KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS

Objektinis programavimas II (P175B123)

Darbų aplankas

Atliko:
IFF-7/7 gr. studentas
Dovydas Zamas
2018 m. gegužės 30 d.
Priėmė:
Andrej Afonin

KAUNAS 2018

TURINYS

1. Rekursija (L1)

1.1. Darbo užduotis

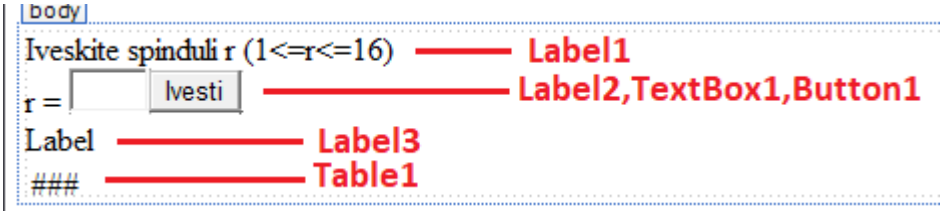
LD_6. Apskritimas.
Turime kvadrato formos languoto popieriaus lapą. Į lapo centre esantį langelių linijų susikirtimą statoma skriestuvo kojelė. Brėžiamas apskritimas, siekiantis lapo kraštus. Sunumeruokite, pradedant nuo 1, languoto popieriaus lapo langelius, pilnai patenkančius į apskritimo vidų, o kitiems langeliams priskirkite po 0. Numeravimo tvarka nėra svarbi. Atvaizduokite languoto popieriaus lapą matrica.
Spindulio r ($r \in N$) dydis nurodomas langelių kiekiu. Pavyzdžiui, kai r yra 5, matrica gali būti užpildoma taip:

0	0	0	0	0	0	0	0	0	0
0	0	25	26	27	28	29	30	0	0
0	17	18	19	20	21	22	23	24	0
0	9	10	11	12	13	14	15	16	0
0	1	2	3	4	5	6	7	8	0
0	31	32	33	34	35	36	37	38	0
0	39	40	41	42	43	44	45	46	0
0	47	48	49	50	51	52	53	54	0
0	0	55	56	57	58	59	60	0	0
0	0	0	0	0	0	0	0	0	0

Duomenys. r įvedamas klaviatūra ($1 \le r \le 16$).

Rezultatai. Gautą matricą spausdinkite ekrane.

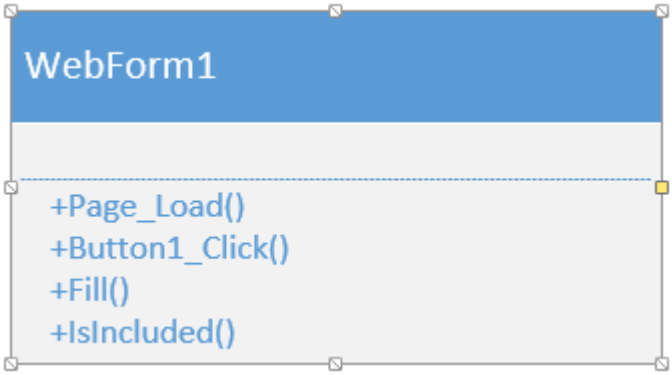
1.2. Grafinės vartotojo sąsajos schema



1.3. Sąsajoje panaudotų komponentų keičiamos savybės

Komponentas	Savybė	Reikšmė
Label1	Text	Iveskite spinduli r (1<=r<=16)
Label2	Text	r =
TextBox1	Height	16px
TextBox	Width	32px
Button1	Height	21px
Button1	Width	46px
Button1	Text	Ivesti
Table1	HorizontalAlign	Left

1.4. Klasių diagrama



1.5. Programos vartotojo vadovas

Ivesti apskritimo spinduli nuo 1 iki 16 ir rezultate pamatysite apskritima kurio langeliai pilnai patenka i apskritimo vidu.

1.6. Programos tekstas

```
using System;
using System.Web.UI.WebControls;

namespace LD6_Apskritimas
{
    public partial class WebForm1 : System.Web.UI.Page
    {
        /// <summary>
        /// pagrindinis puslapio uploadas
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        protected void Page_Load(object sender, EventArgs e)
        {


        }

        /// <summary>
        /// Nuskaito irasyta reiksme, iskviecia rekursini metoda fill
        /// ir spausdina rezultata i "table1"
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        protected void Button1_Click(object sender, EventArgs e)
        {
            int r = int.Parse(TextBox1.Text);
            int[,] matrix = new int[2 * r, 2 * r];
            int index = 0;
            if (r >= 1 && r <= 16)
            {
                Label3.Visible = false;
                Fill(ref matrix, ref index, r, r, r);
                for (int i = 0; i < 2 * r; i++)
                {
                    TableRow row = new TableRow();
                    for (int j = 0; j < 2 * r; j++)
                    {
                        TableCell skaicius = new TableCell();
                        skaicius.Text = matrix[i, j].ToString();
                        row.Cells.Add(skaicius);
                        skaicius.HorizontalAlign = HorizontalAlign.Right;
```

```
                    }
                    Table1.Rows.Add(row);
                }
            }
            else
            {
                Label3.Visible = true;
                Label3.Text = "Iveskite(tinkama) reiksme";
            }
        }

        /// <summary>
        /// Rekursija, matricos uzpildymui
        /// </summary>
        /// <param name="matrix"></param>
        /// <param name="index"></param>
        /// <param name="r"></param>
        /// <param name="x"></param>
        /// <param name="y"></param>
        static void Fill(ref int[,] matrix, ref int index, int r, int x, int y)
        {
            if (matrix[x, y] == 0 && IsIncluded(x, y, r))
                matrix[x, y] = index++;
            else
                return;

            if (x > 0)
                    Fill(ref matrix, ref index, r, x - 1, y);
            if (x + 1 < 2 * r)
                    Fill(ref matrix, ref index, r, x + 1, y);
            if (y > 0)
                    Fill(ref matrix, ref index, r, x, y - 1);
            if (y + 1 < 2 * r)
                    Fill(ref matrix, ref index, r, x, y + 1);

            return;
        }

        /// <summary>
        /// tikrina ar langelis patenka i apskritimo vidu
        /// </summary>
        /// <param name="x"></param>
        /// <param name="y"></param>
        /// <param name="r"></param>
        /// <returns></returns>
        static bool IsIncluded(int x, int y, int r)
        {
            //dist1 - up left, dist2 - up right, dist3 down left, dist4 - down right
            double dist1 = Math.Sqrt(Math.Pow((x - r), 2) + Math.Pow((y - r), 2));
            double dist2 = Math.Sqrt(Math.Pow((x - r + 1), 2) + Math.Pow((y - r), 2));
            double dist3 = Math.Sqrt(Math.Pow((x - r), 2) + Math.Pow((y - r + 1), 2));
            double dist4 = Math.Sqrt(Math.Pow((x - r + 1), 2) + Math.Pow((y - r + 1), 2));
            return dist1 <= r && dist2 <= r && dist3 <= r && dist4 <= r;
        }
    }
}
```

1.7. Pradiniai duomenys ir rezultatai

```
Pradiniai duomenys: 5
Rezultatas:
```

Įveskite spindulį r (1 <= r <=16)
r = 5 Excecute

0 0 0 0 0 0 0 0 0 0
0 0 21 20 5 4 43 44 0 0
0 34 22 19 6 3 42 45 60 0
0 33 23 18 7 2 41 46 59 0
0 32 24 17 8 1 40 47 58 0
0 31 25 16 9 38 39 48 57 0
0 30 26 15 10 37 54 49 56 0
0 29 27 14 11 36 53 50 55 0
0 0 28 13 12 35 52 51 0 0
0 0 0 0 0 0 0 0 0 0

```
Pradiniai duomenys:17
Rezultatas:
```

Įveskite spindulį r (1 <= r <=16)
r = 17 Excecute
Iveskite(tinkama) reiksme

```
Pradiniai duomenys: 16
Rezultatas:
```

Įveskite spindulį r (1 <= r <=16)
r = 16 Excecute

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 136 135 76 75 16 15 412 413 486 487 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 194 193 137 134 77 74 17 14 411 414 485 488 544 545 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 248 247 195 192 138 133 78 73 18 13 410 415 484 489 543 546 598 599 0 0 0 0 0 0 0
0 0 0 0 0 0 297 249 246 196 191 139 132 79 72 19 12 409 416 483 490 542 547 597 600 648 0 0 0 0 0 0
0 0 0 0 0 298 296 250 245 197 190 140 131 80 71 20 11 408 417 482 491 541 548 596 601 647 649 0 0 0 0 0
0 0 0 0 339 299 295 251 244 198 189 141 130 81 70 21 10 407 418 481 492 540 549 595 602 646 650 690 0 0 0 0
0 0 0 340 338 300 294 252 243 199 188 142 129 82 69 22 9 406 419 480 493 539 550 594 603 645 651 689 691 0 0 0
0 0 0 341 337 301 293 253 242 200 187 143 128 83 68 23 8 405 420 479 494 538 551 593 604 644 652 688 692 0 0 0
0 0 371 342 336 302 292 254 241 201 186 144 127 84 67 24 7 404 421 478 495 537 552 592 605 643 653 687 693 722 0 0
0 0 370 343 335 303 291 255 240 202 185 145 126 85 66 25 6 403 422 477 496 536 553 591 606 642 654 686 694 721 0 0
0 372 369 344 334 304 290 256 239 203 184 146 125 86 65 26 5 402 423 476 497 535 554 590 607 641 655 685 695 720 723 0
0 373 368 345 333 305 289 257 238 204 183 147 124 87 64 27 4 401 424 475 498 534 555 589 608 640 656 684 696 719 724 0
0 374 367 346 332 306 288 258 237 205 182 148 123 88 63 28 3 400 425 474 499 533 556 588 609 639 657 683 697 718 725 0
0 375 366 347 331 307 287 259 236 206 181 149 122 89 62 29 2 399 426 473 500 532 557 587 610 638 658 682 698 717 726 0
0 376 365 348 330 308 286 260 235 207 180 150 121 90 61 30 1 398 427 472 501 531 558 586 611 637 659 681 699 716 727 0
0 377 364 349 329 309 285 261 234 208 179 151 120 91 60 31 396 397 428 471 502 530 559 585 612 636 660 680 700 715 728 0
0 378 363 350 328 310 284 262 233 209 178 152 119 92 59 32 395 456 429 470 503 529 560 584 613 635 661 679 701 714 729 0
0 379 362 351 327 311 283 263 232 210 177 153 118 93 58 33 394 455 430 469 504 528 561 583 614 634 662 678 702 713 730 0
0 380 361 352 326 312 282 264 231 211 176 154 117 94 57 34 393 454 431 468 505 527 562 582 615 633 663 677 703 712 731 0
0 381 360 353 325 313 281 265 230 212 175 155 116 95 56 35 392 453 432 467 506 526 563 581 616 632 664 676 704 711 732 0
0 0 359 354 324 314 280 266 229 213 174 156 115 96 55 36 391 452 433 466 507 525 564 580 617 631 665 675 705 710 0 0
0 0 358 355 323 315 279 267 228 214 173 157 114 97 54 37 390 451 434 465 508 524 565 579 618 630 666 674 706 709 0 0
0 0 0 356 322 316 278 268 227 215 172 158 113 98 53 38 389 450 435 464 509 523 566 578 619 629 667 673 707 0 0 0
0 0 0 357 321 317 277 269 226 216 171 159 112 99 52 39 388 449 436 463 510 522 567 577 620 628 668 672 708 0 0 0
0 0 0 0 320 318 276 270 225 217 170 160 111 100 51 40 387 448 437 462 511 521 568 576 621 627 669 671 0 0 0 0
0 0 0 0 0 319 275 271 224 218 169 161 110 101 50 41 386 447 438 461 512 520 569 575 622 626 670 0 0 0 0 0
0 0 0 0 0 0 274 272 223 219 168 162 109 102 49 42 385 446 439 460 513 519 570 574 623 625 0 0 0 0 0 0
0 0 0 0 0 0 0 273 222 220 167 163 108 103 48 43 384 445 440 459 514 518 571 573 624 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 221 166 164 107 104 47 44 383 444 441 458 515 517 572 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 165 106 105 46 45 382 443 442 457 516 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1.8. Dėstytojo pastabos

- Nepalikti saito elementų pavadinimų kaip (default).
- Pvz.: Label1 => LabelRadius.
- Nepakeistas puslapio title.
- Neužkomentuoti metodų parametrai.

