

Vilniaus universiteto

Fizikos fakulteto

Kompiuterinės fizikos ir modeliavimo studijų programos studentas

Mantvydas Lissauskas

E3 laboratorinio darbo sprendimo programa

Projektas

Vilnius 2021

Turinys

Turinys.....	2
Pratarmė.....	4
Programa.....	4
Programos schema.....	5
Naudojamų technologijų sąrašas	6
Technologijos	7
1. Bitinės matematinės operacijos	7
3. nuorodos priskyrimas	7
4. Blokai try, catch, finally	8
5. Ciklai	8
6. Sąlygos	8
7. Klasė, objekto sukūrimas.....	9
8. Konstruktorius	9
9. Metodas klasėje	9
10. Duomuo	10
11. Metodo iškvičimas.....	10
12. Operatoriai this ir super	10
13. Išimties situacijos	11
14. Panaudojimas throws ir throw	11
15. Anoniminė įdėtinė klasė ir pagal ją objektai viduje klasės	11
19. Poklasė	12
20. Viršklasės duomenų ir metodų panaudojimas	12
24. Konstruktorius super su ir be argumentu	12
25. Išimties situacijos: klasės ir poklasės.....	13
29. Metodo perrašymas(overriding)	13
30. Abstrakti klasė	14
36. Objektų derinimas pagal sąsajas(interface).....	15
37 Daugybiniškumas (generics): klasės sukūrimas ir opbjektai pagal ją.	15
43. Masyvas: primityviems duomenims	16

50. Operatorius static: duomuo, metodas.....	16
56 Teisės ir jų panaudojimas: private	17
58. Teisės ir jų panaudojimas: protected.....	17
59. Teisės ir jų panaudojimas: public	17
64. Srauto panaudojimas pagal klasę <i>Stream</i> : ciklas kiekvienam	18
67. Funkcinė nuoroda Lambda technologijoje.....	18
69. GETSET technologija.....	18
70 Aiškinimo (annotation) parametrų panaudojimas	19
72. I/O failų nuskaitymas ir įrašymas	19
73. I/O nuskaitymas iš išorinės nuorodos	20
83. <i>Set</i> duomenų saugojimas, panaudojant pakaitos simboliu (Kolekcijos)	21
84. <i>Map</i> tipo duomenų saugojimas, panaudojant pakaitos simbolius (Kolekcijos).....	21
86: toString technologija	22
Svarbu paminėti.....	22
Literatūra	22

Pratarmė

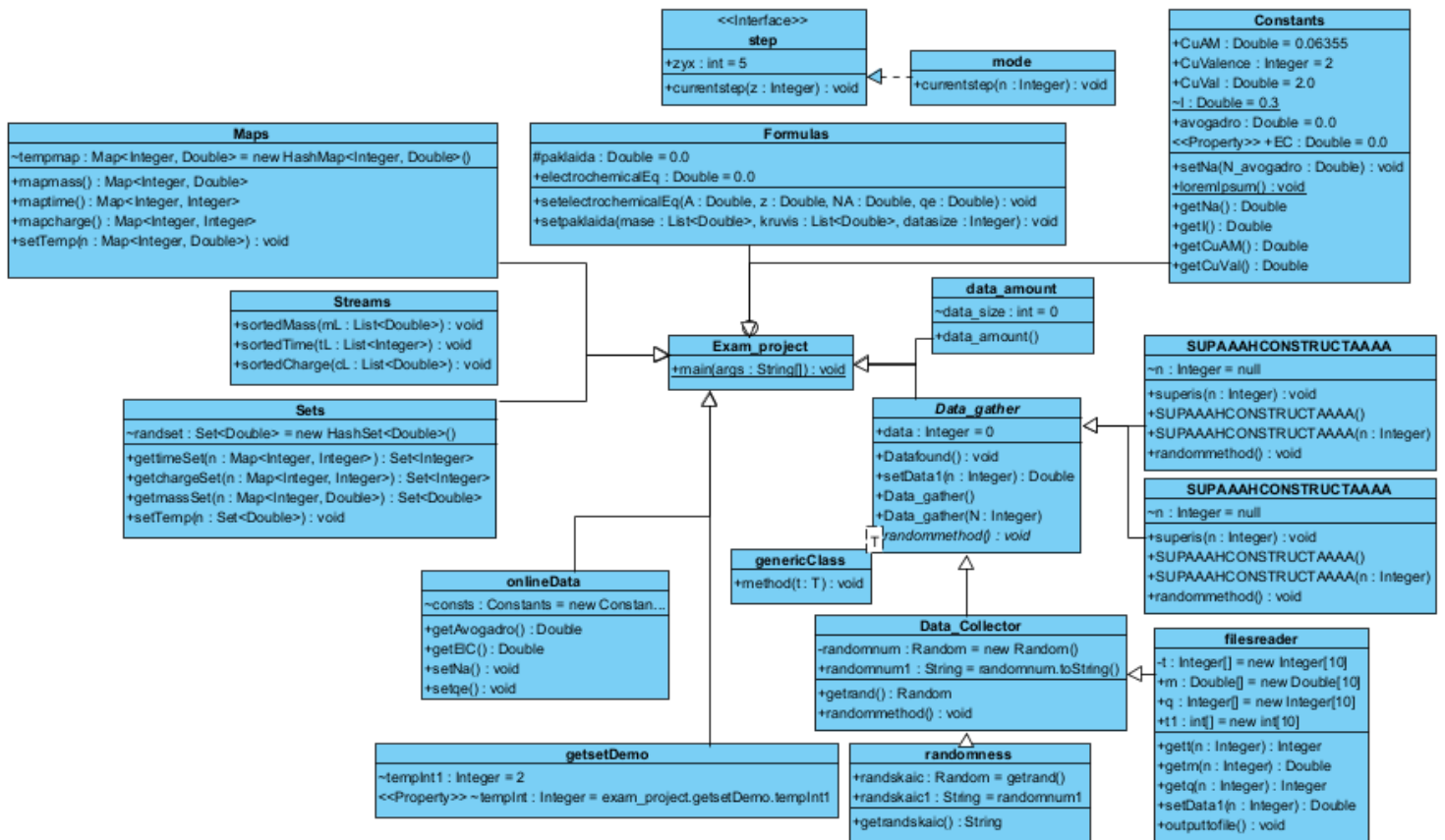
Vienas iš universaliausių užsiėmimų bet kurio fizikos studento(ės) studijų metu yra laboratoriniai darbai. Jie gali būti susiję tiek su elektromagnetizmu, tiek su mechanika, tiek termodinamika. Kai kurie iš šių eksperimentų savo rezultatu saugo kompiuterio failuose. Šių eksperimentų duomenis gali būti sudėtinga apdoroti, arba tai užtrunka labia ilgą laiką. Tam kad palengvinti šių duomenų apdorojimą galima naudoti būtent tam sukurtas programas. Tačiau taip pat šiam tikslui galima pasirašyti specifiskai tam eksperimentui skirtą programą. Šiame projekte bus pritaikomos skirtingos objekcinio programavimo technologijos, tam, kad būtų atliekami E3 laboratorinio darbo “ Elektrolizės Faradėjaus dėsnio tikrinimas” skaičiavimai.

