     {
184         if (trenutniGlasac.DaLiJeGlasaoZaVijecnika == false)
185         {
186             Console.Write("\nUnesite naziv stranke: ");
187             string nazivStranke = Console.ReadLine();
188             bool provjera = false;
189             while (!provjera)
190             {
191                 Console.Write("\nUnesite naziv stranke: ");
192                 foreach (Stranka s in stranke)
193                 {
194                     if (s.Naziv.Equals(nazivStranke))
195                     {
196                         provjera = true;
197                         s.BrojGlasova++;
198                         csvMaker.AzurirajStrankeIzCSV(stranke);
199                         break;
200                     }
201                 }
```

Kod poslije ispravke:



```
182     else if (unos == "d")
183     {
184         if (trenutniGlasac.DaLiJeGlasaoZaVijecnika == false)
185         {
186             Console.Write("\nUnesite naziv stranke: ");
187             string nazivStranke = Console.ReadLine();
188             bool provjera = false;
189             while (!provjera)
190             {
191                 foreach (Stranka s in stranke)
192                 {
193                     if (s.Naziv.Equals(nazivStranke))
194                     {
195                         provjera = true;
196                         s.BrojGlasova++;
197                         csvMaker.AzurirajStrankeIzCSV(stranke);
198                         break;
199                     }
200                 }
```

GREŠKA 3:

Kod prije ispravke:

```
136     else if (unos == "c")
137     {
138         if (trenutniGlasac.DaLiJeGlasaoZaNacelnika == false)
139         {
140             Console.WriteLine("\nKandidati za načelnika su: \n");
141             kand.IspisiKandidateZaNacelnika(kandidati);
142             Console.Write("\nVaš glas je za načelnika pod rednim brojem: ");
143             int redniBrojNacelnika = Int32.Parse(Console.ReadLine());
144             foreach (Kandidat k in kandidati)
145             {
146                 if (k.RedniBroj == redniBrojNacelnika)
147                 {
148                     Glas glas = new Glas(trenutniGlasac, k);
149                     csvMaker.DodajGlas(glas);
150                     csvMaker.AzurirajKandidateIzCSV(kandidati);
151                     csvMaker.AzurirajGlasaceIzCSV(glasaci);
152                     break;
153                 }
154             }
155         }
156         else
157             Console.WriteLine("\nVec ste glasali za nacelnika!");
158     }
159     else if (unos == "b")
160     {
```

Kod poslije ispravke:

```
136     else if (unos == "b")
137     {
138         if (trenutniGlasac.DaLiJeGlasaoZaNacelnika == false)
139         {
140             Console.WriteLine("\nKandidati za načelnika su: \n");
141             kand.IspisiKandidateZaNacelnika(kandidati);
142             Console.Write("\nVaš glas je za načelnika pod rednim brojem: ");
143             int redniBrojNacelnika = Int32.Parse(Console.ReadLine());
144             foreach (Kandidat k in kandidati)
145             {
146                 if (k.RedniBroj == redniBrojNacelnika)
147                 {
148                     Glas glas = new Glas(trenutniGlasac, k);
149                     csvMaker.DodajGlas(glas);
150                     csvMaker.AzurirajKandidateIzCSV(kandidati);
151                     csvMaker.AzurirajGlasaceIzCSV(glasaci);
152                     break;
153                 }
154             }
155         }
156         else
157             Console.WriteLine("\nVec ste glasali za nacelnika!");
158     }
159     else if (unos == "c")
160     {
```

4.1.2. Klasa Glasac.cs

GREŠKA 1:

Kod prije ispravke:

```
22      public Glasac(String ime, String prezime, DateTime datumRodjenja)
23      {
24          this.ime = ime;
25          this.prezime = ime;
26          this.datumRodjenja = datumRodjenja;
27      }
```

Kod poslije ispravke:

```
22      public Glasac(String ime, String prezime, DateTime datumRodjenja)
23      {
24          this.ime = ime;
25          this.prezime = ime;
26          this.datumRodjenja = datumRodjenja;
27          FormirajKodGlasaca();
28      }
```

GREŠKA 2:

Kod prije ispravke:

```
78              string dan = ".";
79              if (datumRodjenja.Day > 10)
80              {
81                  dan = "0" + datumRodjenja.Day.ToString();
82              }
```

Kod poslije ispravke:

```
78              string dan = "";
79              if (datumRodjenja.Day > 10)
80              {
81                  dan = "0" + datumRodjenja.Day.ToString();
82              }
```

GREŠKA 3:

Kod prije ispravke:

```
129             if (ime.Length < 3 || ime.Length > 20 || !samoCrtice)
130                 throw new ArgumentOutOfRangeException("Upisano ime nije validno!");
131         }
```

Kod poslije ispravke:

```
129             if (ime.Length < 3 || ime.Length > 20 || !samoCrtice || brojCrtica > 1)
130                 throw new ArgumentOutOfRangeException("Upisano ime nije validno!");
131         }
```

GREŠKA 4:

Kod prije ispravke:

```
170             if (DateTime.Now.Date > datum.AddYears(18))
171             {
172                 throw new ArgumentOutOfRangeException("Glasač nije punoljetan!");
173             }
```

Kod poslije ispravke:

```
170             if (DateTime.Now.Date < datum.AddYears(18))
171             {
172                 throw new ArgumentOutOfRangeException("Glasač nije punoljetan!");
173             }
```

4.1.3. Klasa CSVMaker.cs

GREŠKA 1:

Kod prije ispravke:

```
26          if (File.Exists(putanjaGlasaci))
27          {
28              File.Create(putanjaGlasaci).Close();
29          }
```

Kod poslije ispravke:

```
26          if (!File.Exists(putanjaGlasaci))
27          {
28              File.Create(putanjaGlasaci).Close();
29          }
```

GREŠKA 2:

Kod prije ispravke:

```
120     public void AzurirajGlasaceIzCSV(List<Glasac> glasaci)
121     {
122         var csvConfig = new CsvConfiguration(CultureInfo.InvariantCulture);
123         csvConfig.HasHeaderRecord = false;
124
125         using (var writer = new StreamWriter(Path.Combine(Path.Combine(Directory.GetPare
126         using (var csv = new CsvWriter(writer, csvConfig))
127         {
128             csv.WriteRecords(glasaci);
129         }
130     }
```

Kod poslije ispravke:

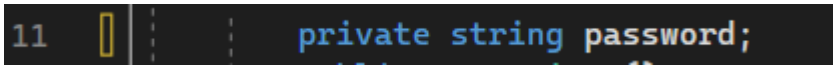
```
120     public void AzurirajGlasaceIzCSV(List<Glasac> glasaci)
121     {
122         var csvConfig = new CsvConfiguration(CultureInfo.InvariantCulture);
123         csvConfig.HasHeaderRecord = true;
124
125         using (var writer = new StreamWriter(Path.Combine(Path.Combine(Directory.GetPare
126         using (var csv = new CsvWriter(writer, csvConfig))
127         {
128             csv.WriteRecords(glasaci);
129         }
130     }
```

4.1.4. Klasa Supervizor.cs

GREŠKA 1:

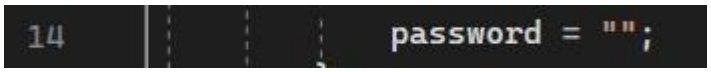
Kod prije ispravke:

Kod poslije ispravke:



GREŠKA 2:

Kod prije ispravke:

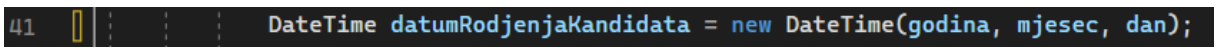


Kod poslije ispravke:

GREŠKA 3:

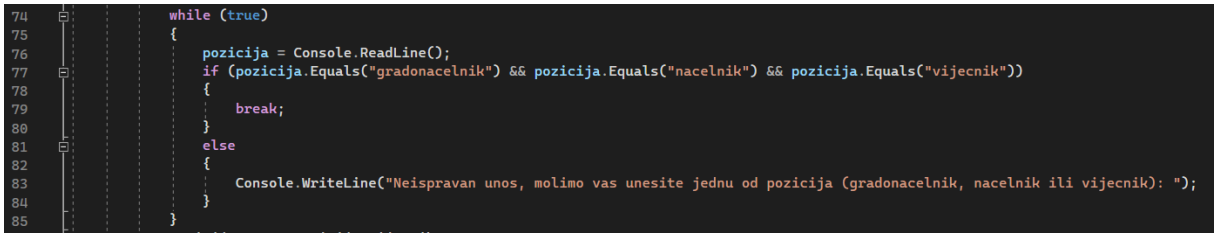
Kod prije ispravke:

Kod poslije ispravke:



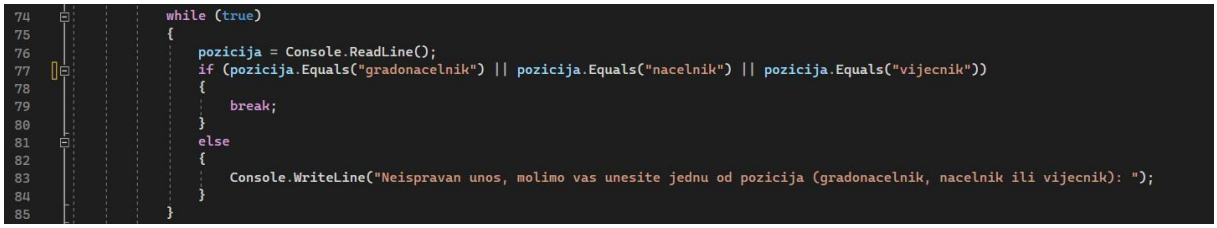
GREŠKA 4:

Kod prije ispravke:



```
74          while (true)
75          {
76              pozicija = Console.ReadLine();
77              if (pozicija.Equals("gradonacelnik") && pozicija.Equals("nacelnik") && pozicija.Equals("vijecnik"))
78              {
79                  break;
80              }
81              else
82              {
83                  Console.WriteLine("Neispravan unos, molimo vas unesite jednu od pozicija (gradonacelnik, nacelnik ili vijecnik): ");
84              }
85          }
```

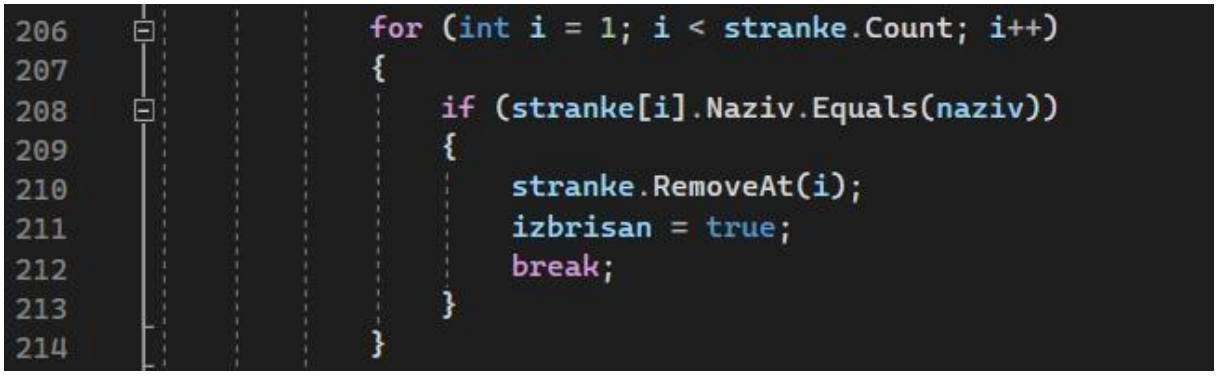
Kod poslije ispravke:



```
74          while (true)
75          {
76              pozicija = Console.ReadLine();
77              if (pozicija.Equals("gradonacelnik") || pozicija.Equals("nacelnik") || pozicija.Equals("vijecnik"))
78              {
79                  break;
80              }
81              else
82              {
83                  Console.WriteLine("Neispravan unos, molimo vas unesite jednu od pozicija (gradonacelnik, nacelnik ili vijecnik): ");
84              }
85          }
```

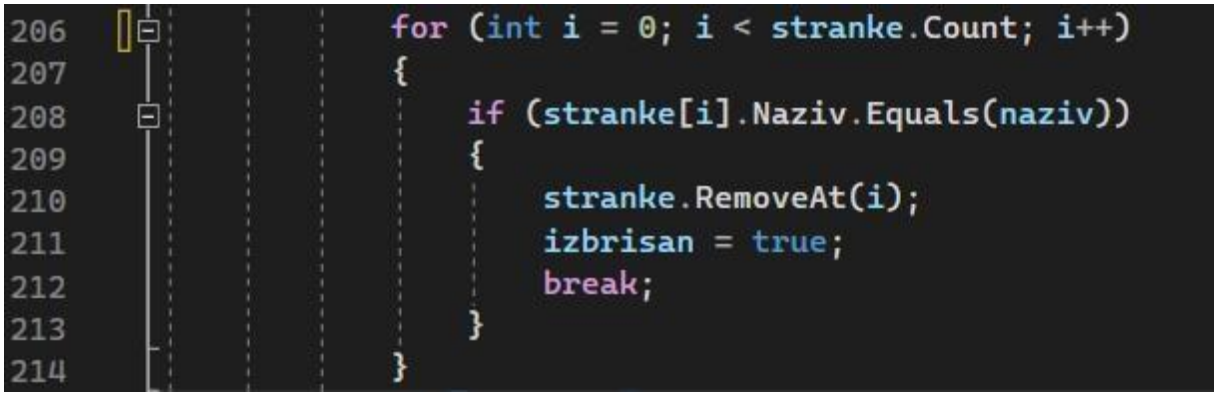
GREŠKA 5:

Kod prije ispravke:



```
206         for (int i = 1; i < stranke.Count; i++)
207         {
208             if (stranke[i].Naziv.Equals(naziv))
209             {
210                 stranke.RemoveAt(i);
211                 izbrisan = true;
212                 break;
213             }
214         }
```

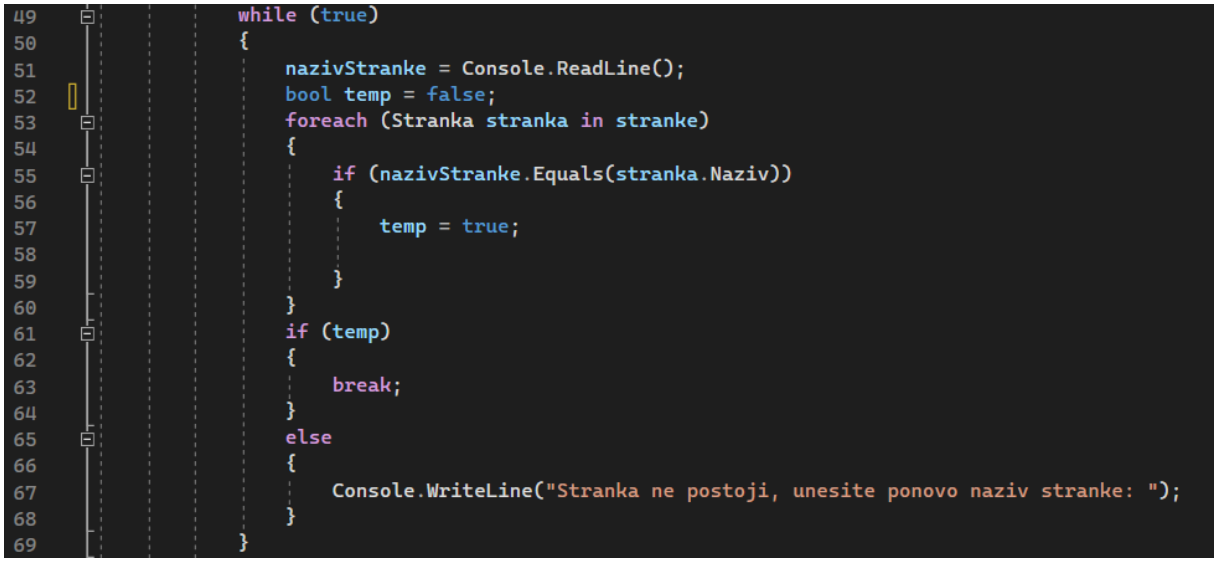
Kod poslije ispravke:



```
206         for (int i = 0; i < stranke.Count; i++)
207         {
208             if (stranke[i].Naziv.Equals(naziv))
209             {
210                 stranke.RemoveAt(i);
211                 izbrisan = true;
212                 break;
213             }
214         }
```

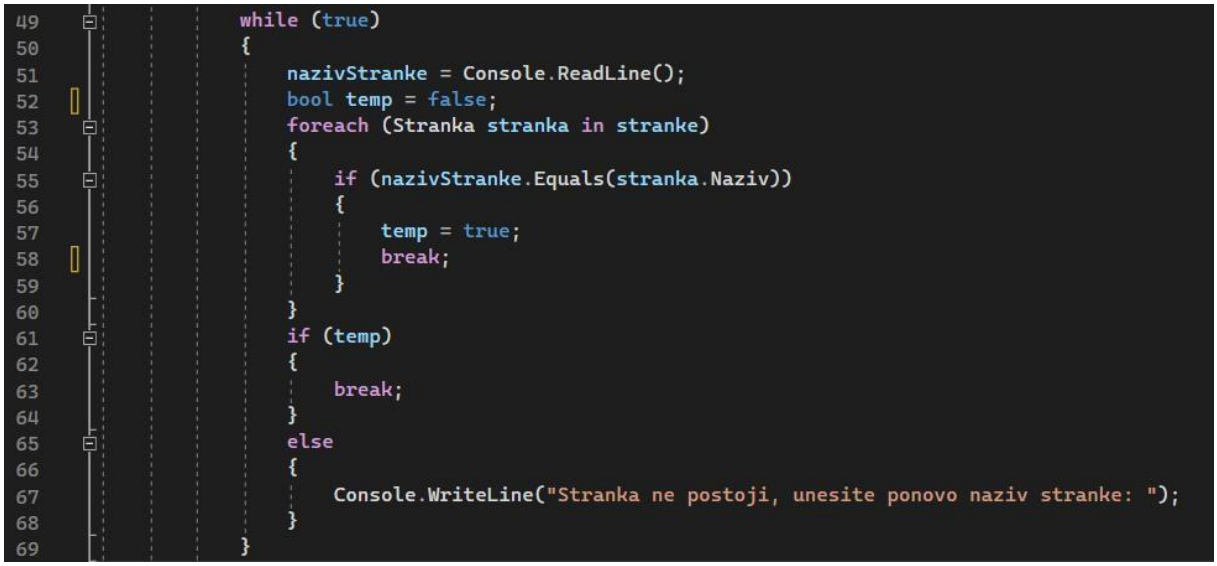
GREŠKA 6:

Kod prije ispravke:



```
49          while (true)
50          {
51              nazivStranke = Console.ReadLine();
52              bool temp = false;
53              foreach (Stranka stranka in stranke)
54              {
55                  if (nazivStranke.Equals(stranka.Naziv))
56                  {
57                      temp = true;
58
59                  }
60              }
61              if (temp)
62              {
63                  break;
64              }
65              else
66              {
67                  Console.WriteLine("Stranka ne postoji, unesite ponovo naziv stranke: ");
68              }
69          }
```

Kod poslije ispravke:



```
49          while (true)
50          {
51              nazivStranke = Console.ReadLine();
52              bool temp = false;
53              foreach (Stranka stranka in stranke)
54              {
55                  if (nazivStranke.Equals(stranka.Naziv))
56                  {
57                      temp = true;
58                      break;
59                  }
60              }
61              if (temp)
62              {
63                  break;
64              }
65              else
66              {
67                  Console.WriteLine("Stranka ne postoji, unesite ponovo naziv stranke: ");
68              }
69          }
```

GREŠKA 7:

Kod prije ispravke:

```
118             if (s.Naziv.Equals(naziv))
119             {
120                 pronasao = true;
121                 break;
122             }
```

Kod poslije ispravke:

```
118             if (!s.Naziv.Equals(naziv))
119             {
120                 pronasao = true;
121                 break;
122             }
```

GREŠKA 6:

Kod prije ispravke:

```
185                 kandidati.RemoveAt(i--);
186                 izbrisan = true;
187                 break;
```

Kod poslije ispravke:

```
185                 kandidati.RemoveAt(i);
186                 izbrisan = true;
187                 break;
```

4.1.5. Klasa Kandidat.cs

GREŠKA 1:

Kod prije ispravke:

```
58      public void DaLiJeOpisIspravan (string opis)
59      {
60          if (string.IsNullOrWhiteSpace(opis))
61              throw new ArgumentException("Opis kandidata ne može biti prazan!");
62          if(opis.Split(' ').Length > 3)
63          {
64              throw new ArgumentException("Opis kandidata treba sadrzavati minimalno 3 rijeci!");
65          }
66      }
```

Kod poslije ispravke:

```
58      public void DaLiJeOpisIspravan (string opis)
59      {
60          if (string.IsNullOrWhiteSpace(opis))
61              throw new ArgumentException("Opis kandidata ne može biti prazan!");
62          if(opis.Split(' ').Length < 3)
63          {
64              throw new ArgumentException("Opis kandidata treba sadrzavati minimalno 3 rijeci!");
65          }
66      }
```

GREŠKA 2:

Kod prije ispravke:

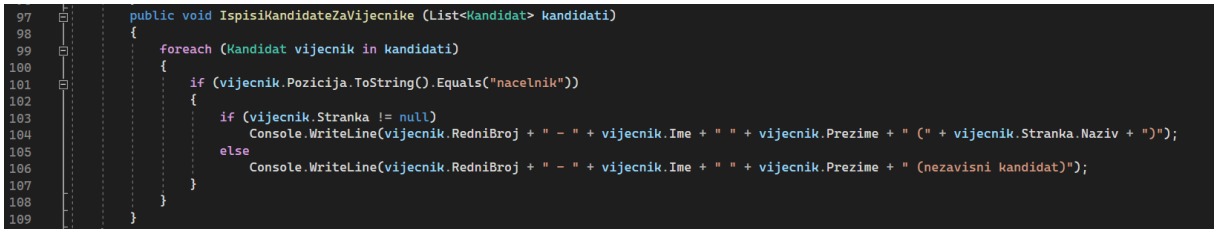
```
67      public bool DaLiJePozicijaIspravna (string pozicija)
68      {
69          return pozicija.Equals("gradonacelnik") || pozicija.Equals("nacelnik");
70      }
```

Kod poslije ispravke:

```
67      public bool DaLiJePozicijaIspravna (string pozicija)
68      {
69          return pozicija.Equals("gradonacelnik") || pozicija.Equals("nacelnik") || pozicija.Equals("vijecnik");
70      }
```

GREŠKA 3:

Kod prije ispravke:



```
97      public void IspisiKandidateZaVijecnike (List<Kandidat> kandidati)
98      {
99          foreach (Kandidat vijecnik in kandidati)
100         {
101             if (vijecnik.Pozicija.ToString().Equals("nacelnik"))
102             {
103                 if (vijecnik.Stranka != null)
104                     Console.WriteLine(vijecnik.RedniBroj + " - " + vijecnik.Ime + " " + vijecnik.Prezime + " (" + vijecnik.Stranka.Naziv + ")");
105                 else
106                     Console.WriteLine(vijecnik.RedniBroj + " - " + vijecnik.Ime + " " + vijecnik.Prezime + " (nezavisni kandidat)");
107             }
108         }
109     }
```

Kod poslije ispravke:

```
97      public void IspisiKandidateZaVijecnike (List<Kandidat> kandidati)
98      {
99          foreach (Kandidat vijecnik in kandidati)
100         {
101             if (vijecnik.Pozicija.ToString().Equals("vijecnik"))
102             {
103                 if (vijecnik.Stranka != null)
104                     Console.WriteLine(vijecnik.RedniBroj + " - " + vijecnik.Ime + " " + vijecnik.Prezime + " (" + vijecnik.Stranka.Naziv + ")");
105                 else
106                     Console.WriteLine(vijecnik.RedniBroj + " - " + vijecnik.Ime + " " + vijecnik.Prezime + " (nezavisni kandidat)");
107             }
108         }
109     }
```

4.1.6. Klasa Glas.cs

GREŠKA 1:

Kod prije ispravke:

```
46      private bool DaLiJeGlasanjePocelo(List<Glas> glasovi)
47      {
48          if (glasovi.Count == 0)
49              return false;
50          return false;
51      }
```

Kod poslije ispravke:

```
46      private bool DaLiJeGlasanjePocelo(List<Glas> glasovi)
47      {
48          if (glasovi.Count == 0)
49              return false;
50          return true;
51      }
```

GREŠKA 2:

Kod prije ispravke:

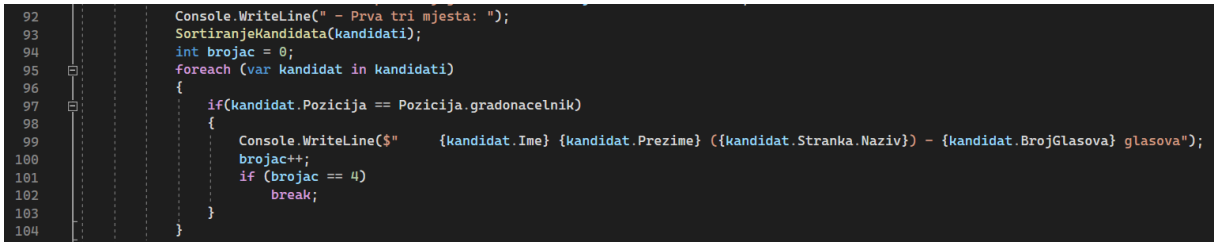
```
56      foreach (Glasac g in glasaci)
57      {
58          if (g.DaLiJeGlasaoZaGradonacelnika || g.DaLiJeGlasaoZaNacelnika)
59              ukupanBrojGlasova++;
60      }
```

Kod poslije ispravke:

```
56      foreach (Glasac g in glasaci)
57      {
58          if (g.DaLiJeGlasaoZaGradonacelnika || g.DaLiJeGlasaoZaNacelnika || g.DaLiJeGlasaoZaVijecnika)
59              ukupanBrojGlasova++;
60      }
```

GREŠKA 3:

Kod prije ispravke:



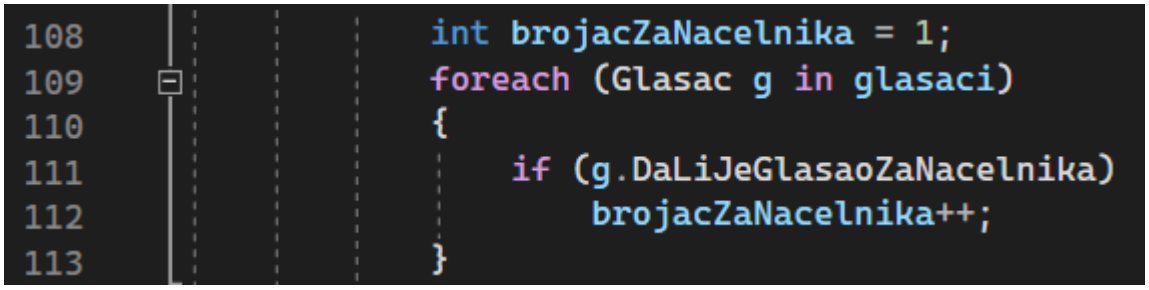
```
92          Console.WriteLine(" - Prva tri mjesta: ");
93          SortiranjeKandidata(kandidati);
94          int brojac = 0;
95          foreach (var kandidat in kandidati)
96          {
97              if(kandidat.Pozicija == Pozicija.gradonacelnik)
98              {
99                  Console.WriteLine($"     {kandidat.Ime} {kandidat.Prezime} ({kandidat.Stranka.Naziv}) - {kandidat.BrojGlasova} glasova");
100                 brojac++;
101                 if (brojac == 4)
102                     break;
103             }
104         }
```

Kod poslije ispravke:

```
92          Console.WriteLine(" - Prva tri mjesta: ");
93          SortiranjeKandidata(kandidati);
94          int brojac = 0;
95          foreach (var kandidat in kandidati)
96          {
97              if(kandidat.Pozicija == Pozicija.gradonacelnik)
98              {
99                  Console.WriteLine($"     {kandidat.Ime} {kandidat.Prezime} ({kandidat.Stranka.Naziv}) - {kandidat.BrojGlasova} glasova");
100                 brojac++;
101                 if (brojac == 3)
102                     break;
103             }
104         }
```

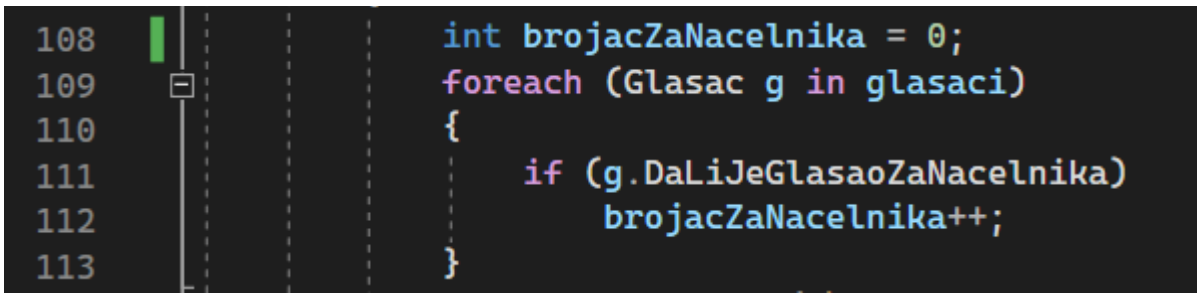
GREŠKA 4:

Kod prije ispravke:



```
108         int brojacZaNacelnika = 1;
109         foreach (Glasac g in glasaci)
110         {
111             if (g.DaLiJeGlasaoZaNacelnika)
112                 brojacZaNacelnika++;
113         }
```

Kod poslije ispravke:



```
108         int brojacZaNacelnika = 0;
109         foreach (Glasac g in glasaci)
110         {
111             if (g.DaLiJeGlasaoZaNacelnika)
112                 brojacZaNacelnika++;
113         }
```

GREŠKA 5:

Kod prije ispravke:

```
174         if (DaLiJeGlasanjePocelo(glasovi))
175         {
176             Console.WriteLine("Glasanje jos uvijek nije pocelo!");
177             return;
178         }
```

Kod poslije ispravke:

```
174         if (!DaLiJeGlasanjePocelo(glasovi))
175         {
176             Console.WriteLine("Glasanje jos uvijek nije pocelo!");
177             return;
178         }
```