2. Dinaminis atminties valdymas (L2)

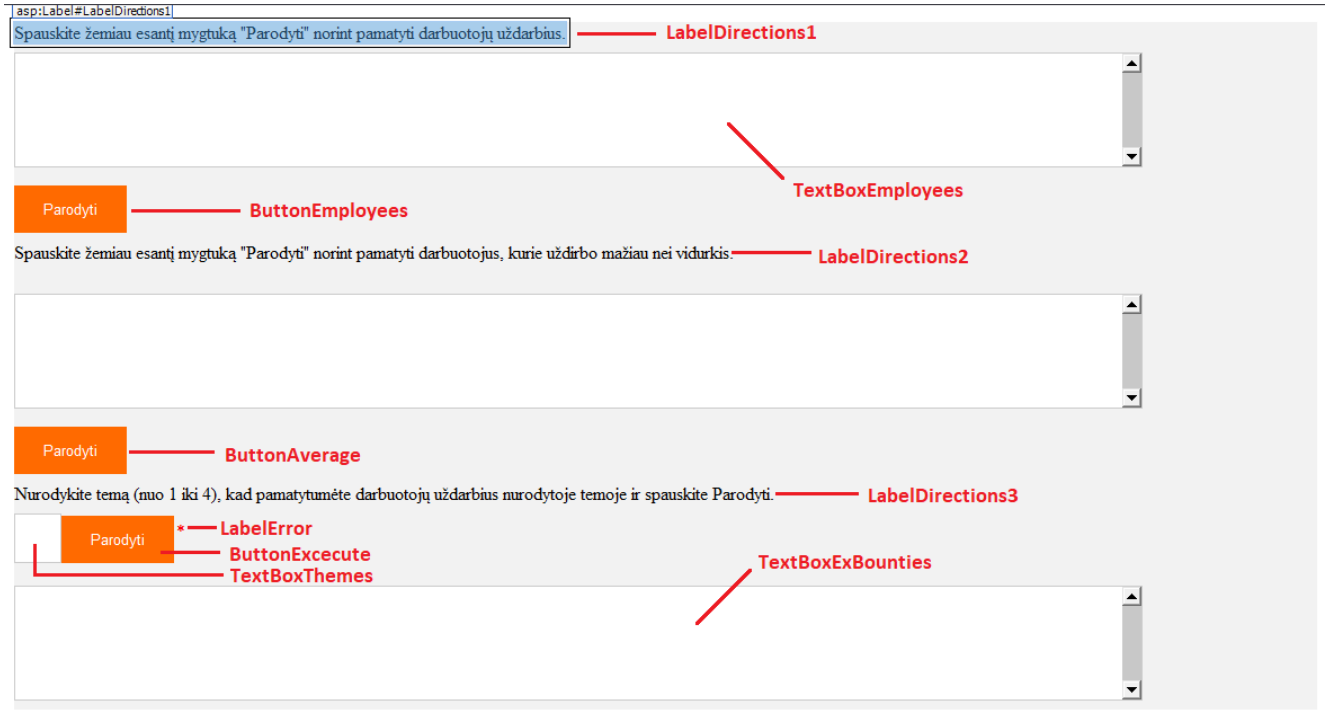
2.1. Darbo užduotis

LD_6. Premijos. Moksliniai darbuotojai atliko darbus 4 skirtingose temose. Visos temos gavo premijas. Remiantis darbuotojų indėliais, suskaičiuokite kiekvienam darbuotojui priklausančios premijos dydį pagal kiekvieną temą atskirai ir bendrą premijų sumą. Darbuotojo indėlis rodo, kokia premijos dalis jam priklauso. Indėlis gali būti išreikštas bet kokiu skaičiumi. Bet tas pats matas naudojamas visiems darbuotojams. Neišdalinkite pinigų daugiau, negu turite, ir turite išdalinti visus pinigus. Suformuokite sąrašą darbuotojų, kurie uždirbo mažiau už vidurkį. Suformuokite kiekvienam bankui atskirai pavedimų sąrašą. Duomenys:

- Tekstiniame faile U6a.txt duota informacija apie darbuotojus: darbuotojų asmens kodai, pavardės, vardai, banko pavadinimas, sąskaitos numeris.
- Tekstiniame faile U6b.txt duota informacija apie indėlius į darbą: pirmoje eilutėje - premijų dydžiai, tolesnėse eilutėse - darbuotojų asmens kodai, darbuotojų indėliai, kurie išreikšti naudingumo koeficientu, į eilinę temą. Informacija apie darbuotoją užima vieną eilutę.

Spausdinamas sąrašas turi būti surikiuotas abėcėlės tvarka. Sudarykite nurodytos temos (įvedama klaviatūra) darbuotojų uždarbių sąrašą.

2.2. Grafinės vartotojo sąsajos schema

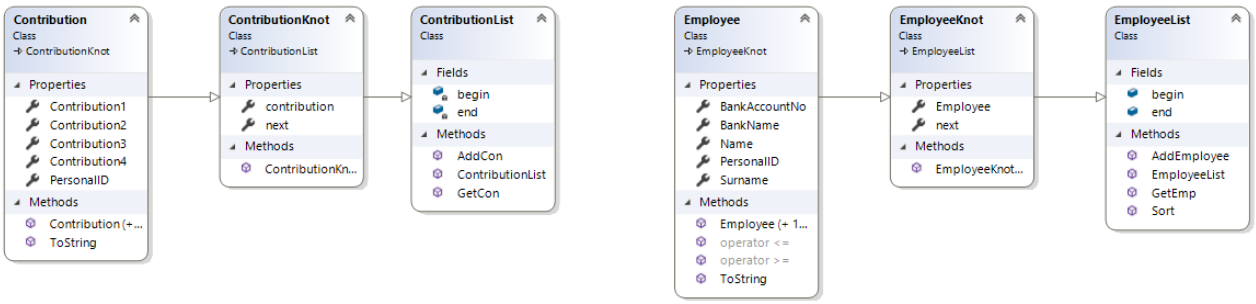


2.3. Sąsajoje panaudotų komponentų keičiamos savybės

Komponentas	Savybė	Reikšmė
LabelDirections1	Text	Spauskite žemiau esantį mygtuką "Parodyti" norint pamatyti darbuotojų uždarbius.
LabelDirections2	Text	Spauskite žemiau esantį mygtuką "Parodyti" norint pamatyti darbuotojus, kurie uždirbo mažiau nei vidurkis.
LabelDirections3	Text	Nurodykite temą (nuo 1 iki 4), kad pamatytumėte darbuotojų uždarbius nurodytoje temoje ir spauskite Parodyti.
TextBoxEmployees	Height	100px

TextBoxEmployees	Width	1000px
TextBoxEmployees	TextMode	MultiLine
TextBoxLessThanAvg	Height	100px
TextBoxLessThanAvg	Width	1000px
TextBoxLessThanAvg	TextMode	MultiLine
TextBoxExBounties	Height	100px
TextBoxExBounties	Width	1000px
TextBoxExBounties	TextMode	MultiLine
ButtonEmployees	Height	42px
ButtonEmployees	Width	100px
ButtonEmployees	Text	Parodyti
ButtonAverage	Height	42px
ButtonAverage	Width	100px
ButtonAverage	Text	Parodyti
ButtonExcecute	Height	42px
ButtonExcecute	Width	100px
ButtonExcecute	Text	Parodyti

2.4. Klasių diagrama



2.5. Programos vartotojo vadovas

Norint pamatyti darbuotojų uždarbius, spaudžiame mygtuką „ButtonEmployees“, darbuotojus, kurie uždirbo mažiau negu vidurkis – „ButtonAverage“, nurodytos temos darbuotojų uždarbius pirmiausia įrašome temą (nuo 1 iki 4) į „TextBoxThemes“ laukelį, tada spaudžame mygtuką „ButtonExcecute“, jeigu reikšmė netinkama išvysite pranešimą:“ `Įveskite tinkamą reikšmę.`“

2.6. Programos tekstas

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace _2laboras
{
    /// <summary>
    ///
    /// </summary>
    public class Contribution
    {
        public long PersonalID { get; set; }
        public double Contribution1 { get; set; }
        public double Contribution2 { get; set; }
        public double Contribution3 { get; set; }
        public double Contribution4 { get; set; }

        /// <summary>
        ///
        /// </summary>
        public Contribution()
        {

        }

        /// <summary>
        /// Konstruktorius
        /// </summary>
        /// <param name="personalID">Asmens kodas</param>
        /// <param name="contribution1">Įnašas į pirmą temą</param>
        /// <param name="contribution2">Įnašas į antrą temą</param>
        /// <param name="contribution3">Įnašas į trečią temą</param>
        /// <param name="contribution4">Įnašas į ketvirtą temą</param>
        public Contribution(long personalID, double contribution1, double contribution2,
                            double contribution3, double contribution4)
        {
            PersonalID = personalID;
            Contribution1 = contribution1;
            Contribution2 = contribution2;
            Contribution3 = contribution3;
            Contribution4 = contribution4;
        }

        /// <summary>
        /// ToString užklojimas
        /// </summary>
        /// <returns></returns>
        public override String ToString()
        {
            return String.Format("| {0,10} | {1,10} | {2,10} | {3, 10} | {4,10} |",
PersonalID, Contribution1, Contribution2, Contribution3, Contribution4);
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace _2laboras
{
    /// <summary>
    ///
    /// </summary>
    public class ContributionKnot
    {
        public Contribution contribution { get; set; }
        public ContributionKnot next { get; set; }

        /// <summary>
        ///
        /// </summary>
        public ContributionKnot() { }

        /// <summary>
        ///
        /// </summary>
        /// <param name="contribution"></param>
        /// <param name="next"></param>
        public ContributionKnot(Contribution contribution, ContributionKnot next)
        {
            this.contribution = contribution;
            this.next = next;
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace _2laboras
{
    /// <summary>
    ///
    /// </summary>
    public class ContributionList
    {
        private ContributionKnot begin, end;

        /// <summary>
        ///
        /// </summary>
        public ContributionList()
        {
            begin = null;
            end = null;
        }

        /// <summary>
        /// Prideda darbuotojo darbo įnašus
        /// </summary>
        /// <param name="contribution"></param>
        public void AddCon(Contribution contribution)
        {
            ContributionKnot dd = new ContributionKnot(contribution, null);
            if (begin != null)
            {
                end.next = dd;
                end = dd;
            }
            else
            {
                begin = dd;
                end = dd;
            }
        }

        public ContributionKnot GetCon()
        {
            return begin;
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace _2laboras
{
    /// <summary>
    ///
    /// </summary>
    public class Employee
    {
        public long PersonalID { get; set; }        //Asmens kodas
        public string Surname { get; set; }         //Pavardė
        public string Name { get; set; }            //Vardas
        public string BankName { get; set; }        //Banko pavadinimas
        public string BankAccountNo { get; set; }  //Banko sąskaitos numeris

        /// <summary>
        ///
        /// </summary>
        public Employee()
        {

        }

        /// <summary>
        /// Konstruktorius
        /// </summary>
        /// <param name="personalID">Asmens kodas</param>
        /// <param name="surname">Pavardė</param>
        /// <param name="name">Vardas</param>
        /// <param name="bankName">Banko pavadinimas</param>
        /// <param name="bankAccountNo">Banko sąskaitos numeris</param>
        public Employee(long personalID, string surname, string name,
                        string bankName, string bankAccountNo)
        {
            PersonalID = personalID;
            Surname = surname;
            Name = name;
            BankName = bankName;
            BankAccountNo = bankAccountNo;
        }

        /// <summary>
        /// ToString užklojimas
        /// </summary>
        /// <returns></returns>
        public override String ToString()
        {
            return String.Format("| {0,10} | {1,-10} | {2,-10} | {3,-15} | {4,10} |",
                                 PersonalID, Surname, Name, BankName, BankAccountNo);
        }

        /// <summary>
        /// Palyginimo operatoriaus užklojimas
        /// </summary>
        public static bool operator <=(Employee lhs, Employee rhs)
        {
            if (lhs.Surname.CompareTo(rhs.Surname) < 0)
            {
                return true;
            }
            return false;
        }

        /// <summary>
        /// Palyginimo operatoriaus užklojimas
```

```
        /// </summary>
        public static bool operator >=(Employee lhs, Employee rhs)
        {
            if (lhs.Surname.CompareTo(rhs.Surname) > 0)
            {
                return true;
            }
            return false;
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace _2laboras
{
    public class EmployeeKnot
    {
        public Employee Employee { get; set; }
        public EmployeeKnot next { get; set; }

        public EmployeeKnot() { }

        public EmployeeKnot(Employee employee, EmployeeKnot next)
        {
            Employee = employee;
            this.next = next;
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace _2laboras
{
    public class EmployeeList
    {
        public  EmployeeKnot begin, end;
        public EmployeeList()
        {
            begin = null;
            end = null;
        }

        /// <summary>
        /// Prideda darbuotoją
        /// </summary>
        /// <param name="emplyee">Darbuotojas</param>
        public void AddEmployee(Employee emplyee)
        {
            EmployeeKnot dd = new EmployeeKnot(emplyee, null);
            if (begin != null)
            {
                end.next = dd;
                end = dd;
            }
            else
            {
                begin = dd;
                end = dd;
            }
        }

        public EmployeeKnot GetEmp()
        {
            return begin;
        }

        /// <summary>
        /// Rikiavimas
        /// </summary>
        public void Sort()
        {
            for (EmployeeKnot d1 = begin; d1 != null; d1 = d1.next)
            {
                EmployeeKnot min = d1;
                for (EmployeeKnot d2 = d1; d2 != null; d2 = d2.next)
                    if (d2.Employee <= min.Employee)
                    {
                        min = d2;
                        Employee emp = d1.Employee;
                        d1.Employee = min.Employee;
                        min.Employee = emp;
                    }
            }
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.IO;

namespace _2laboras
{
    public partial class index : System.Web.UI.Page
    {
        public const string defFileLocation = "";
        public const string failas1 = "U6a.txt";
        public const string failas2 = "U6b.txt";
        EmployeeList employeeList = ReadData1(failas1);
        ContributionList contributionList = ReadData2(failas2);
        double[] bounties = ReadData3(failas2);

        protected void Page_Load(object sender, EventArgs e)
        {
            if (!Page.IsPostBack)
            { }
            else
            { }

            EmployeeList swedEmp = DBank("Swedbank", employeeList);
            EmployeeList SEBEmp = DBank("SEB bankas", employeeList);
            EmployeeList empListLAvg = LAvg(employeeList, contributionList, bounties);
            swedEmp.Sort();
            SEBEmp.Sort();
            empListLAvg.Sort();
            employeeList.Sort();
            writeToFile(employeeList, bounties, contributionList, empListLAvg, swedEmp,
                        SEBEmp);
        }

