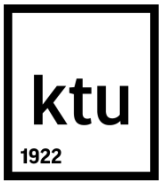


KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

Erikas Bijeikis IFIN-2/1

DUOMENŲ STRUKTŪRŲ PAGRINDŲ ANTRO LABORATORINIO DARBO

Ataskaita

Vadovas
Lekt. Andrėj Afonin

Kaunas 2023,

Turinys

Darbo užduotis

1. Pagal duotą Automobilis klasės pavyzdį sukurti individualiai pasirinkto objekto klasę, įdiegiančią KTUable
sąsają (interfeisą). Tai gali būti prekės iš didmeninės ir mažmeninės prekybos asortimento, elektronikos komponentai,
kompiuteriai, knygos, filmai, žaidimai, rūbai, geografija, istorija, multimedijos kūriniai, kelionės, darbuotojai, sporto
varžybų ir dalyvių duomenys ir kitokie elementai. Turibūti realizuota:
a. daugiau nei penkios įvairių tipų objekto charakteristikos;
b. duomenų kontrolė, panaudojant išimtis (exceptions) - try-catch blokai ar kitos priemonės;
c. programinį kodą rašyti į individualų paketą Lab2Pavardė (pervardinti paketąLab2individualus)
2. Patikrinti individualios klasės panaudojimą savo sukurtose bandymų klasėse (nebūtina aklai kopijuoti
paketo Laboras2demo struktūros, t.y. kurti tiek pat klasių ir tokių pačių metodų).
3. Papildyti studijosKTU paketą bent keturiais naujais metodais, pasirinktais iš Javos sąsajos List (vienas set
metodas ir vienas remove metodas privalomi)1
. Tam būtina:
a. pasirinktus metodus įtraukti į studijosKTU paketo sąsają ListADT
(arba į naują sąsają, kuri paveldėtų sąsają ListADT);
b. realizuoti šiuos pasirinktus metodus ListKTU klasėje;
c. išbandyti realizuotus metodus savo individualaus paketo Lab2Pavardė klasėse.
4. Individualioje klasėje parašyti metodą compareTo. Pagal duotą Automobilis klasės pavyzdįsukurti daugiau
nei 2 komparatorius (pagal vieną ir daugiau individualios klasės laukų). Patikrinti sukurtus komparatorius
rikiavimo metodų pagalba.
5. Sukurti klasę, kurioje būtų individualių elementų atrankos metodai pagal pasirenkamus kriterjus.
6. Sukurti grafinę naudotojo sąsają.
a. Išnagrinėti klasės DialogasSuKlientu metodą bendravimasSuKlientu.
b. Sukurti savo pakete klasę bendravimui su klientu su grafinės vartotojo sąsajos elementais (meniu,
mygtukai, teksto laukai, ...) savo klasės duomenų apdorojimui.
c. Pradinis duomenų nuskaitymas iš pasirenkamo failo (failo pasirinkimo dialogas, pavyzdžiui,
JFileChooser klasė) ir duomenų atvaizdavimas lentelėje (pavyzdžiui, JTable klasė) privalomi.
PASTABA: Galima naudoti NetBeans dizainerį (žr. paskaitos „Vartotojo sąsajos

programavimas - pirma dalis“
31-oji skaidrė). Taip pat pateiktas „Nedidelis Vartotojo sąsajos pavyzdys“.
7. GREITAVEIKA (efektyvumo tyrimas):
a. Sudaryti didelės apimties individualių elementų generavimo metodą greitaveikos tyrimams;
b. Atlikti bent dviejų jūsų realizuotų sąrašo apdorojimo metodų greitaveikostyrimą;
c. Parašyti dar bent vieną rikiavimo metodą (kurio nėra ListKTU klasėje) ir ištirti naujų ir jau esamų
rikiavimo algoritmų greitaveiką;
d. Greitaveikos eksperimentą (skaičiavimus) pakartoti ne mažiau 3 kartus toje pačioje aplinkoje /
kompiuteryje (su skirtinga apkrova), gautus rezultatus palyginti tarpusavyje, pažymėti skirtumus.