Programa

Programa susideda iš 14 atskirų klasių. Kai kurios yra kritiskai svarbios programos funkcionavimui, kitus yra programoje tik tam, kad būtų imanoma pademonstruoti OOP technologijas ribojant technologijų per klasę skaičių iki 3 per klasę.

Pats programos veikimo principas suskirstytas į 4 veikimo pakopas, arba stadijas: 1 surenkami konstantų duomenys iš internetinio tinklalapio; 2 surenkami eksperimento duomenys iš tekstinio failo; 3 surinkta informacija yra surūšiojama ir atspausdinama (su kai kuriais duomenų saugojimo būdais šis žingsnis nėra privalomas); 4 atliekami eksperimentui svarbūs skaičiavimai.

Programos schema



Pav. 1: Programos UML diagrama

Naudojamų technologijų sąrašas

Čia iš eilės išvardijamos visos programoje demonstruojamos technologijos kai kuriais atvejais vienas metodas pademonstravo kelias technologijas,

1. Bitinės matematinės operacijos
3. nuorodos priskyrimas
4. Blokai *try*, *catch*, *finally*
5. Ciklai
6. Sąlygos
7. Klasė, objekto sukūrimas
8. Konstruktorius
9. Metodas klasėje
10. Duomuo
11. Metodo iškvietimas
12. Operatoriai *this* ir *super*
13. Išimtinės situacijos
14. Panaudojimas *throws* ir *throw*
15. **Anoniminė įdėtinė klasė ir pagal ją objektai viduje klasės**
19. Poklasė
20. Viršklasės duomenų ir metodų panaudojimas
24. Konstruktorius *super* su ir be argumentų
25. **Išimtinės situacijos: klasės ir poklasės**
29. Metodo perrašymas (*overriding*)

30. **Abstrakti klasė ir abstraktus metodas; objekto sukūrimas**
36. Objektų derinimas panaudojant sąsajas
37. Daugybiniškumas (*Generics*): **klasė sukūrimas ir pagal ją objektai**
43. Masyvas: primitiviems duomenims
50. Operatorius *static*: **duomuo, metodas**
56. Teisės ir jų panaudojimas: *private*
58. Teisės ir jų panaudojimas: *protected*
59. Teisės ir jų panaudojimas: *public*
64. Srauto panaudojimas pagal klasę ***Stream*: ciklas kiekvienam**
67. Funkcinė nuoroda Lambda technologijoje
69. GETSET technologija
70. *Aiškinimo* (Annotation) parametrų panaudojimas
72. I/O: failų nuskaitymas ir įrašymas
73. I/O: nuskaitymas iš internetinės nuorodos
83. *Set* duomenų saugojimas, panaudojant pakaitos simboliu (Kolekcijos)
84. ***Map* tipo duomenų saugojimas, panaudojant**
86. **Technologija *toString***