        /// <summary>
        /// Nuskaitymo metodas pirmam failui:"U6a.txt"
        /// </summary>
        /// <returns>Darbuotojų sąrašą</returns>
        static EmployeeList ReadData1(string failas)
        {
            EmployeeList employeeList = new EmployeeList();
            string line;
            StreamReader reader = new StreamReader("Z:\\Objektinis
                                                    programavimas\\2semestras\\2laboras\\
                                                    2laboras\\App_Data\\" + failas);
            while (null != (line = reader.ReadLine()))
            {
                string[] values = line.Split(';');
                Employee employee = new Employee(long.Parse(values[0]), values[1], values[2],
                                                 values[3], values[4]);
                employeeList.AddEmployee(employee);
            }
            reader.Close();
            return employeeList;
        }
```

```
/// <summary>
/// Nuskaitymo metodas antram failui:"U6b.txt"
/// </summary>
/// <returns>Darbuotojų asmens kodus ir darbo įnašus į atitinkamas temas.</returns>
static ContributionList ReadData2(string failas2)
{
    ContributionList contributionList = new ContributionList();
    string line;
    StreamReader reader = new StreamReader("Z:\\Objektinis
                                           programavimas\\2semestras\\2laboras\\
                                           2laboras\\App_Data\\"+failas2);
    while (null != (line = reader.ReadLine()))
    {
        string[] values = line.Split(';');

        if (values.Count() != 4)
        {
            Contribution contribution = new Contribution(long.Parse(values[0]),
                                double.Parse(values[1]), double.Parse(values[2]),
                                double.Parse(values[3]), double.Parse(values[4]));
            contributionList.AddCon(contribution);
        }
    }
    reader.Close();
    return contributionList;
}

/// <summary>
/// Nuskaitymo metodas antram failui:"U6b.txt"
/// </summary>
/// <returns>Premijų dydžius atitinkamom temom</returns>
static double[] ReadData3(string failas2)
{
    double[] bounties = new double[4];
    string line;
    StreamReader reader = new StreamReader("Z:\\Objektinis
                                           programavimas\\2semestras\\2laboras\\
                                           2laboras\\App_Data\\"+failas2);
    while (null != (line = reader.ReadLine()))
    {
        string[] values = line.Split(';');
        if (values.Count() == 4)
        {
            for (int i = 0; i < values.Count(); i++)
            {
                bounties[i] = int.Parse(values[i]);
            }
        }
    }
    reader.Close();
    return bounties;
}
```

```
/// <summary>
/// Randa premijas skirtas darbuotojams.
/// </summary>
/// <param name="contributionlist">Darbo įnašų sąrašas</param>
/// <param name="bounties">Premijos skirtos temoms</param>
/// <param name="personalID">Darbuotojų asmens kodas</param>
/// <returns>Premijas skirtas darbuotojui</returns>
static double[] FindBounties(ContributionList contributionlist, double[] bounties,
                             long personalID)
{
    double[] empBounties = new double[4];
    for (ContributionKnot d = contributionlist.GetCon(); d != null; d = d.next)
    {
        if (personalID == d.contribution.PersonalID)
        {
            empBounties[0] = d.contribution.Contribution1 * bounties[0];
            empBounties[1] = d.contribution.Contribution2 * bounties[1];
            empBounties[2] = d.contribution.Contribution3 * bounties[2];
            empBounties[3] = d.contribution.Contribution4 * bounties[3];
        }
    }
    return empBounties;
}

/// <summary>
/// Randa vidurkį
/// </summary>
/// <param name="digits">skaičiai</param>
/// <returns>Visų premijų vidurkį</returns>
static double FindAvg(double[] digits)
{
    double sum = 0;
    for (int i = 0; i < digits.Count(); i++)
    {
        sum += digits[i];
    }
    return sum / digits.Count();
}

/// <summary>
/// Randa premijų, skirtas darbuotojui, sumą.
/// </summary>
/// <param name="contributionList">Darbo įnašų sąrašas</param>
/// <param name="bounties">Premijos skirtos temoms</param>
/// <param name="personalID">Darbuotojų asmens kodas</param>
/// <returns>Premijų sumą</returns>
static double FindSum(ContributionList contributionList, double[] bounties, long
                      personalID)
{
    double sum = 0;
    double[] bounties1 = FindBounties(contributionList, bounties, personalID);
    for (int i = 0; i < bounties1.Count(); i++)
    {
        sum += bounties1[i];
    }
    return sum;
}
```

```
/// <summary>
/// Randa darbuotous kurie uždirbo mažiau nei vidurkis.
/// </summary>
/// <param name="employeeList">Darbuotojų sąrašas</param>
/// <param name="contributionList">Darbo įnašų sąrašas</param>
/// <param name="bounties">Premijos</param>
/// <returns>Darbuotojų sąrašą, kurių premijų suma žemesnė už vidurkį.</returns>
static EmployeeList LAvg(EmployeeList employeeList, ContributionList contributionList,
                         double[] bounties)
{
    EmployeeList empListLAvg = new EmployeeList();
    double avg = FindAvg(bounties);
    for (EmployeeKnot d = employeeList.GetEmp(); d != null; d = d.next)
    {
        double sum = FindSum(contributionList, bounties, d.Employee.PersonalID);

        if (sum < avg)
        {
            empListLAvg.AddEmployee(d.Employee);
        }
    }
    return empListLAvg;
}

/// <summary>
/// Randa nurodytos temos skirtą premiją darbuotojui.
/// </summary>
/// <param name="contributionList">>Darbo įnašų sąrašas</param>
/// <param name="index">Nurodo kuri tema</param>
/// <param name="bounties">Premijos</param>
/// <param name="personalID">Asmens kodas</param>
/// <returns>Atitinkamos temos premiją skirta darbuotojui</returns>
static double FindExBounty(ContributionList contributionList, int index, double[]
                           bounties, long personalID)
{
    double bounty = 0;
    for (ContributionKnot d = contributionList.GetCon(); d != null; d = d.next)
    {
        if (personalID == d.contribution.PersonalID)
        {
            if (index == 1)
            {
                bounty = bounties[0] * d.contribution.Contribution1;
            }

            if (index == 2)
            {
                bounty = bounties[1] * d.contribution.Contribution2;
            }
            if (index == 3)
            {
                bounty = bounties[2] * d.contribution.Contribution3;
            }
            if (index == 4)
            {
                bounty = bounties[3] * d.contribution.Contribution4;
            }
        }
    }
    return bounty;
}
```

```
/// <summary>
/// Atrenka darbuotojus kurie turi nurodytą banką.
/// </summary>
/// <param name="bankName">Banko pavadinimas</param>
/// <param name="employeeList">Darbuotojų sąrašas</param>
/// <returns></returns>
static EmployeeList DBank(string bankName, EmployeeList employeeList)
{
    EmployeeList employees = new EmployeeList();
    for (EmployeeKnot d = employeeList.GetEmp(); d != null; d = d.next)
    {
        if (d.Employee.BankName.Equals(bankName))
        {
            employees.AddEmployee(d.Employee);
        }
    }
    return employees;
}

/// <summary>
/// Išsaugo visus duomenis į .txt failą
/// </summary>
/// <param name="employeeList">Darbuotojų sąrašas</param>
/// <param name="bounties">Premijos</param>
/// <param name="contributionList">Darbo įnašų sąrašas</param>
/// <param name="LAvgEmp">Darbuotojų sarašas,  kurių premijų suma
                          mažesnė už vidurkį</param>
/// <param name="swedEmp">Darbuotojų sąrašas, kurie turi Swedbank</param>
/// <param name="SEBEmp">Darbuotojų sąrašas, kurie turi SEB banką</param>
static void writeToFile(EmployeeList employeeList, double[] bounties, ContributionList
                        contributionList, EmployeeList LAvgEmp,
                        EmployeeList swedEmp, EmployeeList SEBEmp)
{
    StreamWriter writer = new StreamWriter("Z:\\Objektinis
                                            programavimas\\2semestras\\2laboras\\
                                            2laboras\\App_Data\\Results.txt");
    writer.WriteLine("Duomenys iš U6a.txt:");
    writer.WriteLine(" ------------------------------------------------------------");
    for (EmployeeKnot d = employeeList.GetEmp(); d != null; d = d.next)
    {
        writer.WriteLine(d.Employee);
    }
    writer.WriteLine("Duomenys iš U6b.txt:");
    writer.WriteLine(" ------------------------------------------------------------");
    for (int i = 0; i < 4; i++)
    {
        writer.Write(bounties[i] + " ");
    }
    writer.WriteLine("");
    for (ContributionKnot d = contributionList.GetCon(); d != null; d = d.next)
    {
        writer.WriteLine(d.contribution);
    }
    writer.WriteLine("Premijų dydžiai");
    writer.WriteLine(" ------------------------------------------------------------");
    for (EmployeeKnot d = employeeList.GetEmp(); d != null; d = d.next)
    {
        double[] empBounties = empBounties = FindBounties(contributionList, bounties,
                                                          d.Employee.PersonalID);
        writer.Write(d.Employee);
        for (int i = 0; i < empBounties.Count(); i++)
        {
            writer.Write(empBounties[i] + " ");
        }
        writer.WriteLine("Iš viso: " + FindSum(contributionList, bounties,
                                               d.Employee.PersonalID));
    }
```

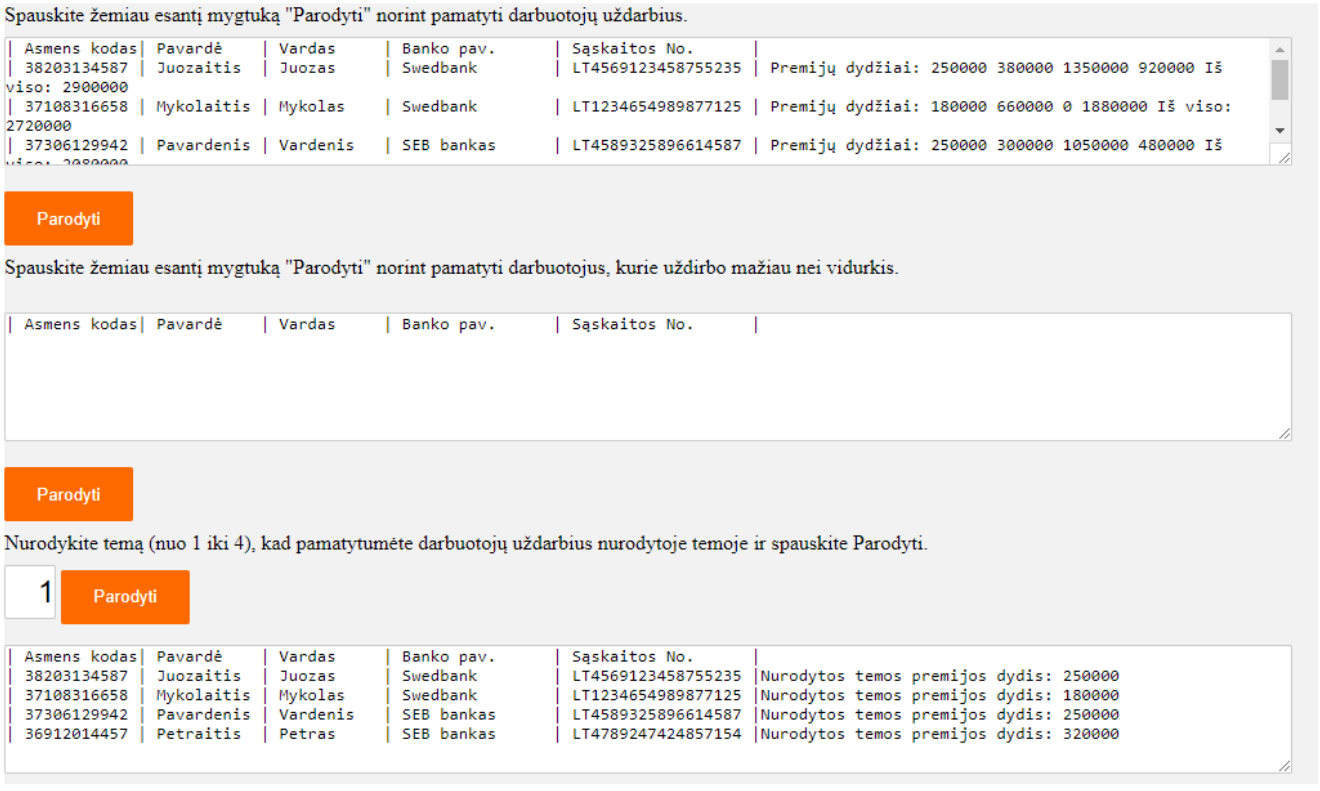
```
            writer.WriteLine("Uždirbo mažiau nei vidurkis");
            writer.WriteLine(" ------------------------------------------------------------------
--");
            for (EmployeeKnot d = LAvgEmp.GetEmp(); d != null; d = d.next)
            {
                writer.WriteLine(d.Employee);
            }
            writer.WriteLine("Swedbank pavedimai");
            writer.WriteLine(" ------------------------------------------------------------------
--");
            for (EmployeeKnot d = swedEmp.GetEmp(); d != null; d = d.next)
            {
                writer.WriteLine(d.Employee);
            }
            writer.WriteLine("SEB bankas pavedimai");
            writer.WriteLine(" ------------------------------------------------------------------
--");
            for (EmployeeKnot d = SEBEmp.GetEmp(); d != null; d = d.next)
            {
                writer.WriteLine(d.Employee);
            }
            writer.Close();
        }

        /// <summary>
        ///
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        protected void Button1_Click(object sender, EventArgs e)
        {