Duomenų klasės „Kompiuteris“ kodas

```
package Lab2Bijeikis;

import Laboras2demo.Automobilis;
import studijosKTU.KTUable;
import studijosKTU.Ks;
import studijosKTU.ListKTU;

import java.util.*;

/**
 * Duomenų klasė
 */
public class Kompiuteris implements KTUable<Kompiuteris> {
   private String procesoriausGamintojas;
   private String procesoriausModelis;
   private String ramuKiekis;
   private String kietojoDiskoVieta;
   private int nasumas;
   private double kaina;

   /**
    * Metodas Paimti Kainai
    * @return Kaina
    */
   public double getKaina() {
      return kaina;
   }

   /**
    * Metodas nustatyti kainai
    * @param kaina kainos reiksme
    */
   public void setKaina(double kaina) {
      this.kaina = kaina;
   }
   /**
    * Metodas Paimti Procesoriaus Gamintojui
    * @return procesoriaus gamintojas
```

```
 */
public String getProcesoriausGamintojas() {
   return procesoriausGamintojas;
}

/**
 * Metodas nustatyti procesoriaus Gamintoja
 * @param procesoriausGamintojas
 */
public void setProcesoriausGamintojas(String procesoriausGamintojas) {
   this.procesoriausGamintojas = procesoriausGamintojas;
}

/**
 * Metodas Paimti procesoriaus modeli
 * @return procesoriaus modeli
 */
public String getProcesoriausModelis() {
   return procesoriausModelis;
}

/**
 * Metodas nustatyti procesoriaus modeli
 * @param procesoriausModelis
 */
public void setProcesoriausModelis(String procesoriausModelis) {
   this.procesoriausModelis = procesoriausModelis;
}

/**
 * Metodas Paimti nasuma
 * @return nasuma
 */
public int getNasumas() {
   return nasumas;
}

/**
 * Metodas nustatyti Nasumas
 * @param nasumas
 */
public void setNasumas(int nasumas) {
   this.nasumas = nasumas;
}

/**
 * Metodas Paimti ramu kieki
 * @return ramu kieki
 */
public String getRamuKiekis() {
   return ramuKiekis;
}

/**
 * Metodas nustatyti Ramu kieki
 * @param ramuKiekis
 */
public void setRamuKiekis(String ramuKiekis) {
   this.ramuKiekis = ramuKiekis;
}
/**
```

```
 * Metodas paimti Kietojo disko vieta
 * @return Kietojo disko vieta
 */
public String getKietojoDiskoVieta() {
   return kietojoDiskoVieta;
}

/**
 * Metodas nustatyti kietojo disko vietai
 * @param kietojoDiskoVieta
 */
public void setKietojoDiskoVieta(String kietojoDiskoVieta) {
   this.kietojoDiskoVieta = kietojoDiskoVieta;
}

/**
 * Tuščias Konstruktorius
 */
public Kompiuteris(){}

/**
 * Metodas Kuris nieko nedaro
 * @return null
 */
public String validate(){
   return null;
}

/**
 * Konstruktorius
 * @param procesoriausGamintojas Gamintojas procesoriaus
 * @param procesoriausModelis Procesoriaus Modelis
 * @param ramuKiekis Ramu kiekis
 * @param kietojoDiskoVieta Kietojo disko vieta
 * @param nasumas Nasumas
 * @param kaina Kaina
 */
public Kompiuteris(String procesoriausGamintojas,String procesoriausModelis, String ramuKiekis,
            String kietojoDiskoVieta, int nasumas, double kaina){
   this.procesoriausGamintojas = procesoriausGamintojas;
   this.procesoriausModelis = procesoriausModelis;
   this.ramuKiekis = ramuKiekis;
   this.kietojoDiskoVieta = kietojoDiskoVieta;
   this.nasumas = nasumas;
   this.kaina = kaina;
}

/**
 * Konstrutorius
 * @param dataString duomenu eilute
 */
public Kompiuteris(String dataString){
   this.parse(dataString);
}

/**
 * Metodas sukurti duomenu Klase
 * @param dataString duomenu eilute
 * @return nauja duomenu klase
 */
```

```
    public Kompiuteris create(String dataString) {
        return new Kompiuteris(dataString);
    }

    /**
     * Metodas ivesti duomenis i klase is duomenu eilutes
     * @param dataString duomenu eilute
     */
    public final void parse(String dataString) {
        try {
            String[] data = dataString.split(";");
            procesoriausGamintojas = data[0];
            procesoriausModelis = data[1];
            ramuKiekis = data[2];
            kietojoDiskoVieta = data[3];
            nasumas = Integer.parseInt(data[4]);
            try {
                setKaina(Double.parseDouble(data[5]));
            }
                catch(NumberFormatException e){
                String line = data[5].replace(',','.');
                setKaina(Double.parseDouble(line));
                }

        } catch (InputMismatchException e) {
            Ks.ern("Blogas duomenų formatas apie kompiuterį -> " + dataString);
        } catch (NoSuchElementException e) {
            Ks.ern("Trūksta duomenų apie kompiuterį -> " + dataString);
        }
    }

    /**
     * Metodas kuris sukuria duomenu eilute
     * @return duomenu eilute
     */
    @Override
    public String toString() {
        return  String.format("| %-23s | %-20s | %11s | %19s | %7d | %7.2f |\n"+
                    "|-------------------------------------------------------------------------------------------------------|\n",
                    procesoriausGamintojas,procesoriausModelis,ramuKiekis,kietojoDiskoVieta,nasumas,kaina);
    }

    /**
     * Metodas kuris sukuria duomenu masyva
     * @return duomenu masyva
     */
    public String[] toStringArray(){
        String[] array =
{procesoriausGamintojas,procesoriausModelis,ramuKiekis,kietojoDiskoVieta,Integer.toString(nasumas),Double.toString(kai
na)};
        return array;
    }

    /**
     * Komparatorius pagal gamintoja
     */

    public final static Comparator<Kompiuteris> pagalGamintoja = new Comparator <Kompiuteris>(){
        public int compare(Kompiuteris komp1, Kompiuteris komp2) {
            int cmp = komp1.getProcesoriausGamintojas().compareTo(komp2.getProcesoriausGamintojas());
            return cmp;
```

```
        }
    } ;
    /**
     * Komparatorius pagal Modeli
     */
    public final static Comparator<Kompiuteris> pagalModeli = new Comparator <Kompiuteris>(){
        public int compare(Kompiuteris komp1, Kompiuteris komp2) {
            int cmp = komp1.getProcesoriausModelis().compareTo(komp2.getProcesoriausModelis());
            return cmp;
        }
    } ;
    /**
     * Komparatorius pagal Nasuma
     */
    public final static Comparator<Kompiuteris> pagalNasuma = new Comparator <Kompiuteris>(){
        public int compare(Kompiuteris komp1, Kompiuteris komp2) {
            int cmp = -1;
            if(komp1.getNasumas() > komp2.getNasumas())
                cmp = 1;
            else if(komp1.getNasumas() == komp2.getNasumas())
                cmp = 0;
            return cmp;
        }
    } ;
    /**
     * Komparatorius pagal Kaina
     */
    public final static Comparator<Kompiuteris> pagalKaina = new Comparator<Kompiuteris>() {
        @Override
        public int compare(Kompiuteris komp1, Kompiuteris komp2) {
            int cmp = -1;
            if(komp1.getKaina() > komp2.getKaina())
                cmp = 1;
            else if(komp1.getKaina() == komp2.getKaina())
                cmp = 0;
            return cmp;
        }
    };

    /**
     * Tuscias metodas
     * @param o the object to be compared.
     * @return
     */
    @Override
    public int compareTo(Kompiuteris o) {
        return 0;
    }
}
```