Iš viso: 36 technologijos

Technologijos

1. Bitinės matematinės operacijos

Paprščiausios matematinės operacijos kaip sudėtis ar atimtis .

```
37      * Nr. 1 (Bitines matematinės operacijos)
38      */
39      public void setpaklaida(List<Double> mase, List<Double> kruvis, Integer datasize){
40          Double deltamase1 = (mase.get(datasize-1)-mase.get(0))/1000; //nes g ->kg
41          Double deltakruvis1 = kruvis.get(datasize-1)-kruvis.get(0);
42          Double vidurkis = deltamase1/deltakruvis1;
43          Double tarpineVerte1 = Math.pow(0.0-vidurkis, 2);
44          Double tarpineVerte2 = 0.0;
45          Integer i = 1;
```

2 pav. Bitinių matematinių operacijų panaudojimas klasėje “Formulas”. Ši kodo dalis su tarpiniais veiksmiais naudoja paklaidų skaičiavimo formulę

$$S_n = \sqrt{\frac{(x_1 - \bar{x})^2 + (x_2 - \bar{x})^2 + (x_n - \bar{x})^2}{n-1}} \quad (1)$$

3. nuorodos priskyrimas

Kai sukuriamas elementas, jam galima duoti specifinę vertę, arba jo vertę galima nustatyti kaip kito element vertę. Tuo atveju kuriant elementą po “=” nurodoma nuorod į kitą elementą.

```
8      /**
9      * @param tempInt1 parametras reikalingas 3 technologijai;
10     * @param tempInt pademonstruoja Nr. 3 technologija
11     */
12     Integer tempInt1 = 2;
13     Integer tempInt = tempInt1;
```

3 pav. Nuorodos priskyrimo pavyzdys „getsetDemo“ klasėje.

4. Blokai try, catch, finally

Kai kurie veiksmai gali sukelti problem su programos logika. Arba negaunamas reikiamas priėjimas prie duomens. Tokiais ir panašiais atvejais galima naudoti blokus.

Try: veiksmai, kuriuos mes bandome atlikti; Catch: veiksmai kuriuos mes bandome atlikti, kai Try nesuveikia. Finally paleidziamas kadi r kas benutiktu su try/catch.

```
30      /**
31       * nr. 4 (try),
32       */
33      try{
34          URL url = new URL("https://physics.nist.gov/cuu/Constants/Table/allascii.txt");
35          URLConnection con = url.openConnection();
```

Pav. 4 Try pavyzdys, šiuo atveju programa gali negauti atsakymo iš tinklalapio.

```
51      * nr. 4 (catch),
52      */
53      catch (MalformedURLException e) {
54
55          System.out.println("URL is not working");
56          throw new Exception(e);
57      }
58  }
```

Pav. 5 Catch ismeta zinute, bando apeiti problema su Exception .

```
60      * nr. 4 (finally),
61      */
62      finally {
63          System.out.println("Na collection complete");
64      }
```

Pav. 6 Finally aktyvuojasi kad ir kas benutiktų su try/catch

5. Ciklai

Jei reikia kad tam tikra operacija būtų atlikta tam tikrą kiekį kartų, galima naudoti ciklus kaip for.

```
195      * demonstruojama nr. 5 Ciklai
196      */
197      for(Integer i = 1; i<=DataPoints;i++)
198          HMmass.put(i, flr.getm(i));
199          HMtime.put(i, flr.gett(i));
200          HMcharge.put(i, flr.getq(i));
201      }
```

Pav. 7 for ciklas naudojamas tam, kad programa atliktų specifines operacijas jai nurodytą kiekį kartų.

6. Sąlygos


```

34      switch (n){
35          /**
36           * demonstruojama Nr. 6 salygos (bet vietoj if else naudojami switchai)
37           */
38          case 1:
39              System.out.println("\n acquiring online data \n");
40              break;
41          case 2:
42              System.out.println("\n collecting data from experiment file \n");
43              break;

```

Pav. 8 sąlyga nurodo atlikti specijinę operaciją tik jei galioja tam tikri nurodyti parametria, tam galima tiek operatorius if/else if/else, tiek case switch.

7. Klasė, objekto sukūrimas

```

66      * demonstruojama Nr. 7 klase
67      */
68      public class Exam_project {

```

Pav. 9 sukuriamas klasė, tai kaip brėžinys, koks turės būti kuriamas objektas.

```

171      /**
172      * demonstruojama Nr. 7 objekto sukurimas
173      */
174      onlineData web = new onlineData();

```

Pav. 10 pagal kitą klasę programoje sukuriamas objektas, kuris gali atlikti jo klasėje nurodytas funkcijas bei saugo klasėje sukurtus elementus.

8. Konstruktorius

```

18      * Nr. 8 konstruktorius
19      */
20      public data_amount() {
21          System.out.println("input the amount of data points requested");
22          Scanner in = new Scanner(System.in);
23          data_size = in.nextInt();
24          this.data_size=data_size;
25      }

```

Pav. 11 konstruktorius pasileidžia sukuriant klasės objektą, tačiau jam taip pat galima duoti specifines komandas, taip įvedant pradines objekto vertes. Šiuo atveju konstruktorius panaudojamas duomenų įvesčiai iš klaviatūros.