            TextBoxEmployees.Text = String.Format("| {0,-10}| {1,-10} | {2,-10} | {3,-15} |
{4,-18} |" + "\n", "Asmens kodas", "Pavardė", "Vardas", "Banko pav.", "Sąskaitos No.");
            for (EmployeeKnot d = employeeList.GetEmp(); d != null; d = d.next)
            {

                double[] empBounties = empBounties = FindBounties(contributionList, bounties,
d.Employee.PersonalID);
                TextBoxEmployees.Text += d.Employee;
                TextBoxEmployees.Text += " Premijų dydžiai: ";
                for (int i = 0; i < empBounties.Count(); i++)
                {
                    TextBoxEmployees.Text += empBounties[i] + " ";
                }
                TextBoxEmployees.Text += "Iš viso: " + FindSum(contributionList, bounties,
d.Employee.PersonalID) + "\n";
            }
        }

        /// <summary>
        ///
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        protected void Button2_Click(object sender, EventArgs e)
        {
            EmployeeList empListLAvg = LAvg(employeeList, contributionList, bounties);
            TextEmpLAvg.Text = String.Format("| {0,-10}| {1,-10} | {2,-10} | {3,-15} | {4,-18}
|" + "\n", "Asmens kodas", "Pavardė", "Vardas", "Banko pav.", "Sąskaitos No.");
            for (EmployeeKnot d = empListLAvg.GetEmp(); d != null; d = d.next)
            {
                TextEmpLAvg.Text += d.Employee + "\n";
            }
        }

        protected void Button3_Click(object sender, EventArgs e)
        {
```

```
            int index;
            {
                if (IsPostBack)
                {
                    TextBoxExBounties.Text = "";
                    int.TryParse(TextBoxThemes.Text, out index);
                    if (index >= 1 && index <= 4)
                    {
                        LabelError.Visible = false;
                        TextBoxExBounties.Text = String.Format("| {0,-10}| {1,-10} | {2,-10} |
                                                               {3,-15} | {4,-18} |" + "\n",
                                                               "Asmens kodas", "Pavardė",
                                                               "Vardas", "Banko pav.",
                                                               "Sąskaitos No.");
                        for (EmployeeKnot d = employeeList.GetEmp(); d != null; d = d.next)
                        {
                            double[] empBounties  = FindBounties(contributionList, bounties,
                                                    d.Employee.PersonalID);
                            TextBoxExBounties.Text += d.Employee;
                            TextBoxExBounties.Text += "Nurodytos temos premijos dydis: " +
                                                      FindExBounty(contributionList, index,
                                                      bounties, d.Employee.PersonalID) +
                                                      "\n";
                        }
                    }
                    else
                    {
                        LabelError.Visible = true;
                        LabelError.Text = "Įveskite tinkamą reikšmę.";
                    }
                }
            }
        }
    }
}
```

2.7. Pradiniai duomenys ir rezultatai

```
Pradiniai duomenys:

U6a.txt:
37108316658;Mykolaitis;Mykolas;Swedbank;LT1234654989877125
36912014457;Petraitis;Petras;SEB bankas;LT4789247424857154
37306129942;Pavardenis;Vardenis;SEB bankas;LT4589325896614587
38203134587;Juozaitis;Juozas;Swedbank;LT4569123458755235

U6b.txt:
10000;20000;30000;40000
37108316658;0,18;0,33;0;0,47
36912014457;0,32;0,33;0,20;0,18
37306129942;0,25;0,15;0,35;0,12
38203134587;0,25;0,19;0,45;0,23
```

```
Rezultatai:
Duomenys iš U6a.txt:
 ---------------------------------------------------------------------
| 38203134587 | Juozaitis  | Juozas     | Swedbank         | LT4569123458755235 |
| 37108316658 | Mykolaitis | Mykolas    | Swedbank         | LT1234654989877125 |
| 37306129942 | Pavardenis | Vardenis   | SEB bankas       | LT4589325896614587 |
| 36912014457 | Petraitis  | Petras     | SEB bankas       | LT4789247424857154 |
Duomenys iš U6b.txt:
 ---------------------------------------------------------------------
10000 20000 30000 40000
| 37108316658 |         18 |         33 |          0 |         47 |
| 36912014457 |         32 |         33 |         20 |         18 |
| 37306129942 |         25 |         15 |         35 |         12 |
| 38203134587 |         25 |         19 |         45 |         23 |
Premijų dydžiai
 ---------------------------------------------------------------------
| 38203134587 | Juozaitis  | Juozas     | Swedbank         | LT4569123458755235
|250000 380000 1350000 920000 Iš viso: 2900000
| 37108316658 | Mykolaitis | Mykolas    | Swedbank         | LT1234654989877125
|180000 660000 0 1880000 Iš viso: 2720000
| 37306129942 | Pavardenis | Vardenis   | SEB bankas       | LT4589325896614587
|250000 300000 1050000 480000 Iš viso: 2080000
| 36912014457 | Petraitis  | Petras     | SEB bankas       | LT4789247424857154
|320000 660000 600000 720000 Iš viso: 2300000
Uždirbo mažiau nei vidurkis
 ---------------------------------------------------------------------
Swedbank pavedimai
 ---------------------------------------------------------------------
| 38203134587 | Juozaitis  | Juozas     | Swedbank         | LT4569123458755235 |
| 37108316658 | Mykolaitis | Mykolas    | Swedbank         | LT1234654989877125 |
SEB bankas pavedimai
 ---------------------------------------------------------------------
| 37306129942 | Pavardenis | Vardenis   | SEB bankas       | LT4589325896614587 |
| 36912014457 | Petraitis  | Petras     | SEB bankas       | LT4789247424857154 |
```



2.8. Dėstytojo pastabos

- Trūksta komentarų.
- Pagrindiniam kode mygtukų pavadinimai nepakeisti.
- Nuskaitymo metodai vykdomi kaskart kai puslapis yra perkraunamas.
- Programą galima optimizuoti iškeliant kai kuriuos veiksmus į metodus.

Programos kodas:5
Testas:0
Ataskaita: -

3. Bendrinės klasės ir sąsajos (L3)

3.1. Darbo užduotis

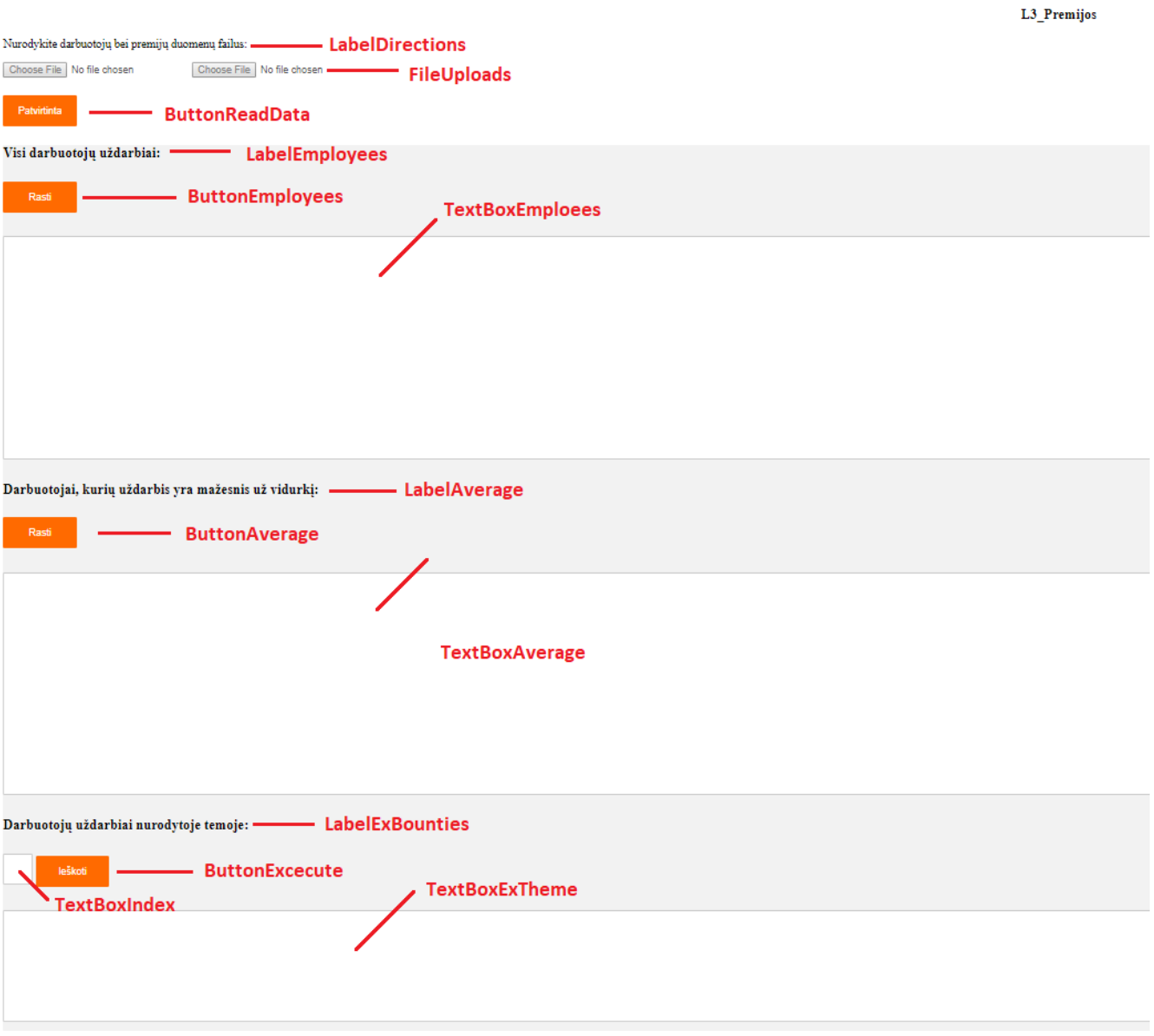
LD_6. Premijos.

Moksliniai darbuotojai atliko darbus 4 skirtingose temose. Visos temos gavo premijas. Remiantis darbuotojų indėliais, suskaičiuokite kiekvienam darbuotojui priklausančios premijos dydį pagal kiekvieną temą atskirai ir bendrą premijų sumą. Darbuotojo indėlis rodo, kokia premijos dalis jam priklauso. Indėlis gali būti išreikštas bet kokiu skaičiumi. Bet tas pats matas naudojamas visiems darbuotojams. Neišdalinkite pinigų daugiau, negu turite, ir turite išdalinti visus pinigus. Suformuokite sąrašą darbuotojų, kurie uždirbo mažiau už vidurkį. Suformuokite kiekvienam bankui atskirai pavedimų sąrašą. Duomenys:

- Tekstiniame faile U6a.txt duota informacija apie darbuotojus: darbuotojų asmens kodai, pavardės, vardai, banko pavadinimas, sąskaitos numeris.
- Tekstiniame faile U6b.txt duota informacija apie indėlius į darbą: pirmoje eilutėje - premijų dydžiai, tolesnėse eilutėse - darbuotojų asmens kodai, darbuotojų indėliai, kurie išreikšti naudingumo koeficientu, į eilinę temą. Informacija apie darbuotoją užima vieną eilutę.

Spausdinamas sąrašas turi būti surikiuotas abėcėlės tvarka. Sudarykite nurodytos temos (įvedama klaviatūra) darbuotojų uždarbių sąrašą.

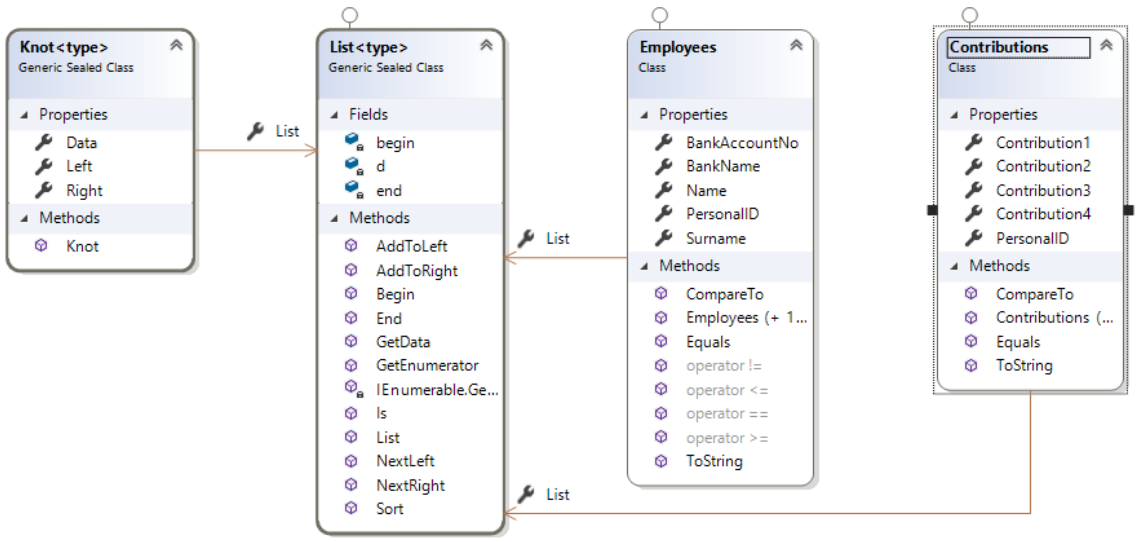
3.2. Grafinės vartotojo sąsajos schema



3.3. Sąsajoje panaudotų komponentų keičiamos savybės

Komponentas	Savybė	Reikšmė
ButtonReadData	Text	Patvirtinti
ButtonReadData	Height	42px
ButtonReadData	Width	100px
ButtonEmployees	Text	Rasti
ButtonEmployees	Height	42px
ButtonEmployees	Width	100px
ButtonAverage	Text	Rasti
ButtonAverage	Height	42px
ButtonAverage	Width	100px
ButtonExcecute	Text	Ieškoti
ButtonExcecute	Height	42px
ButtonExcecute	Width	100px
TextBoxEmployees	TextMode	MultiLine
TextBoxEmployees	Height	200px
TextBoxEmployees	Width	1000px
TextBoxLAverage	TextMode	MultiLine
TextBoxLAverage	Height	200px
TextBoxLAverage	Width	1000px
TextBoxIndex	Height	40px
TextBoxIndex	Width	40px
TextBoxExTheme	TextMode	MultiLine
TextBoxExTheme	Height	200px
TextBoxExTheme	Width	1000px

3.4. Klasių diagrama



3.5. Programos vartotojo vadovas

Darbuotojų ir duomenų failus reikia įkelti į „FileUpload“, kad nuskaityti spausti „Patvirtinti“. Norint pamatyti rezultatus spaudžiame atitinkamai „ButtonEmployees“,“ButtonAverage“ ir „ButtonExcecute“

3.6. Programos tekstas

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace LD6_Premijos2
{
    /// <summary>
    /// Darbo įnašų klasė
    /// </summary>
    public class Contributions : IComparable<Contributions>, IEquatable<Contributions>
    {
        public long PersonalID { get; set; }            //Darbuotojo asmens kodas.
        public double Contribution1 { get; set; }       //Darbo įnašas į 1 temą.
        public double Contribution2 { get; set; }       //Darbo įnašas į 2 temą.
        public double Contribution3 { get; set; }       //Darbo įnašas į 3 temą.
        public double Contribution4 { get; set; }       //Darbo įnašas į 4 temą.

        /// <summary>
        /// Tuščias konstruktorius
        /// </summary>
        public Contributions()
        {

        }

        /// <summary>
        /// Konstruktorius
        /// </summary>
        /// <param name="personalID">Asmens kodas</param>
        /// <param name="contribution1">Įnašas į pirmą temą</param>
        /// <param name="contribution2">Įnašas į antrą temą</param>
        /// <param name="contribution3">Įnašas į trečią temą</param>
        /// <param name="contribution4">Įnašas į ketvirtą temą</param>
        public Contributions(long personalID, double contribution1, double contribution2,
                             double contribution3, double contribution4)
        {
            PersonalID = personalID;
            Contribution1 = contribution1;
            Contribution2 = contribution2;
            Contribution3 = contribution3;
            Contribution4 = contribution4;
        }

        /// <summary>
        /// ToString užklojimas
        /// </summary>
        /// <returns></returns>
        public override String ToString()
        {
            return String.Format("| {0,10} | {1,10} | {2,10} | {3, 10} | {4,10} |",
                                 PersonalID, Contribution1, Contribution2, Contribution3,
                                 Contribution4);
        }
```

```
        /// <summary>
        /// CompareTo metodas
        /// </summary>
        /// <param name="other">Kiti įnašai</param>
        /// <returns></returns>
        public int CompareTo(Contributions other)
        {
            throw new NotImplementedException();
        }
        /// <summary>
        /// Equals metodas
        /// </summary>
        /// <param name="other">kiti įnašai</param>
        /// <returns></returns>
        public bool Equals(Contributions other)
        {
            throw new NotImplementedException();
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace LD6_Premijos2
{
    /// <summary>
    /// Darbuotojo duomenų klasė
    /// </summary>
    public class Employees:IComparable<Employees>,IEquatable<Employees>
    {
        public long PersonalID { get; set; }        //Asmens kodas
        public string Surname { get; set; }         //Pavardė
        public string Name { get; set; }            //Vardas
        public string BankName { get; set; }        //Banko pavadinimas
        public string BankAccountNo { get; set; }   //Banko sąskaitos numeris

        /// <summary>
        /// Tuščias konstruktorius
        /// </summary>
        public Employees()
        {

        }

        /// <summary>
        /// Konstruktorius
        /// </summary>
        /// <param name="personalID">Asmens kodas</param>
        /// <param name="surname">Pavardė</param>
        /// <param name="name">Vardas</param>
        /// <param name="bankName">Banko pavadinimas</param>
        /// <param name="bankAccountNo">Banko sąskaitos numeris</param>
        public Employees(long personalID, string surname, string name,
                         string bankName, string bankAccountNo)
        {
            PersonalID = personalID;
            Surname = surname;
            Name = name;
            BankName = bankName;
            BankAccountNo = bankAccountNo;
        }

        /// <summary>
        /// ToString užklojimas
        /// </summary>
        /// <returns></returns>
        public override String ToString()
        {
            return String.Format("| {0,10} | {1,-10} | {2,-10} | {3,-15} | {4,10} |",
                                 PersonalID, Surname, Name, BankName, BankAccountNo);
        }
        /// <summary>
        /// Palyginimo operatoriaus užklojimas
        /// </summary>
        public static bool operator <=(Employees lhs, Employees rhs)
        {
            if (lhs.Surname.CompareTo(rhs.Surname) < 0)
            {
                return true;
            }
            return false;
        }
```

```
        /// <summary>
        /// Palyginimo operatoriaus užklojimas
        /// </summary>
        public static bool operator >=(Employees lhs, Employees rhs)
        {
            if (lhs.Surname.CompareTo(rhs.Surname) > 0)
            {
                return true;
            }
            return false;
        }
        /// <summary>
        /// Palyginimo operatoriaus užklojimas
        /// </summary>
        public static bool operator == (Employees lhs, Contributions rhs)
        {
            if (rhs == null)
                return false;
            if (lhs.PersonalID == rhs.PersonalID)
                return true;
            return false;
        }
        /// <summary>
        /// Palyginimo operatoriaus užklojimas
        /// </summary>
        public static bool operator !=(Employees lhs, Contributions rhs)
        {
            if (rhs == null)
                return false;
            if (lhs.PersonalID != rhs.PersonalID)
                return true;
            return false;
        }
        /// <summary>
        /// CompareTo metodas
        /// </summary>
        /// <param name="other">Kitas darbuotojas</param>
        /// <returns></returns>
        public int CompareTo(Employees other)
        {
            if (other == null)
            {
                return 0;
            }
            return Surname.CompareTo(other.Surname);
        }
        /// <summary>
        /// Equals metodas
        /// </summary>
        /// <param name="other">kitas darbuotojas</param>
        /// <returns></returns>
        public bool Equals(Employees other)
        {
            if (other == null)
            {
                return false;
            }
            if (Surname==other.Surname)
            {
                return true;
            }
            return false;
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace LD6_Premijos2
{
    //Mazgo klasė
    public sealed class Knot<type>
where type : IComparable<type>, IEquatable<type>
    {
        public type Data { get; set; }          //Duomenys
        public Knot<type> Right { get; set; }   //Reikšmė dešinėje
        public Knot<type> Left { get; set; }    //Reikšmė kairėje

        public List<type> List
        {
            get => default(List<type>);
            set
            {
            }
        }

        /// <summary>
        /// Konstruktorius
        /// </summary>
        /// <param name="data">Duomenys</param>
        /// <param name="right">Reikšmė dešinėje</param>
        /// <param name="left">Reikšmė kairėje</param>
        public Knot(type data, Knot<type> right, Knot<type> left)
        {
            this.Data = data;
            this.Right = right;
            this.Left = left;
        }
    }
}
```