„GUI“ klasės kodas

```
package Lab2Bijeikis;

import studijosKTU.ListKTU;

import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import javax.swing.table.TableModel;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;

/**
 * Klasė darbui su GUI
 */
public class Gui extends JFrame {
   private JPanel panelMain;
   private JTable table;
   private JLabel label;
   private JMenuBar meniuBaras;
   private static ListKTU kompiuteriai;
   private DefaultTableModel tableModel;

   /**
    * Konstruktorius
    */
   public Gui() {
      initialize();
   }

   /**
    * Metodas naudojamas sukurti Gui
    */
   public void initialize(){
      setTitle("Programa");
      setContentPane(panelMain);
      meniuBaras = new JMenuBar();
      setJMenuBar(meniuBaras);
      Font f = new Font("Courier New", Font.PLAIN, 12);
      JMenu mFailai = new JMenu("Failai");
      meniuBaras.add(mFailai);
      JMenu mKomp = new JMenu("Kompiuterių apskaita");
      meniuBaras.add(mKomp);
      JMenuItem miSkaityti = new JMenuItem("Skaityti iš failo...");
      mFailai.add(miSkaityti);
      miSkaityti.addActionListener(e1 -> readFromFile(e1));
      JMenu mGene = new JMenu("Generuoti");
      JMenuItem miGene = new JMenuItem("Generuoti");
```

```
    miGene.addActionListener(new ActionListener() {
      @Override
      public void actionPerformed(ActionEvent e) {
        int amount = Integer.parseInt(JOptionPane.showInputDialog(panelMain,"","Iveskite kiekį",1));
        Generator gen = new Generator();
        gen.Generate(amount);
      }
    });
    mGene.add(miGene);
    meniuBaras.add(mGene);
    JMenuItem miBaigti = new JMenuItem("Pabaiga");
    mFailai.add(miBaigti);
    table.setFont(f);
    table.setVisible(true);


    miBaigti.addActionListener((ActionEvent e) -> System.exit(0));

    JMenuItem miGamintojas = new JMenuItem("Atranka pagal gamintoją…");
    mKomp.add(miGamintojas);
    miGamintojas.addActionListener(this::atrankaPagalGamintoja);

    JMenuItem miModelis = new JMenuItem("Atranka pagal modeli…");
    mKomp.add(miModelis);
    miModelis.addActionListener(this::atrankaPagalModeli);

    JMenuItem miKaina = new JMenuItem("Surikiuoja pagal kainą");
    mKomp.add(miKaina);
    miKaina.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent e) {
        kompiuteriai.sortBuble(Kompiuteris.pagalKaina);
        displayText("Surikiuoti Duomenys",kompiuteriai);
      }
    });

    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    setSize(800,900);
    setVisible(true);
  }

  /**
   * Metodas vykdyti atranka pagal Gamintoja
   * @param e1
   */
  private void atrankaPagalGamintoja(ActionEvent e1) {
    String filter = null;
    filter = JOptionPane.showInputDialog(panelMain,"","Iveskite Gamintoja",1);
    Atranka atr = new Atranka();
    ListKTU atrinktiKompiuteriai = atr.procesoriausGamintojas(kompiuteriai,filter);
    displayText("Atrinkti pagal gamintoja "+filter,atrinktiKompiuteriai);
  }

  /**
   * Metodas vykdyti atranka pagal Modeli
```

```
 * @param e1
 */
private void atrankaPagalModeli(ActionEvent e1) {
   String filter = null;
   filter = JOptionPane.showInputDialog(panelMain,"","Iveskite Modeli",1);
   Atranka atr = new Atranka();
   ListKTU atrinktiKompiuteriai = atr.procesoriausModelis(kompiuteriai,filter);
   displayText("Atrinkti pagal modeli "+filter,atrinktiKompiuteriai);
}

/**
 * Metodas nuskaityti domenims iš failo
 * @param e1
 */
private void readFromFile(ActionEvent e1) {
   JFileChooser fc = new JFileChooser(".");
   fc.showOpenDialog(panelMain);
   File file = new File(String.valueOf(fc.getSelectedFile()));
   try {
      Scanner sc = new Scanner(file);
      while(sc.hasNext()){
         Kompiuteris komp = new Kompiuteris();
         komp.parse(sc.nextLine());
         kompiuteriai.add(komp);
      }
   } catch (FileNotFoundException e) {
      System.out.println("Nepasirinktas joks failas");
   }
   displayText("Pradiniai Duomenys",kompiuteriai);
}

/**
 * Metodas spausdinti duomenis į lentele
 * @param antraste Antraste lenteliai
 * @param kompiuteriai duomenų klasė
 */
public void displayText(String antraste, ListKTU kompiuteriai){
   label.setText("\n"+antraste+"\n");
   tableModel = new DefaultTableModel();
   tableModel.addColumn("Gamintojas");
   tableModel.addColumn("Modelis");
   tableModel.addColumn("Ramų Kiekis");
   tableModel.addColumn("Disko vieta");
   tableModel.addColumn("Nasumas");
   tableModel.addColumn("Kaina");
   if(kompiuteriai.get(0)!=null){
      for(int i=0;i<kompiuteriai.size();i++){
         tableModel.addRow(((Kompiuteris) kompiuteriai.get(i)).toStringArray());
      }
      table.setModel(tableModel);
   }
   else label.setText("Nėra duomenų");
}
```

```
    public static void main(String[] args) {

        kompiuteriai = new ListKTU();
        Gui gui = new Gui();
    }
}
```