9. Metodas klasėje

```

90  * Demonstruojamas Nr. 9 Metodas klaseje
91  */
92  public void setNa(Double N_avogadro){
93      N_avogadro*= Math.pow(10, 23);
94      this.avogadro = N_avogadro;
95      System.out.println("obtained avogadro constant: "+avogadro);
96  }

```

Pav. 12 konstantų klasėje esantis metodas, jis atlieka jam priskirtas operacijas po to kai sukuriamas objektas. Čiuo atveju jis nustato ir išsaugo Avogadro skaičių, gautą iš kito objekto.

10. Duomuo

```

13  * Nr. 10 Duomuo.
14  */
15  int data_size = 0;

```

Pav. 13 tai primityvūs duomenų saugojimo tipai, pvz. skaičiai gali būti saugomi kaip duomuo int, double float ir t.t.

11. Metodo iškviertimas

```

176  * Demonstruojama Nr. 11 metodo iskvietimas
177  */
178  web.setNa();
179  web.setqe();

```

Pav. 14 kaip anksčiau minėta, sukūrus objektą, galima iškviesti jo funkcijas(metodus) šios funkcijos gali turėti parametrus arba jų neturėti, gali atlikti daug veiksmų arba nieko nedaryti, gali grąžinti vertę, arba nieko negrąžinti.

12. Operatoriai this ir super

Super naudojamas poklasės objekte, kai norima panaudoti viršklasės metodą, This naudojamas būtent to specifinio objekto atributams.

```

16  * @param n kartu su visu metodu pademonstruoja Nr. 12;
17  * @throws Exception
18  */
19  public void superis(Integer n) throws Exception{
20      super.setDatal(n);
21      super.Datafound();
22      this.n=n;

```

Pav. 15 objekte panaudojamas super operatorius, kviečiantis viršklasės metodą, iškart po to this operatorius išsaugo metodo parametro paduotą funkciją kaip savo objekto element vertę.

13. Išimties situacijos

Normalaus programos paleidimo metu gali iškilti kliūtys, kurios priverčia program per anksti pabaigti veikimą. Tam kad taip neįvyktų, naudojamos išimties.

```
26      * demonstruojama Nr. 13 išimties situacijos
27      * @param n gaunamas duomenų kiekis
28      * @return null metodas nieko negrąžina
29      * @throws Exception
30      */
31      public Double setPatal(Integer n) throws Exception{
32
33          this.data=n;
34          return null;
```

Pav. 16 išimtis gali naudoti metodai(skirtingais operatoriais). Šiuo atveju išimtis yra panaudojama tik tam, kad pademonstruoti, kaip metodas panaudoja išimtis.

14. Panaudojimas throws ir throw

Abu šie operatoriai pažymi, kad gali būti panaudota išimtis. Tačiau throw naudojamas panaudoti tiek išimtis, tiek specifinį kodo bloką, tuo tarpu throws yra naudojamas prie metodo aprašymo, ir nurodo, kad metodas gali panaudoti nurodytą išimtį.

```
25      * nr. 14 panaudojamas throws
26      * @throws Exception išimtis su interneto nuorodomis
27      * is interneto nuskaitymas ir išsaugomas avogadro skaičius
28      */
29      public void setNa() throws Exception{
30          /**
31           * nr. 4 (try),
32           */
33          try{
34              URL url = new URL("https://physics.nist.gov/cuu/Constants/Table/allascii.txt")
35              URLConnection con = url.openConnection();
```

Pav. 17 metodas nurodo, kad gali panaudoti išimtį.

```
90      catch (MalformedURLException e) {
91
92          System.out.println("URL is not working")
93          /**
94           * nr. 14 throw
95           */
96          throw new Exception(e);
```

Pav. 18 klaidos atveju metodui nurodyta panaudoti išimtį.

15. Anoniminė įdėtinė klasė ir pagal ją objektai viduje klasės

Anoniminė klasė gali praplėsti kitas klases. Taip galima panaudoti kitos klasės metodus nepriverčiant klasę, kurioje yra anoniminė klasė, praplėsti anoniminės klasės panaudojamą klasę.

```
155 * Nr.15 Anoniminė idėtinė klasė ir pagal ją objektai viduje klasės
156 */
157 Thread t = new Thread()
158 {
159     @Override/**
160     * perrasomas paleidimo metodus, paleidimo pradzioja i ekrana is
161     */
162     public void run()
163     {
164         System.out.println("program start");
165     }
166 };
```

Pav. 19 šiuo atveju anoniminė klasė praplėčia Thread klasę, nekuriant papildomos klasės, kuri praplėstų Thread.

19. Poklasė

Trumpai tariant, poklasė paveldi viršklasę, t.y. poklasė gali naudoti viršklasės metodus,, public kintamuosius ir t.t.

```
12 * Nr. 19 si klase yra poklase (extends Exam_project)
13 */
14 public class Maps extends Exam_project{
15 }
```

Pav. 20 šiuo atveju paveldėjimas nėra būtinas, tačiau Maps poklasė gali naudoti visus public metodus ir elementus Exam_project klasėje.