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace LD6_Premijos2
{
    /// <summary>
    /// Sąrašo klasė
    /// </summary>
    public sealed class List<type> : IEnumerable<type>
        where type : IComparable<type>, IEquatable<type>
    {
        private Knot<type> begin, end, d;

        public List()
        {
            begin = end = d = null;
        }

        /// <summary>
        /// Duomenų pridėjimas į sąrašą
        /// </summary>
        /// <param name="module">Nurodytas modulis</param>
        public void AddToRight(type data)
        {
            Knot<type> dd = new Knot<type>(data, null, end);
            if (begin != null)
                end.Right = dd;
            else
                begin = dd;
            end = dd;
        }
        public void AddToLeft(type data)
        {
            Knot<type> dd = new Knot<type>(data, begin, null);
            if (begin != null)
                begin.Left = dd;
            else
                end = dd;
            begin = dd;
        }

        /// <summary>
        /// Sąsajos metodai
        /// </summary>
        public type GetData()
        {
            if (d != null)
                return d.Data;
            return begin.Data;
        }
        public bool Is()
        {
            return d != null;
        }
        public void Begin()
        {
            d = begin;
        }
        public void End()
        {
            d = end;
        }
        public void NextRight()
        {
```

```
            if (d != null)
                d = d.Right;
        }
        public void NextLeft()
        {
            if (d != null)
                d = d.Left;
        }



        /// <summary>
        /// Rikiavimas
        /// </summary>
        public void Sort()
        {
            for (Knot<type> d1 = begin; d1 != null; d1 = d1.Right)
            {
                Knot<type> min = d1;
                for (Knot<type> d2 = d1; d2 != null; d2 = d2.Right)
                    if (d2.Data.CompareTo(min.Data) < 0)
                        min = d2;
                type st = d1.Data;
                d1.Data = min.Data;
                min.Data = st;
            }
        }

        /// <summary>
        /// IEnumerator
        /// </summary>
        public IEnumerator<type> GetEnumerator()
        {
            for (Knot<type> dd = begin; dd != null; dd = dd.Right)
            {
                yield return dd.Data;
            }
        }

        IEnumerator IEnumerable.GetEnumerator()
        {
            throw new NotImplementedException();
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.IO;

namespace LD6_Premijos2
{
    public partial class index : System.Web.UI.Page
    {
        private List<Employees> employeeList;
        private List<Contributions> contributionList;
        private double[] bounties;
        private List<Employees> swedEmp;
        private List<Employees> sebEmp;
        private List<Employees> empListLAvg;
        protected void Page_Load(object sender, EventArgs e)
        {
            if (IsPostBack)
            {
                employeeList = (List<Employees>)Session["duomenys1"];
                bounties = (double[])Session["duomenys2"];
                contributionList = (List<Contributions>)Session["duomenys3"];

            }
            else
            {
                return;
            }
        }
        /// <summary>
        /// Nuskaitymo metodas pirmam failui.
        /// </summary>
        /// <param name="file_path"></param>
        /// <param name="file_name"></param>
        /// <returns></returns>
        static List<Employees> ReadData1(string file_path,string file_name)
        {
            List<Employees> employeesList = new List<Employees>();
            string line;
            StreamReader reader = new StreamReader(file_path+file_name);
            while ((line=reader.ReadLine())!=null)
            {
                string[] values = line.Split(';');
                Employees employee = new Employees(long.Parse(values[0]), values[1],
                                                   values[2], values[3], values[4]);
                employeesList.AddToRight(employee);
            }
            reader.Close();
            return employeesList;
        }
```

```
/// <summary>
/// Nuskaitymo metodas antram failui.
/// </summary>
/// <param name="file_path"></param>
/// <param name="file_name"></param>
/// <param name="contributionList"></param>
/// <param name="bounties"></param>
static void ReadData2(string file_path, string file_name, out List<Contributions>
                      contributionList, out double[] bounties)
{
    contributionList = new List<Contributions>();
    bounties = new double[4];
    string line;
    StreamReader reader = new StreamReader(file_path+file_name);
    while (null != (line = reader.ReadLine()))
    {
        string[] values = line.Split(';');
        if (values.Count() == 4)
        {
            for (int i = 0; i < values.Count(); i++)
            {
                bounties[i] = int.Parse(values[i]);
            }
        }
        if (values.Count() != 4)
        {
            Contributions contribution = new Contributions(long.Parse(values[0]),
                                                           double.Parse(values[1]),
                                                           double.Parse(values[2]),
                                                           double.Parse(values[3]),
                                                           double.Parse(values[4]));
            contributionList.AddToRight(contribution);
        }
    }
    reader.Close();
}
/// <summary>
/// Randa premijas skirtas darbuotojams.
/// </summary>
/// <param name="contributionList">Darbo įnašų sąrašas</param>
/// <param name="bounties">Premijos skirtos temoms</param>
/// <param name="personalID">Darbuotojų asmens kodai</param>
/// <returns></returns>
static double[] FindBounties(List<Contributions> contributionList, double[] bounties,
                             long personalID)
{
    double[] empBounties = new double[4];
    for (contributionList.Begin();contributionList.Is(); contributionList.NextRight())
    {
        if (personalID == contributionList.GetData().PersonalID)
        {
            empBounties[0] = contributionList.GetData().Contribution1 * bounties[0];
            empBounties[1] = contributionList.GetData().Contribution2 * bounties[1];
            empBounties[2] = contributionList.GetData().Contribution3 * bounties[2];
            empBounties[3] = contributionList.GetData().Contribution4 * bounties[3];
        }
    }
    return empBounties;
}
```

```
/// <summary>
/// Randa vidurkį.
/// </summary>
/// <param name="digits">Skaičiai</param>
/// <returns></returns>
static double FindAvg(double[] digits)
{
    double sum = 0;
    for (int i = 0; i < digits.Count(); i++)
    {
        sum += digits[i];
    }
    return sum / digits.Count();
}
/// <summary>
/// Randa premijų, skirtas darbuotojui, sumą.
/// </summary>
/// <param name="contributionList">Darbo įnašų sąrašas</param>
/// <param name="bounties">premijos</param>
/// <param name="personalID">darbuotojų asmens kodai</param>
/// <returns></returns>
static double FindSum(List<Contributions> contributionList, double[] bounties, long
                      personalID)
{
    double sum = 0;
    double[] bounties1 = FindBounties(contributionList, bounties, personalID);
    for (int i = 0; i < bounties1.Count(); i++)
    {
        sum += bounties1[i];
    }
    return sum;
}
/// <summary>
/// Randa darbuotojus, kurie uždirbo mažiau nei vidurkis.
/// </summary>
/// <param name="employeeList"></param>
/// <param name="contributionList"></param>
/// <param name="bounties"></param>
/// <returns></returns>
static List<Employees> LAvg(List<Employees> employeeList, List<Contributions>
                            contributionList, double[] bounties)
{
    List<Employees> empListLAvg = new List<Employees>();
    double avg = FindAvg(bounties);
    for (employeeList.Begin(); employeeList.Is(); employeeList.NextRight())
    {
        double sum = FindSum(contributionList, bounties,
                             employeeList.GetData().PersonalID);

        if (sum < avg)
        {
            empListLAvg.AddToRight(employeeList.GetData());
        }
    }
    return empListLAvg;
}
```