Atrankos metodu klasės „Atranka“ kodas

```
package Lab2Bijeikis;

import studijosKTU.ListKTU;

import java.util.Objects;

/**
 * Klasė atrankom pagal filtra atlikimui
 */
public class Atranka {
    /**
     * Konstruktorius
     */
    public Atranka(){};

    /**
     * Atranka pagal procesoriaus gamintojus
     * @param Kompiuteriai Duomenų sarasas
     * @param filtras Filtras pagal kuri atrinkti
     * @return Naujai sukurta sarara pagal filtra
     */
    public ListKTU procesoriausGamintojas(ListKTU Kompiuteriai, String filtras){
        Kompiuteris element;
        ListKTU KompListNew = new ListKTU();
        for(int i=0;i<Kompiuteriai.size();i++){
            element = (Kompiuteris) Kompiuteriai.get(i);
            if(element.getProcesoriausGamintojas().toLowerCase().compareTo(filtras.toLowerCase())==0){
                KompListNew.add(element);
            }
        }
        return KompListNew;
    }
    /**
     * Atranka pagal procesoriaus Modeli
     * @param Kompiuteriai Duomenų sarasas
     * @param filtras Filtras pagal kuri atrinkti
     * @return Naujai sukurta sarara pagal filtra
     */
    public ListKTU procesoriausModelis(ListKTU Kompiuteriai, String filtras){
        Kompiuteris element;
        ListKTU KompListNew = new ListKTU();
        for(int i=0;i<Kompiuteriai.size();i++){
            element = (Kompiuteris) Kompiuteriai.get(i);
            if(element.getProcesoriausModelis().compareTo(filtras)==0){
                KompListNew.add(element);
            }
        }
        return KompListNew;
    }
    /**
     * Atranka pagal Ramu Kieki
     * @param Kompiuteriai Duomenų sarasas
     * @param filtras Filtras pagal kuri atrinkti
```

```
     * @return Naujai sukurta sarara pagal filtra
     */

    public ListKTU ramKiekis(ListKTU Kompiuteriai, String filtras){
        Kompiuteris element;
        ListKTU KompListNew = new ListKTU();
        for(int i=0;i<Kompiuteriai.size();i++){
            element = (Kompiuteris) Kompiuteriai.get(i);
            if(element.getRamuKiekis() == filtras){
                KompListNew.add(element);
            }
        }
        return KompListNew;
    }
}
```