20. Viršklasės duomenų ir metodų panaudojimas

```
14 * demonstruojama Nr. 20 Viršklasės duomenų ir metodų panaudojimas
15 */
16 public Random randskaic = getrand();
17 /**
18 * saugomas duomuo randskaic, tai jokios svarbios funkcijos neturintis duomuo,
19 * tačiau jo vertė yra nuoroda į viršklasės elementą
20 */
21 public String randskaicl = randomnum1;
```

Pav. 21 tam kad sukurtų du atskirus duomenis, poklasė pasinaudoja viršklasės duomeniu, bei metodu.

24. Konstruktorius super su ir be argumentu

Kaip jau minėta, super operatorius naudojamas kaip naudojamas viršklasės metodas(nevisada būtinas) su super operatoriumi poklasės konstruktorius gali panaudoti viršklasės konstruktorių, taip panaudodama jo operacijas, kai kurie konstruktoriai turi parametrus, kai kurie ne.

```
25  *Nr. 24 konstruktorius super be argumentu
26  */
27  public SUPAAHCONSTRUCTAAAA() {
28      super();
29  }
29  /**
30   *Nr.24 konstruktorius super be argumentu
31   * @param n demonstraacijai reikalingas parametras
32   */
33  public SUPAAHCONSTRUCTAAAA(Integer n) {
34      super(n);
35  }}
```

Pav. 22 jau anksčiau demonstruotas metodas panaudoja super operatorių Viršklasės konstruktorius su ir be argumentų.

25. Išimties situacijos: klasės ir poklasės

Išimtys turi savo atskiras klases ir poklases. Pavyzdžiui IOException yra klasės Exception poklasė. Priklausomai nuo situacijos, galima panaudoti tiek specifinę Exception poklasę, tiek pačią klasę Exception

```
63  * demonstruojama nr. 25 Išimtinės situacijos: poklasė
64  */
65  catch (FileNotFoundException e) {
66      System.out.println("File Not Found");
67      throw new EmptyStackException();
68  }
```

Pav. 23 panaudojamos Exception poklasės IOException poklasės

```
39  * demonstruojama nr. 25 Išimtinės situacijos: klasė
40  */
41  @Override
42  public Double setDatal(Integer n) throws Exception {
```

Pav. 24 panaudojama Exception klasė

29. Metodo perrašymas(overriding)

Poklasės gali perrašyti višklasėje esantį metodą(pvz. Abstraktų metodą) vienas iš būdų tai padaryti yra su @Override anotacija.

```
26 class mode implements step{
27     /**
28      * demonstruojama Nr. 29 Metodo perrasydas
29      * perrasydas interface metodus, metodus nusako, kuriame zingsnije yra programa
30      * @param n programos zingsnis
31      */
32     @Override
33     public void currentstep(Integer n){
34         switch (n){
```

Pav. 25 klasė panaudoja @Override anotaciją, kad perrašytų abstraktų metodą.

30. Abstrakti klasė

Mes negalime sukurti objekto iš abstrakčios klasės(klasė, kurios bent vienas iš metodų yra abstraktus, t.y. nėra aprašytas. Tačiau šios klasės metodus ir duomenis galima panaudoti sukūrus objektą iš jo poklasių

```
8 * Nr.30 abstrakti klase
9 */
public abstract class Data_gather extends Exam_project{
```

Pav. 26 sukuriamas abstrakti klasė

```
30 * Nr. 30 abstraktus metodus
30 */
abstract void randommethod();
```

Pav. 27 sukuriamas abstraktus metodus

```
29 @Override
30 public void randommethod() {
31 }
```

Pav. 28 poklasėje perrašomas abstraktus metodus

```
183 /**
184 * Nr. 30 panaudojamas anksčiau buvęs abstraktus metodus
185 */
186 Data_Collector obj1 = new Data_Collector();
187 obj1.randommethod();
```

Pav. 29 objektas panaudoja anksčiau abstrakčią klasę

36. Objektų derinimas pagal sąsajas(interface)

Interface klasės tipo kintamojo suderinimas prilyginant jį interface implementacijos objektui

```
11  * Demonstruojama technologija Nr. 36 sąsajos interface
12  **/
13  interface step {
14      /**
15       * abstraktus metodas
16       * @param z programos žingsnis
17       */
18      public void currentstep(Integer z);
```

Pav. 30 interface klasė

```
168  /**
169   * 36 Objektų derinimas pagal sąsajas
170   */
171   mode rezimas = new mode();
172   step zyx = rezimas;
```

Pav. 31 interface klasės tipo kintamasis prilyginamas interface implementacijos objektui.