```
/// <summary>
/// Randa nurodytos temos skirtą premiją darbuotojui.
/// </summary>
/// <param name="contributionList">darbo įnašų sarašas</param>
/// <param name="index">indeksas nurodantis temą</param>
/// <param name="bounties">premijos</param>
/// <param name="personalID">darbuotojo asmens kodas</param>
/// <returns></returns>
static double FindExBounty(List<Contributions> contributionList, int index, double[]
                           bounties, long personalID)
{
    double bounty = 0;
    for (contributionList.Begin(); contributionList.Is();
         contributionList.NextRight())
    {
        if (personalID == contributionList.GetData().PersonalID)
        {
            if (index == 1)
            {
                bounty = bounties[0] * contributionList.GetData().Contribution1;
            }
            if (index == 2)
            {
                bounty = bounties[1] * contributionList.GetData().Contribution2;
            }
            if (index == 3)
            {
                bounty = bounties[2] * contributionList.GetData().Contribution3;
            }
            if (index == 4)
            {
                bounty = bounties[3] * contributionList.GetData().Contribution4;
            }
        }
    }
    return bounty;
}
/// <summary>
/// Atrenka darbuotojus kurie turi nurodytą banką.
/// </summary>
/// <param name="bankName">banko pavadinimas</param>
/// <param name="employeeList">darbuotojų sąrašas</param>
/// <returns></returns>
static List<Employees> DBank(string bankName, List<Employees> employeeList)
{
    List<Employees> employees = new List<Employees>();
    for (employeeList.Begin(); employeeList.Is(); employeeList.NextRight())
    {
        if (employeeList.GetData().BankName.Equals(bankName))
        {
            employees.AddToRight(employeeList.GetData());
        }
    }
    return employees;
}
/// <summary>
/// Spausdina darbuotojų sarašą į failą.
/// </summary>
/// <param name="employees">darbuotojų sąrašas</param>
static void empListWriter(List<Employees> employees, StreamWriter writer)
{
    for (employees.Begin(); employees.Is(); employees.NextRight())
    {
        writer.WriteLine(employees.GetData().ToString());
    }
}
```

```
        /// <summary>
        /// Spausdina darbuotojų įnašus į darbus.
        /// </summary>
        /// <param name="contributions">darbo įnašai</param>
        static void contrListWriter(List<Contributions> contributions, StreamWriter writer)
        {
            for (contributions.Begin(); contributions.Is(); contributions.NextRight())
            {
                writer.WriteLine(contributions.GetData().ToString());
            }
        }
        /// <summary>
        /// spausdina į failą
        /// </summary>
        /// <param name="employeeList">darbuotojų sarašas</param>
        /// <param name="contributionList">darbo įnašų sarašas</param>
        /// <param name="bounties">premijos</param>
        /// <param name="swedEmp">Darbuotojai turintys swedbank</param>
        /// <param name="sebEmp">Darbuotojai turintys sebbank</param>
        static void WriteToFile(List<Employees>employeeList, List<Contributions>
                                contributionList,double[] bounties, List<Employees>
                                swedEmp,List<Employees>sebEmp)
        {
            StreamWriter writer = new
StreamWriter("C:\\Users\\dovyd\\Desktop\\LD6_Premijos2\\LD6_Premijos2\\Results.txt");
            writer.WriteLine("Duomenys iš U6a.txt:");
            writer.WriteLine(" ----------------------------------------------------------------");
            empListWriter(employeeList, writer);
            writer.WriteLine("Duomenys iš U6b.txt:");
            writer.WriteLine(" ----------------------------------------------------------------");
            for (int i = 0; i < 4; i++)
            {
                writer.Write(bounties[i] + " ");
            }
            writer.WriteLine();
            contrListWriter(contributionList, writer);
            writer.WriteLine("Swedbank pavedimai");
            writer.WriteLine(" ----------------------------------------------------------------");
            empListWriter(swedEmp, writer);
            writer.WriteLine("SEB bankas pavedimai");
            writer.WriteLine(" ----------------------------------------------------------------");
            empListWriter(sebEmp, writer);
            writer.Close();
        }
```

```
        /// <summary>
        /// Paspaudus "ButtonReadData" nuskaitomi failai.
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        protected void ButtonReadData_Click(object sender, EventArgs e)
        {
            string file_path = "C:\\Users\\dovyd\\Desktop\\LD6_Premijos2\\LD6_Premijos2\\";
            string file_name1 = FileUpload1.PostedFile.FileName;
            string file_name2 = FileUpload2.PostedFile.FileName;
            if (!File.Exists(Server.MapPath(file_name1)) ||
                !File.Exists(Server.MapPath(file_name2)))
            {
                ButtonReadData.Text = String.Format("Nerasti failai!");
                return;
            }
            ButtonReadData.Text = String.Format("Patvirtinta");
            ButtonReadData.Enabled = false;
            FileUpload1.Enabled = FileUpload2.Enabled = false;

            Label5.Text = MapPath(file_name1);
            Label6.Text = MapPath(file_name2);
            employeeList = ReadData1(file_path,file_name1);
            ReadData2(file_path,file_name2, out contributionList, out bounties);
            swedEmp = DBank("Swedbank", employeeList);
            sebEmp = DBank("SEB bankas", employeeList);
            Session["duomenys1"] = employeeList;
            Session["duomenys2"] = bounties;
            Session["duomenys3"] = contributionList;
            employeeList.Sort();
            swedEmp.Sort();
            sebEmp.Sort();
            LabelEmployees.Visible = ButtonEmployees.Visible = TextBoxEmployees.Visible =
                                     LabelLAverage.Visible = ButtonAverage.Visible =
                                     TextBoxLAverage.Visible = LabelExBounties.Visible =
                                     TextBoxIndex.Visible = ButtonExcecute.Visible =
                                     TextBoxExTheme.Visible = true;
            WriteToFile(employeeList, contributionList, bounties,  swedEmp, sebEmp);

        }
        /// <summary>
        /// Paspaudus "ButtonEmployees" spausdinami darbuotojai su uždarbiais.
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        protected void ButtonEmployees_Click1(object sender, EventArgs e)
        {
            TextBoxEmployees.Text = String.Format("| {0,-10}| {1,-10} | {2,-10} | {3,-15} |
{4,-18} |" + "\n", "Asmens kodas", "Pavardė", "Vardas", "Banko pav.", "Sąskaitos No.");
            for (employeeList.Begin(); employeeList.Is(); employeeList.NextRight())
            {

                double[] empBounties = empBounties = FindBounties(contributionList, bounties,
                                                        employeeList.GetData().PersonalID);
                TextBoxEmployees.Text += employeeList.GetData().ToString();
                TextBoxEmployees.Text += " Premijų dydžiai: ";
                for (int i = 0; i < empBounties.Count(); i++)
                {
                    TextBoxEmployees.Text += empBounties[i] + " ";
                }
                TextBoxEmployees.Text += "Iš viso: " + FindSum(contributionList, bounties,
                                                 employeeList.GetData().PersonalID) + "\n";
            }
        }
```

```
        /// <summary>
        /// Paspaudus "ButtonAverage" spausdina darbuotojus kurie uždirbo mažiau nei vidurkis
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        protected void ButtonAverage_Click1(object sender, EventArgs e)
        {
            empListLAvg = LAvg(employeeList, contributionList, bounties);
            TextBoxLAverage.Text = String.Format("| {0,-10}| {1,-10} | {2,-10} | {3,-15} |
                                                 {4,-18} |" + "\n", "Asmens kodas", "Pavardė",
                                                 "Vardas", "Banko pav.", "Sąskaitos No.");
            for (empListLAvg.Begin(); empListLAvg.Is(); empListLAvg.NextRight())
            {
                TextBoxLAverage.Text += empListLAvg.GetData().ToString() + "\n";
            }
        }
        /// <summary>
        /// Paspaudus "ButtonExcecute" randa darbuotojų uždarbius nurodytoje temoje.
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        protected void ButtonExcecute_Click(object sender, EventArgs e)
        {
            int index;
            if (IsPostBack)
            {
                TextBoxExTheme.Text = "";
                int.TryParse(TextBoxIndex.Text, out index);
                if (index >= 1 && index <= 4)
                {
                    LabelError.Visible = false;
                    TextBoxExTheme.Text = String.Format("| {0,-10}| {1,-10} | {2,-10} | {3,-
                                                       15} | {4,-18} |" + "\n", "Asmens
                                                       kodas", "Pavardė", "Vardas", "Banko
                                                       pav.", "Sąskaitos No.");
                    for (employeeList.Begin(); employeeList.Is(); employeeList.NextRight())
                    {
                        double[] empBounties = empBounties = FindBounties(contributionList,
                                                       bounties,
                                                       employeeList.GetData().PersonalID);
                        TextBoxExTheme.Text += employeeList.GetData().ToString();
                        TextBoxExTheme.Text += "Nurodytos temos premijos dydis: " +
FindExBounty(contributionList, index, bounties, employeeList.GetData().PersonalID) + "\n";
                    }
                }
                else
                {
                    LabelError.Visible = true;
                    LabelError.Text = "Įveskite tinkamą reikšmę.";
                }
            }
        }
    }
}
```

3.7. Pradiniai duomenys ir rezultatai

```
Pradiniai duomenys:
U6a.txt:
37108316658;Mykolaitis;Mykolas;Swedbank;LT1234654989877125
36912014457;Petraitis;Petras;SEB bankas;LT4789247424857154
37306129942;Pavardenis;Vardenis;SEB bankas;LT4589325896614587
38203134587;Juozaitis;Juozas;Swedbank;LT4569123458755235


U6b.txt:
10000;20000;30000;40000
37108316658;0.18;0.33;0;0.47
36912014457;0.32;0.33;0.20;0.18
37306129942;0.25;0.15;0.35;0.12
38203134587;0.25;0.19;0.45;0.23

Rezultatai:
Duomenys iš U6a.txt:
 ------------------------------------------------------------------
| 38203134587 | Juozaitis  | Juozas     | Swedbank        | LT4569123458755235 |
| 37108316658 | Mykolaitis | Mykolas    | Swedbank        | LT1234654989877125 |
| 37306129942 | Pavardenis | Vardenis   | SEB bankas      | LT4589325896614587 |
| 36912014457 | Petraitis  | Petras     | SEB bankas      | LT4789247424857154 |
Duomenys iš U6b.txt:
 ------------------------------------------------------------------
10000 20000 30000 40000
| 37108316658 |       0.18 |       0.33 |          0 |       0.47 |
| 36912014457 |       0.32 |       0.33 |        0.2 |       0.18 |
| 37306129942 |       0.25 |       0.15 |       0.35 |       0.12 |
| 38203134587 |       0.25 |       0.19 |       0.45 |       0.23 |
Swedbank pavedimai
 ------------------------------------------------------------------
| 38203134587 | Juozaitis  | Juozas     | Swedbank        | LT4569123458755235 |
| 37108316658 | Mykolaitis | Mykolas    | Swedbank        | LT1234654989877125 |
SEB bankas pavedimai
 ------------------------------------------------------------------
| 37306129942 | Pavardenis | Vardenis   | SEB bankas      | LT4589325896614587 |
| 36912014457 | Petraitis  | Petras     | SEB bankas      | LT4789247424857154 |
```

3.8. Dėstytojo pastabos

- Main metode esančius veiksmus galima iškelti į metodus.
- Tekstiniam faile spausdinami ne visi rezultatai.

Įvertinimas:

- Programos kodas: 5
- Testas: 0
- Ataskaita -
- Galutinis -

4. Kolekcijos ir išimčių valdymas (L4)

4.1. Darbo užduotis

LDD_6. Moduliai. Pirmoje failo eilutėje nurodytas fakulteto pavadinimas. Tekstiniame faile duota informacija apie studentų pasirenkamus modulius: modulio pavadinimas, studento pavardė, vardas, grupė. Kitame faile yra informacija apie modulius: modulio pavadinimas, atsakingo dėstytojo pavardė, vardas, kreditų kiekis. Suskaičiuoti kiekvieno dėstytojo darbo krūvį, jei modulio kreditų skaičius dauginamas iš jį pasirinkusių studentų skaičiaus. Atspausdinti nurodyto dėstytojo (įvedama klaviatūra) kiekvieno modulio studentų sąrašus pagal grupes ir studentų pavardes.

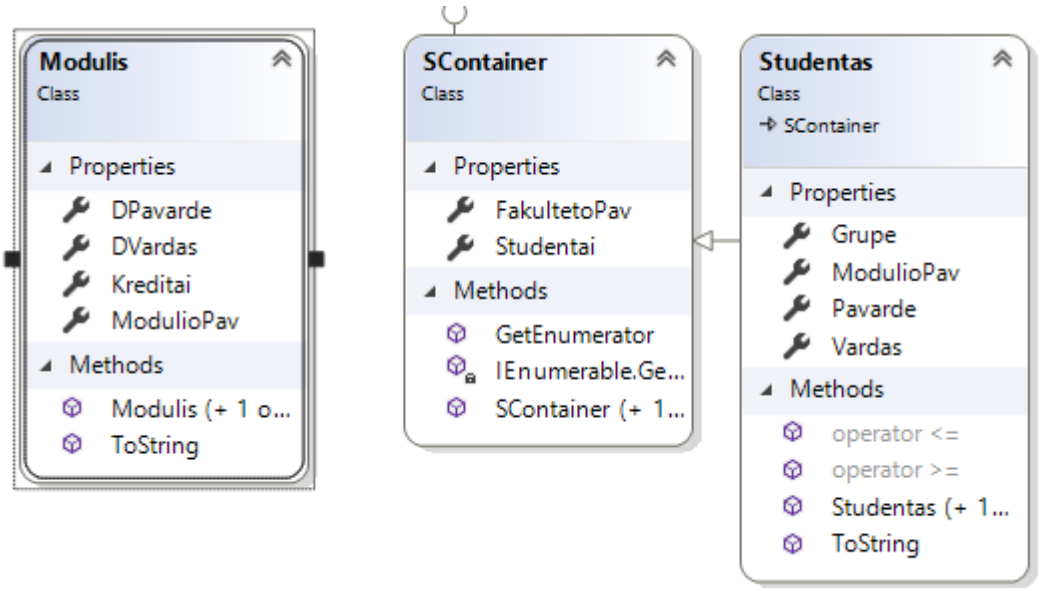
4.2. Grafinės vartotojo sąsajos schema

4.3. Sąsajoje panaudotų komponentų keičiamos savybės

Komponentas	Savybė	Reikšmė
TextBoxResults1	Height	200px
TextBoxResults1	Width	1000px
TextBoxResults1	TextMode	MultiLine
TextBoxResults2	Height	200px
TextBoxResults2	Width	1000px
TextBoxResults2	TextMode	MultiLine
TextBoxInput	Height	40px
TextBoxInput	Width	40px
TextBoxInput	TextMode	MultiLine
Button_Execute	Text	Vykdyti

Button_Execute	Height	42px
Button_Execute	Width	100px
Button_Input	Text	Įvesti
Button_Input	Heigt	42px
Button_Input	Width	100px
Label_Error	Visible	False
Label_Error	Height	45px
Label_Error	ForeColor	Red
Label_Error	Text	*

4.4. Klasių diagrama



4.5. Programos vartotojo vadovas

Norint pamatyti modulius ir už juos atsakingus dėstytojus spaudžiame mygtuką „Vykdyti“, norint pamatyti studentus, kur turi pasirinkę modulį pas nurodytą dėstytoja, įvedame dėstytojo skaitliuką į laukelį „TextBoxInput“ ir spaudžiame mygtuką „Įvesti“.

4.6. Programos tekstas