Klasės „Greitaveika“ kodas

```
package Lab2Bijeikis;

import studijosKTU.ListKTU;

import javax.swing.*;
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Random;
import java.util.Scanner;

/**
 * Klasė greitaveikai testuoti
 */
public class Greitaveika {
    public static void main(String[] args) {
        Generator gen = new Generator();
        gen.Generate(30000);
        ListKTU kompiuteriai = Read();
        long startTime = System.currentTimeMillis();
        Equals(kompiuteriai);
        long endTime = System.currentTimeMillis();
        System.out.printf("Equal metodo greitaveika: "+String.valueOf(endTime-startTime));

        Kompiuteris komp = new Kompiuteris("","","","",0,0);
        startTime = System.currentTimeMillis();
        kompiuteriai.Contains(komp);
        endTime = System.currentTimeMillis();
        System.out.printf("\nContains metodo greitaveika: "+String.valueOf(endTime-startTime));

        startTime = System.currentTimeMillis();
        int index = kompiuteriai.indexOf(komp);
        endTime = System.currentTimeMillis();
        System.out.printf("\nIndexOf metodo greitaveika: "+String.valueOf(endTime-startTime));

        startTime = System.currentTimeMillis();
        Remove(kompiuteriai);
        endTime = System.currentTimeMillis();
```

```
    System.out.printf("\nRemove 100 metodo greitaveika: "+String.valueOf(endTime-startTime));

    ListKTU kompClone = kompiuteriai.clone();

    startTime = System.currentTimeMillis();
    kompClone.sortInsertion(Kompiuteris.pagalGamintoja);
    endTime = System.currentTimeMillis();
    System.out.printf("\nRikiavimo metodo greitaveika: "+String.valueOf(endTime-startTime));

    kompClone = kompiuteriai.clone();

    startTime = System.currentTimeMillis();
    kompClone.sortBuble(Kompiuteris.pagalGamintoja);
    endTime = System.currentTimeMillis();
    System.out.printf("\nRikiavimo burb metodo greitaveika: "+String.valueOf(endTime-startTime));

    kompClone = kompiuteriai.clone();

    startTime = System.currentTimeMillis();
    kompClone.sortJava(Kompiuteris.pagalGamintoja);
    endTime = System.currentTimeMillis();
    System.out.printf("\nRikiavimo java metodo greitaveika: "+String.valueOf(endTime-startTime));

  }

  /**
   * Metodas equals metodui tikrinti
   * @param kompiuteriai Duomenų sarasas
   */
  public static void Equals(ListKTU kompiuteriai){
    Kompiuteris komp = new Kompiuteris("","","","",0,0);
    for(int i=0;i<kompiuteriai.size();i++){
      kompiuteriai.getNext();
      kompiuteriai.Equals(komp);
    }
  }

  /**
   * Metodas nuskaityti duomenims iš failo
   * @return Kompiuteriu duomenu sarasa
   */
  public static ListKTU Read(){
    File file = new File("@..//..//Kompiuteriai.txt");
    ListKTU kompiuteriai = new ListKTU();
    try {
      Scanner sc = new Scanner(file);
      while(sc.hasNext()){
        Kompiuteris komp = new Kompiuteris();
        komp.parse(sc.nextLine());
        kompiuteriai.add(komp);
      }
    } catch (FileNotFoundException e) {
      System.out.println("Nepasirinktas joks failas");
    }
    return kompiuteriai;
```