37 Daugybiniškumas (generics): klasės sukūrimas ir opjektai pagal ją.

Generics klasė savo kintamuosiuose ir metodų parametruose neturi tipo. Vietoj to dažniausiai naudojamas žymėjimas T. sukūrus objektą ir naudojant kintamuosius bei metodus visi T žymėjimai patampa objekto naudojamu kintamojo tipu, o kintamieji kaip t, objekto kintamuoju nurodomu.

```
3  * pademonstruojama generics technologija
4  * Technologija Nr. 37 Daugybiniškumas (Generics): klasės sukūrimas ir pagal ją objekt
5  * @author Administratorius
6  * @param <T> reiškia kad ši klase yra "generic",
7  * t.y. T raide bus pakeista "main" programoje nurodytu tipu.
8  */
9  public class genericClass <T>{
10      /**
11       * @param t generic parametras.
12       * Šis metodas main programoje objekto pagalba
13       * naudojamas išspinti du i'hardcode'intus
14       * duomenis iš klases "Constants"
15       */
16       public void method(T t){
17           System.out.println(t+"\n");
```

Pav. 32 sukuriama generic klasė.

```

228      * Nr. 37 panaudojamas generic objektas
229      */
230      genericClass printer = new genericClass();
231      System.out.println("\n hardcoded constants: ");
232      Constants consts = new Constants();
233      printer.method(consts.CuValence);
234      printer.method(consts.CuAM);
235

```

Pav. 33 generic klasės objektas panaudoja “constants” klasės kintamuosius....

```

77      static class Constants{
78          public Double CuAM = 0.06355;
79          public Integer CuValence = 2;

```

Pav. 34 ...kurie yra skirtingų tipų.

43. Masyvas: primityviems duomenims

Primityvių duomenų(kaip int, double ir t.t.) masyvas java kalboje yra laikomas objektu, galinčiu savyje talpinti to paties tipo vertes. Kiekviena vertė yra elementas, kuris gali būti pasiektas naudojant sveiką skaičiaus indeksą.

```

18      * technologija nr 43 duomenys saugomi masyve
19      */
20      public int[] t1 = new int[10];

```

Pav. 35 sukuriamas primityvių element masyvas. Atydžiai pasižiūrėjus galima pastebėti, kad jo kūrimas labai panašus į objekto kūrimą.

50. Operatorius static: duomuo, metodas

Static deklaracija reiškia, kad kad ir kiek inicijuosime tą duomenį ar metodą, jis visada bus vienas(visos inicijacijos turės dalintis tuo pačiu duomeniu/masyvu) taip kintamieji yra nepriklausomi nuo objektų, metodai gali būti šaukiami nesukuriant objekto.

```

81      /**
82      * Nr. 50 saugomas static duomuo
83      */
84      static Double I =0.3;

```

Pav. 36 statinis duomuo


```

107 |      * Nr. 50 statinis metodas
108 |      */
109 |      public static void loremIpsum() {
110 |          System.out.println("this is static method");
111 |      }
112 |

```

Pav. 37 statinis metodas

56 Teisės ir jų panaudojimas: private

Private teisė reiškia kad klasės duomuo yra matomas tik toje klasėje. Jo nemato net poklasės, tačiau galima jį perduoti naudojant public metodą.

```

10 |      /**
11 |       * nr.56 privatus duomuo
12 |       */
13 |       private Random randomnum = new Random();
14 |      /**

```

Pav. 38 privatus duomuo

```

22 |      public Random getrand() {
23 |          return randomnum;
24 |      }

```

Pav. 39 viešas metodas, perduodantis private metodą.

58. Teisės ir jų panaudojimas: protected

Protected raktažodis reiškia kad duomuo prieinamas iš: tos pačios klasės; to paties paketo puklasėms; kitoms to paties paketo klasėms; poklasėms iš kitų paketų.

Tačiau kitų paketų klasės, kurios nėra protected element klasės poklasės neturi prie jo priėjimo.

```

13 |      /**
14 |       * Nr. 58 saugomas protected duomuo paklaida, tai yra duomenų rezultatu paklaida
15 |       */
16 |       protected Double paklaida = 0.0;

```

Pav. 40 protected duomuo

59. Teisės ir jų panaudojimas: public

Duomenis, metodus bei klases gali pasiekti bet kuri kita klasė. Jo pagalba galima perduoti ir privačius laukus.

```

15  * Nr. 59 - public duomuo
16  */
17  public Random randskaic = getrand();
18  /**
19   * saugomas duomuo randskaic, tai jokios svarbios funkcijos neturintis duomuo,
20   * tačiau jo vertė yra nuoroda į viršklasės elementą
21   */
22  public String randskaic1 = randomnum1;
23  /**
24   * Nr. 59 public metodas gražinantis duomeni
25   * @return randskaic grazina duomeni randskaic
26   */
27  public String getrandskaic(){
28      return randskaic.toString();
29  }

```

Pav. 41 public laukai, ir metodai. Public metodas grąžina private lauką.

64. Srauto panaudojimas pagal klasę *Stream*: ciklas kiekvienam

Stream atvaizduoja element seką, bei palaiko skirting operacijų atlikumos Stream sekos nariams.

Ciklas kiekvienam(forEach()) atlieka nurodytą funkciją su kiekvienu sekos nariu per juos iteruodamas.

```

40  * 64 streame naudojamas forEach ciklas
41  */
42  Stream.of(cL)
43      .sorted()
44      .forEach((I) -> System.out.format("surusiuotas kruvis"+I+"\n"));

```

Pav. 42 forEach ciklo pritaikymas isprintinti surusiuotus List'o narius.