```
using System;

namespace LD4
{
    /// <summary>
    /// Klasė, dėst. modulio duomenims saugoti.
    /// </summary>
    public class Modulis
    {
        public string ModulioPav { get; set; }  //Modulio pavadinimas
        public string DPavarde { get; set; }    //Dėstytojo pavardė
        public string DVardas { get; set; }     //Dėstytojo vardas
        public int Kreditai { get; set; }       //Kreditų skaičius už modulį

        /// <summary>
        /// Tuščias konstruktorius
        /// </summary>
```

```
        public Modulis()
        {
        }

        /// <summary>
        /// Konstruktorius
        /// </summary>
        /// <param name="modulioPav"></param>
        /// <param name="dPavarde"></param>
        /// <param name="dVardas"></param>
        /// <param name="kreditai"></param>
        public Modulis(string modulioPav, string dPavarde, string dVardas, int kreditai)
        {
            ModulioPav = modulioPav;
            DPavarde = dPavarde;
            DVardas = dVardas;
            Kreditai = kreditai;
        }
        /// <summary>
        /// ToString() metodo užklojimas
        /// </summary>
        /// <returns></returns>
        public override string ToString()
        {
            return String.Format("|{0,-30}|{1,-20}|{2,-20}|{3,20}|", ModulioPav, DPavarde,
DVardas, Kreditai);
        }
    }
}
```

```
using System;
using System.Collections;
using System.Collections.Generic;

namespace LD4
{
    /// <summary>
    /// Klasė, fakulteto pavadinimui ir studentų duomenims saugoti.
    /// </summary>
    public class SContainer:IEnumerable<Studentas>
    {
        public string FakultetoPav { get; set; }            // Fakulteto pavadinimas
        public List<Studentas> Studentai { get; set; }      // Studentų sąrašas

        /// <summary>
        /// Tuščias konstruktorius
        /// </summary>
        public SContainer() { }

        /// <summary>
        /// Konstruktorius
        /// </summary>
        public SContainer(List<Studentas> studentai,string fakultetoPav)
        {
            FakultetoPav = fakultetoPav;
            Studentai = studentai;
        }

        //-------------------------------------
        public IEnumerator<Studentas> GetEnumerator()
        {
            for (int i = 0; i < Studentai.Count; i++)
                yield return Studentai[i];
        }

        //-------------------------------------
        IEnumerator IEnumerable.GetEnumerator()
        {
            throw new NotImplementedException();
        }
    }
}


using System;

namespace LD4
{
    /// <summary>
    /// Klasė, studentų duomenims saugoti.
    /// </summary>
    public class Studentas : SContainer
    {
        public string ModulioPav { get; set; }     //Modulio pavadinimas
        public string Pavarde { get; set; }        //Studento pavardė
        public string Vardas { get; set; }         //Studento vardas
        public string Grupe { get; set; }          //Studento grupė

        /// <summary>
        /// Tuščias konstruktorius
        /// </summary>
        public Studentas()
        {
        }

        /// <summary>
        /// Konstruktorius
```

```
        /// </summary>
        /// <param name="modulioPav"></param>
        /// <param name="pavarde"></param>
        /// <param name="vardas"></param>
        /// <param name="grupe"></param>
        public Studentas(string modulioPav, string pavarde, string vardas, string grupe)
        {
            ModulioPav = modulioPav;
            Pavarde = pavarde;
            Vardas = vardas;
            Grupe = grupe;
        }

        /// <summary>
        /// Palyginimo operatoriaus užklojimas (1)
        /// </summary>
        /// <param name="lhs"></param>
        /// <param name="rhs"></param>
        /// <returns></returns>
        public static bool operator <=(Studentas lhs, Studentas rhs)
        {
            if (lhs.Grupe.CompareTo(rhs.Grupe) < 0)
            {
                return true;
            }
            else if (lhs.Grupe.CompareTo(rhs.Grupe) == 0)
            {
                if (lhs.Pavarde.CompareTo(rhs.Pavarde) <= 0)
                {
                    return true;
                }
                return false;
            }
            else
                return false;
        }
        /// <summary>
        /// Palyginimo operatoriaus užklojimas (2)
        /// </summary>
        /// <param name="lhs"></param>
        /// <param name="rhs"></param>
        /// <returns></returns>
        public static bool operator >=(Studentas lhs, Studentas rhs)
        {
            if (lhs.Grupe.CompareTo(rhs.Grupe) > 0)
            {
                return true;
            }
            else if (lhs.Grupe.CompareTo(rhs.Grupe) == 0)
            {
                if (lhs.Pavarde.CompareTo(rhs.Pavarde) >= 0)
                {
                    return true;
                }
                return false;
            }
            else
                return false;
        }

        /// <summary>
        /// ToString() metodo užklojimas
        /// </summary>
        /// <returns></returns>
        public override string ToString()
        {
            return String.Format("|{0,-30}|{1,-20}|{2,-20}|{3,-20}|", ModulioPav, Pavarde,
Vardas, Grupe);
```

```
            }
        }
    }
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.IO;

namespace LD4
{
    public partial class index : System.Web.UI.Page
    {
        public const string FailuVieta = "~/App_Data/Studentai/";
        public const string FailoVieta = "~/App_Data/U6a.txt";
        public const string RezultatoFailas = "Rezultatai.txt";
        protected void Page_Load(object sender, EventArgs e)
        {
            // nieko nedaro
        }

        /// <summary>
        /// Nuskaito fakulteto pavadinimą ir studentus.
        /// </summary>
        /// <param name="directory"></param>
        /// <returns></returns>
        public SContainer ReadFromFile1 (string directory)
        {
            SContainer inventoryData = new SContainer();
            try
            {
                foreach (string currentFile in Directory.GetFiles(Server.MapPath(FailuVieta)))
                {
                    try
                    {
                        List<Studentas> studentai = new List<Studentas>();
                        string fakultetas = null;
                        string[] lines = File.ReadAllLines(currentFile);
                        for (int y = 0; y < lines.Count(); y++)
                        {
                            string[] values = lines[y].Split(';');
                            if (values.Count()!=1)
                            {
                                Studentas studentas = new Studentas(values[0], values[1],
                                                                    values[2], values[3]);
                                studentai.Add(studentas);
                            }
                            else
                            {
                                fakultetas = lines[y];
                            }
                        }

                        inventoryData = new SContainer(studentai,fakultetas);
                    }
                    catch (Exception)
                    {

                    }
                }
            }
            catch (FormatException)
            {
                Console.WriteLine("Blogas duomenų failų formatas");
            }
            return inventoryData;
        }

        /// <summary>
        /// Nuskaito modulius ir destytojus kruie atsakingi
        /// </summary>
```

```
        /// <param name="file"></param>
        /// <returns></returns>
        public List<Modulis> ReadFromFile2(string file)
        {
            List<Modulis> Moduliai = new List<Modulis>();
            try
            {
                string[] lines = File.ReadAllLines(file);

                foreach (string value in lines.Skip(0))
                {
                    string[] values = value.Split(';');
                    Modulis modulis = new Modulis(values[0], values[1], values[2],
int.Parse(values[3]));
                    Moduliai.Add(modulis);
                }
            }

            catch (FormatException)
            {
                Console.WriteLine("");
                throw;
            }
            return Moduliai;
        }

        /// <summary>
        /// Randa dėstytojo krūvį
        /// </summary>
        /// <param name="modulis"></param>
        /// <param name="studentai"></param>
        /// <param name="index"></param>
        /// <returns></returns>
        static int Kruvis(List<Modulis> modulis, SContainer studentai,int index)
        {
            int sum = 0;
            for (int i = 0; i < studentai.Count(); i++)
            {
                if (modulis[index].ModulioPav==studentai.Studentai[i].ModulioPav)
                {
                    sum++;
                }
            }
            return sum * modulis[index].Kreditai;
        }

        /// <summary>
        /// Sudeda dėstytojus su jų krūviais į sąrašą.
        /// </summary>
        /// <param name="modulis"></param>
        /// <param name="studentai"></param>
        /// <returns></returns>
        static List<Modulis> DestDarboKruvis(List<Modulis> modulis, SContainer studentai)
        {
            List<Modulis> destytojai = new List<Modulis>();
            for (int i = 0; i < modulis.Count(); i++)
            {
                destytojai.Add(modulis[i]);
                destytojai[i].Kreditai = Kruvis(modulis, studentai, i);
            }
            return destytojai;
        }

        /// <summary>
        /// Spausdina modulius ir dėstytojus.
        /// </summary>
        /// <param name="modulis"></param>
        public void SpausdintiModulius(List<Modulis> modulis)
```

```
        {
            TextBoxResults1.Text = TextBoxResults1.Text + "+---------------------------------
-------------------------------------------------------+\n";
            TextBoxResults1.Text = TextBoxResults1.Text +
+String.Format("|{0,-30}|{1,-20}|{2,-20}|{3,-20}|", "Modulio pavadinimas", "Dėstytojo
              pavardė",Dėstytojo vardas", "Krūvis")+"\n";
            TextBoxResults1.Text = TextBoxResults1.Text + "+---------------------------------
-------------------------------------------------------+\n";
            for (int i = 0; i < modulis.Count(); i++)
            {
                TextBoxResults1.Text = TextBoxResults1.Text + modulis[i].ToString() + "\n";
            }
            TextBoxResults1.Text = TextBoxResults1.Text + "+---------------------------------
-------------------------------------------------------+\n";
        }

        /// <summary>
        /// spausdina studentus
        /// </summary>
        /// <param name="studentai"></param>
        public void SpausdintiStudentus(List<Studentas> studentai)
        {
            TextBoxResults2.Text = TextBoxResults2.Text + "+---------------------------------
-------------------------------------------------------+\n";
            TextBoxResults2.Text = TextBoxResults2.Text +
+ String.Format("|{0,-30}|{1,-20}|{2,-20}|{3,-20}|", "Modulio pavadinimas", "Studento
              pavardė", "Studento vardas", "Grupė") + "\n";
            TextBoxResults2.Text = TextBoxResults2.Text + "+---------------------------------
-------------------------------------------------------+\n";
            for (int i = 0; i < studentai.Count(); i++)
            {
                TextBoxResults2.Text = TextBoxResults2.Text + studentai[i].ToString() + "\n";
            }
            TextBoxResults2.Text = TextBoxResults2.Text + "+---------------------------------
-------------------------------------------------------+\n";
        }

        /// <summary>
        /// Išrenka studentus kurie priklauso nurodytam dėstytojui.
        /// </summary>
        /// <param name="modulis"></param>
        /// <param name="studentai"></param>
        /// <param name="index"></param>
        /// <returns></returns>
        static List<Studentas> IsrinktiStudentus(List<Modulis> modulis, List<Studentas>
                                                 studentai, int index)
        {
            List<Studentas> atrinktiStudentai = new List<Studentas>();
            index = index - 1;
            for (int i = 0; i < studentai.Count(); i++)
            {
                if (modulis[index].ModulioPav==studentai[i].ModulioPav)
                {
                    atrinktiStudentai.Add(studentai[i]);
                }
            }
            return atrinktiStudentai;
        }