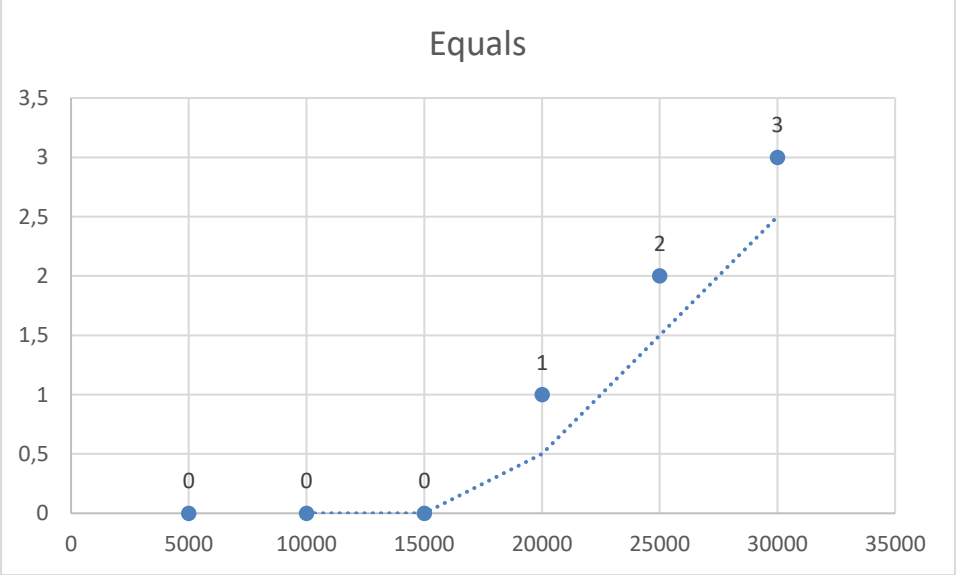
```
    }

    /**
     * Metodas tikrinti Remove metodo greitaveika
     * @param kompiuteriai Duomenų sarasas
     */
    public static void Remove(ListKTU kompiuteriai){
        Random random = new Random();
        int index;
        for(int i=0;i<100;i++){
            index = random.nextInt(kompiuteriai.size()-1);
            Kompiuteris komp = (Kompiuteris) kompiuteriai.get(index);
            ListKTU clone = kompiuteriai.clone();
            clone.Remove(komp);
        }
    }
}
```