67. Funkcinė nuoroda Lambda technologijoje

Lambda technologija tai metodas, kuris gali būti sukurtas nebūdamas jokioje klasėje. Iš dalies tai yra anoniminė klasė. Lambda technologijos nuoroda žymima “->”.

```

21  * 67 funkcinė nuoroda Lambda
22  */
23  .forEach((I) -> System.out.format("surusiuota mase"+I+"\n"));

```

Pav. 43 Stream'as panašus į Pav. 42, tačiau naudojamas kitam List'ui. 23'oje eilutėje matomas Lambda technologijos panaudojimas.

69. GETSET technologija

Klasės metodai skirti išsaugoti arba perleisti jos laukus pagal konvenciją setters rašomi kaip set metodas, kurio setinamas kintamasis iš didžiosios raidės eina iš kart o set. Tas pats galioja getters galioja ta pati nuostatata, tik su get.

```
24      * Nr. 69 Get dalis  
25      * @return tempint grazina tempInt verte  
26      */  
27      public Integer getTempInt () {  
28          return tempInt;
```

Pav. 44 get metodai naudojami gražinti atributą

```
6      * Nr.69 Set technologijos dalis:  
7      * @param NablaXB0 - metodo parametras, pakeiciantis  
8      * tempInt verte  
9      */  
10     public void setTempInt(Integer NablaXB0) {  
11         this.tempInt = NablaXB0;
```

Pav. 45 set metodai naudojami klasės atributui priskiria metodo parametron vertę, dažniausiai naudojant this operatorių.

70 Aiškinimo (annotation) parametrų panaudojimas

Skirtingai nei java “tags” anotacijos gali būti perskaitomos iš klasių failų. Tai sintaksiniai metaduomenys, kuriuos galima naudoti su java kodu. Trumpai tariant, anotacijos suteikia informaciją kompiliatoriui apie kodą. Anotacija kaip @Override sukelia kompiliacijos klaidą, jei abstrakčios klasės poklasė neperrašo abstrakčių metodų.

```
55      @Deprecated  
56      public void setTemp(Set<Double> n) {  
57          this.randset=n;  
58      }
```

Pav. 46 @Deprecated anotacija praneša kompiliatoriui, kad metodas nebenaudojamas, tolimesnis jo panaudojimas sukeltų klaidą.

72. I/O failų nuskaitymas ir įrašymas

Java galimybė nuskaityti ir įrašyti informaciją į failus. Tam dažniausiai galima naudoti Streams. Tačiau galima naudotis ir klasėmis kaip Scanner input'ui, arba FileWriter output'ui.

```

46     try{
47         try (Scanner datafile = new Scanner(new File("C:\\Users\\Administratorius\\Documents\\NetBeansProjects\\Exam_project\\
48             String line = datafile.nextLine();
49             System.out.println(line);
50
51         SUPAAAHCONSTRUCTAAAA supconst =new SUPAAAHCONSTRUCTAAAA ();
52         supconst.superis(n);
53         for (Integer i = 1; i<=n; i++) {
54             m[i]= datafile.nextDouble();
55             t[i]= datafile.nextInt();
56             q[i]= datafile.nextInt();
57             this.m[i]=m[i];
58             this.t[i]=t[i];
59             this.q[i]=q[i];
60         }
61     }

```

Pav. 47 programoje naudojama logika nuskaityti informaciją nuo failo

```

74     *metodo tikslas, įrašyti žinutę į failą. demonstruojama nr. 72 įrašymo dalis
75     */
76     public void outputtofile() {
77         try {
78             try (FileWriter outputas = new FileWriter("outputofailas.txt")) {
79                 outputas.write("rezultatai: isspausdinami konsoles ivestyje");
80             }
81         } catch (IOException e) {
82             System.out.println("An error occurred.");
83         }
84     }
85     /**

```

Pav. 48 programoje (ne)naudojama įrašymo į failą logika

73. I/O nuskaitymas iš išorinės nuorodos

```

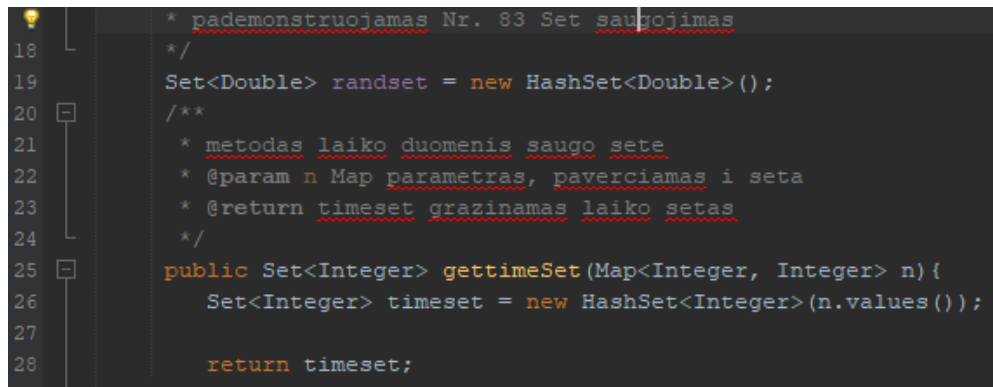
68     * pademonstruojama nr 73 nuskaitymas iš internetinės nuorodos
69     * iš interneto nuskaitymas ir issaugomas elektrono kruvis
70     * @throws IOException isimtis dėl interneto nuorodu
71     */
72     public void setqe() throws IOException{
73         try{
74             URL url = new URL("https://physics.nist.gov/cuu/Constants/Table/allascii.txt");
75             URLConnection con = url.openConnection();
76             InputStream is =con.getInputStream();
77             BufferedReader br = new BufferedReader(new InputStreamReader(is));
78             String line = null;
79
80             while ((line = br.readLine()) != null) {
81                 if (line.contains("atomic unit of charge "))
82                 {
83                     String str1 = line;
84                     String stripedValue = (str1.replaceAll("[\\s+a-zA-Z(-)^~_ :]", ""));
85                     Double dbl1 = Double.parseDouble(stripedValue);
86                     consts.setEC(dbl1);
87                 }
88             }
89             br.close();
90         } catch (MalformedURLException e) {