        /// <summary>
        /// Rikiuoja studentus pagal grupę ir pavardes
        /// </summary>
        /// <param name="studentai"></param>
        static void Rikiuoti(List<Studentas> studentai)
        {
            for (int j = 0; j < studentai.Count()-1; j++)
            {
                for (int i = 1; i < studentai.Count(); i++)
```

```
                {
                    if (studentai[j]>=studentai[i])
                    {
                        Studentas laikinas = studentai[j];
                        studentai[j] = studentai[i];
                        studentai[i] = laikinas;
                    }
                }
            }
        }
        public void SpausdintiIFaila(SContainer studentai, List<Modulis> moduliai,
List<Modulis> destKruviai,string failas)
        {
            using (StreamWriter writer = new StreamWriter(Server.MapPath(failas)))
            {
                writer.WriteLine("+-------------------------------------DUOMENYS-------------
----------------------------------+)");
                writer.WriteLine("|{0,-30}|{1,-20}|{2,-20}|{3,-20}|", "Modulio pavadinimas",
                                 "Dėstytojo pavardė", "Dėstytojo vardas", "Kreditai už
                                 modulį");
                writer.WriteLine("+-----------------------------------------------------------
----------------------------------+");
                for (int i = 0; i < moduliai.Count(); i++)
                {
                    writer.WriteLine(moduliai[i].ToString());
                }
                writer.WriteLine("+-----------------------------------------------------------
----------------------------------+");
                writer.WriteLine(studentai.FakultetoPav);
                writer.WriteLine("|{0,-30}|{1,-20}|{2,-20}|{3,-20}|", "Modulio pavadinimas",
                                 "Studento pavardė", "Studento vardas", "Grupė");
                for (int i = 0; i < studentai.Studentai.Count(); i++)
                {
                    writer.WriteLine(studentai.Studentai[i].ToString());
                }
                writer.WriteLine("+-------------------------------------REZULTATAI-----------
                                 ----------------------------------+)");
                for (int i = 0; i < destKruviai.Count(); i++)
                {
                    writer.WriteLine(destKruviai[i].ToString());
                }
            }
        }

        /// <summary>
        /// Atlieka veiksmus paspaudus mygtuką "Execute"
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        protected void Button_Execute_Click(object sender, EventArgs e)
        {
            try
            {
                TextBoxResults1.Text = null;
                List<Modulis> Destytojai = ReadFromFile2(Server.MapPath(FailoVieta));
                SContainer Studentai = ReadFromFile1(Server.MapPath(FailuVieta));
                List<Modulis> DestytojuKruvis = DestDarboKruvis(Destytojai, Studentai);
                SpausdintiModulius(DestytojuKruvis);
                SpausdintiIFaila(Studentai, Destytojai, DestytojuKruvis,RezultatoFailas);
            }
            catch (Exception)
            {

            }
        }

        /// <summary>
        /// Atlieka veiksmus paspaudus mygtuką "Lecturer".
```

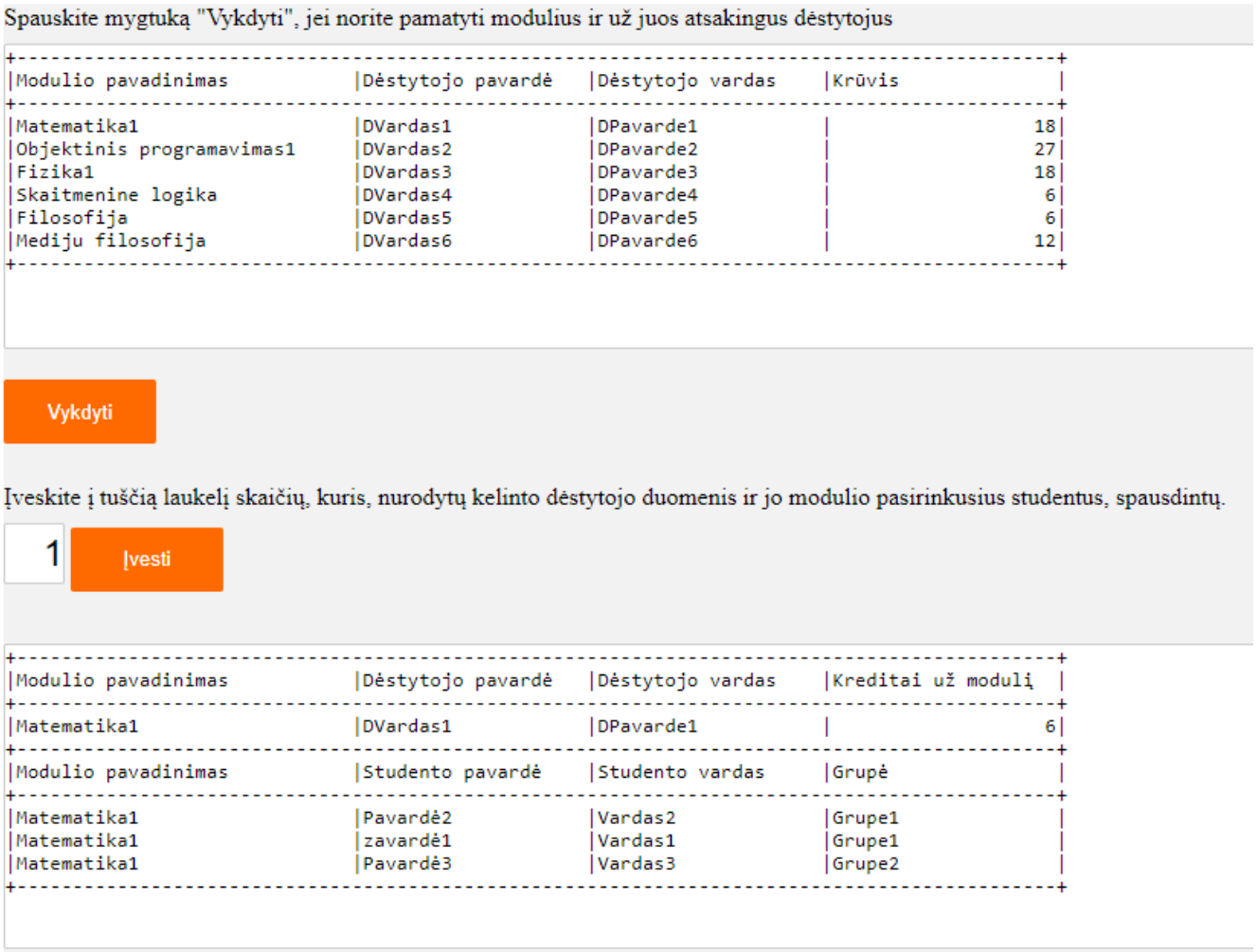
```
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        protected void Button_Lecturer_Click(object sender, EventArgs e)
        {
            try
            {
                TextBoxResults2.Text = null;
                List<Modulis> Destytojai = ReadFromFile2(Server.MapPath(FailoVieta));
                SContainer Studentai = ReadFromFile1(Server.MapPath(FailuVieta));
                int index = int.Parse(TextBoxInput.Text);
                List<Studentas> atrinktiStudentai = IsrinktiStudentus(Destytojai,
Studentai.Studentai, index);
                Rikiuoti(atrinktiStudentai);
                TextBoxResults2.Text = TextBoxResults2.Text + "+-----------------------------
-------------------------------------------------------------+\n";
                TextBoxResults2.Text = TextBoxResults2.Text +
+String.Format("|{0,-30}|{1,-            20}|{2,-20}|{3,-20}|", "Modulio pavadinimas",
             "Dėstytojo pavardė", "Dėstytojo vardas", "Kreditai už modulį") + "\n";
                TextBoxResults2.Text = TextBoxResults2.Text + "+-----------------------------
-------------------------------------------------------------+\n";
                TextBoxResults2.Text = TextBoxResults2.Text + Destytojai[index - 1].ToString()
                                       + "\n";
                SpausdintiStudentus(atrinktiStudentai);
                LabelError.Visible = false;
            }
            catch (Exception)
            {
                LabelError.Visible = true;
                LabelError.Text = "Neteisinga reikšmė";
            }
        }
    }
}
```

4.7. Pradiniai duomenys ir rezultatai

```
Pradiniai duomenys:


U6a.txt:

Matematika1;DVardas1;DPavarde1;6
Objektinis programavimas1;DVardas2;DPavarde2;9
Fizika1;DVardas3;DPavarde3;6
Skaitmenine logika;DVardas4;DPavarde4;6
Filosofija;DVardas5;DPavarde5;6
Mediju filosofija;DVardas6;DPavarde6;6

U6b.txt:

Informatikos fakultetas
Matematika1;zavardė1;Vardas1;Grupe1
Fizika1;qavardė1;Vardas1;Grupe1
Mediju filosofija;Pavardė1;Vardas1;Grupe1
Objektinis programavimas1;Pavardė1;Vardas1;Grupe1
Skaitmeninė logika;Pavardė1;Vardas1;Grupe1
Matematika1;Pavardė2;Vardas2;Grupe1
Fizika1;Pavardė2;Vardas2;Grupe1
Filosofija;Pavardė2;Vardas2;Grupe1
Objektinis programavimas1;Pavardė2;Vardas2;Grupe1
Skaitmenine logika;Pavardė2;Vardas2;Grupe1
Matematika1;Pavardė3;Vardas3;Grupe2
Fizika1;Pavardė3;Vardas3;Grupe2
Mediju filosofija;Pavardė3;Vardas3;Grupe2
```

```
Objektinis programavimas1;Pavardė3;Vardas3;Grupe2
Skaitmeninė logika;Pavardė3;Vardas3;Grupe2
```

Rezultatai:



Rezultatai.txt:

```
+--------------------------------------DUOMENYS-----------------------------------
------------+)
|Modulio pavadinimas            |Dėstytojo pavardė    |Dėstytojo vardas     |Kreditai
už modulį   |
+--------------------------------------------------------------------------------
------------+
|Matematika1                    |DVardas1             |DPavarde1            |
18|
|Objektinis programavimas1      |DVardas2             |DPavarde2            |
27|
|Fizika1                        |DVardas3             |DPavarde3            |
18|
|Skaitmenine logika             |DVardas4             |DPavarde4            |
6|
|Filosofija                     |DVardas5             |DPavarde5            |
6|
|Mediju filosofija              |DVardas6             |DPavarde6            |
12|
+--------------------------------------------------------------------------------
------------+
Informatikos fakultetas
|Modulio pavadinimas            |Studento pavardė     |Studento vardas      |Grupė
|
|Matematika1                    |zavardė1             |Vardas1              |Grupe1
|
|Fizika1                        |qavardė1             |Vardas1              |Grupe1
|
```

```
|Mediju filosofija             |Pavardė1            |Vardas1             |Grupe1
|
|Objektinis programavimas1     |Pavardė1            |Vardas1             |Grupe1
|
|Skaitmeninė logika            |Pavardė1            |Vardas1             |Grupe1
|
|Matematika1                   |Pavardė2            |Vardas2             |Grupe1
|
|Fizika1                       |Pavardė2            |Vardas2             |Grupe1
|
|Filosofija                    |Pavardė2            |Vardas2             |Grupe1
|
|Objektinis programavimas1     |Pavardė2            |Vardas2             |Grupe1
|
|Skaitmenine logika            |Pavardė2            |Vardas2             |Grupe1
|
|Matematika1                   |Pavardė3            |Vardas3             |Grupe2
|
|Fizika1                       |Pavardė3            |Vardas3             |Grupe2
|
|Mediju filosofija             |Pavardė3            |Vardas3             |Grupe2
|
|Objektinis programavimas1     |Pavardė3            |Vardas3             |Grupe2
|
|Skaitmeninė logika            |Pavardė3            |Vardas3             |Grupe2
|
+-------------------------------------REZULTATAI---------------------------------
------------+)
|Matematika1                   |DVardas1            |DPavarde1           |
18|
|Objektinis programavimas1     |DVardas2            |DPavarde2           |
27|
|Fizika1                       |DVardas3            |DPavarde3           |
18|
|Skaitmenine logika            |DVardas4            |DPavarde4           |
6|
|Filosofija                    |DVardas5            |DPavarde5           |
6|
|Mediju filosofija             |DVardas6            |DPavarde6           |
12|
```

4.8. Dėstytojo pastabos

- Nesilaikyta nurodymų, kurie yra modulio apraše.
- Spausdinimo metodą galima išskaidyti į kelis metodus.
- Valdomos išimtįs negali būti paliktos tuščios.

Testas:0
Kodas:4
Ataskaita: -
Bendras: -

5. Deklaratyvusis programavimas (L5) (Neatlikta, paimta dovana)

5.1. Darbo užduotis

LDD_6. Moduliai. Pirmoje failo eilutėje nurodytas fakulteto pavadinimas. Tekstiniame faile duota informacija apie studentų pasirenkamus modulius: modulio pavadinimas, studento pavardė, vardas, grupė. Kitame faile yra informacija apie modulius: modulio pavadinimas, atsakingo dėstytojo pavardė, vardas, kreditų kiekis. Suskaičiuoti kiekvieno dėstytojo darbo krūvį, jei modulio kreditų skaičius dauginamas iš jį pasirinkusių studentų skaičiaus. Atspausdinti nurodyto dėstytojo (įvedama klaviatūra) kiekvieno modulio studentų sąrašus pagal grupes ir studentų pavardes.

5.2. Grafinės vartotojo sąsajos schema

-

5.3. Sąsajoje panaudotų komponentų keičiamos savybės

Komponentas	Savybė	Reikšmė
-	-	-
-	-	-
-	-	-

5.4. Klasių diagrama

-

5.5. Programos vartotojo vadovas

-

5.6. Programos tekstas

–

5.7. Pradiniai duomenys ir rezultatai

–

5.8. Dėstytojo pastabos

- Paimta dovana

  Bendras: 5