```

Pav. 49 programoje naudojam URL klasė, kad nuskaityti internet nuorodą, paimiti programai aktualią raidžių ir skaičių liniją, pašalinti visus nereikalingus simbolius ir į konstantų klasę įrašyti aktualią vertę. Šiuo atveju tai elektrono krūvis. **Svarbu:** jei mes ne'throw'insime IOException arba Exception klasių šie veiksmai nėra leidžiami.

83. Set duomenų saugojimas, panaudojant pakaitos simboliu (Kolekcijos)

Set klasė paveldi kolekcijos sąsają(Interface). Viena iš naujų Set'o galimybių palyginus su jos pirmtkais tai tai, kad Set neleidžia dėti pasikartojančių elementų. Kadangi Set yra sąsaja, norėdami sukurti objektą, mes privalome naudotis klase, kuri paveldi mūsų List'ą(kurį norime patalpinti į Set'a).

A screenshot of a code editor showing Java code. The code is for a class named 'pademonstruojamas Nr. 83 Set saugojimas'. It includes a comment '*/' and a line 'Set<Double> randset = new HashSet<Double>();'. There is a block of comments: '/*', '* metodus laiko duomenis saugo sete', '* @param n Map parametras, paverčiamas i seta', '* @return timeset grazinamas laiko setas', and '*/'. Below this is a method signature 'public Set<Integer> gettimeSet(Map<Integer, Integer> n){' and its implementation: 'Set<Integer> timeset = new HashSet<Integer>(n.values());' followed by 'return timeset;'. The code is numbered 18 to 28 on the left margin.

Pav. 50 Set panaudojimas programoje. Šiuo atveju į Set'a dedamas Map(speaking of Maps...)

84. Map tipo duomenų saugojimas, panaudojant pakaitos simbolius (Kolekcijos)

Java kalboje Map klasė yra sąsaja, kurią paveldi SortedMap, bei kuria implementuoja HashMap, Map kaupia vertes pagal "raktus"(keys) Map laidžia turėti pasikartojančias vertes, tačiau jame negali pasikartoti raktai.

```

15  /**
16   * Nr. 84 sukuriamas Map, kuriame jei reikia galima saugoti duomenis
17   */
18   Map<Integer, Double> tempmap = new HashMap<Integer, Double>();
19   /**
20   * metodas sukuria mases map
21   * @return HMmass grazinamas mases map
22   */
23   public Map<Integer, Double> mapmass(){
24       Map<Integer, Double> HMmass = new HashMap<Integer, Double>();
25       return HMmass;

```

Pav. 51 Map panaudojimas programoje. Matome kad kuriant Map reikia dviejų kintamųjų. Pirmasis(Integer) yra mūsų raktas, antrasis(čia Double) yra vertė, kurią norime įdėti į Map.

86: toString technologija

Tai “in-built” metodas, paverčiantis bet kokio primityvaus(pvz int, double, float) tipo duomenį į String tipo objektą.

```

14  /**
15   * nr. 86 demonstruoja toString technologija
16   */
17   public String randomnum1 = randomnum.toString();
18  /**

```

Pav. 52 toString panaudojimas programoje

Svarbu paminėti

Šis projektas neišvengė nusižengimo taisyklai “1 technologijai pademonstruoti skirti vieną klasę”. Bandant perkurti projektą, kad jis labiau atitiktų šį reikalavimą(3 klasės → 14 klasių) buvo suprasta, kad daugelis panaudotų technologijų turės 0 praktinių panaudojimų. Taip pat dėl laiko ir gebėjimų stokos buvo nuspręsta nedaryti vienos iš privalomų technologijų: 78. Grafinis elementas mygtukas. Mygtuko paspaudimas. Autorius gerai supranta, kad dėl to kentės galutinis pažymys ir yra su tuo susitaikęs.

Literatūra

1. A. Medeišis „Mechanika, molekulinė fizika, elektra ir magnetizmas. Fizikos praktikumas.“, Vilnius, *Vilniaus universiteto leidykla*, 2000.
2. A. Matvejevas, „Elektra ir magnetizmas“, Vilnius, *Mokslas*, 1991.
3. V. Rinkevičius, „Elektra ir magnetizmas“, Vilnius, *Vilniaus universiteto leidykla*, 2001