Laiko priklausomybės nuo duomenų kiekio tyrimas

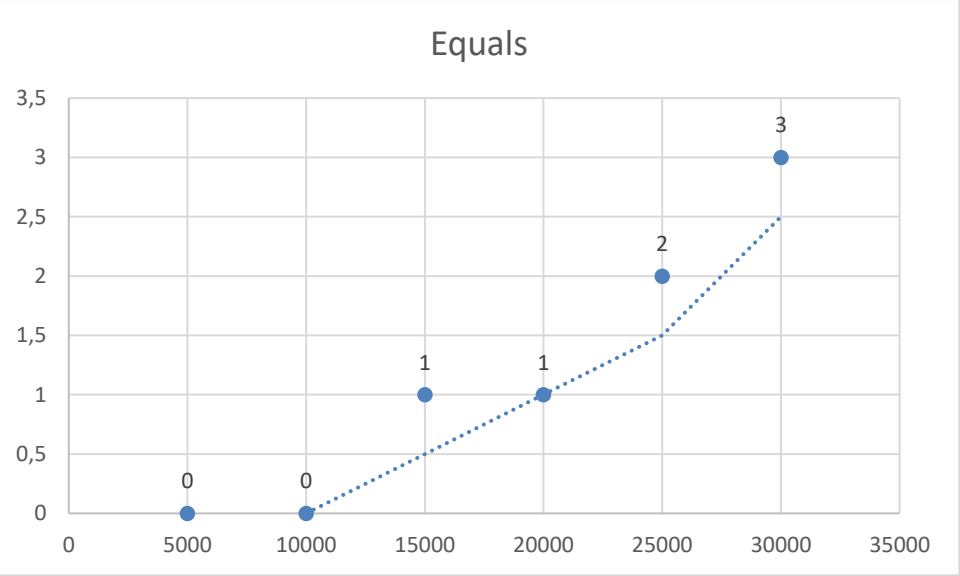
• Šis tyrimas bus atliekamas, sužinoti kaip skiriasi tam tikrų veiksmų atlikimo greitis, priklausomai nuo duomenų kiekio arba skirtingomis kompiuterio apkrovomis. Duomenys, kurie bus naudojami tyrimui ir išvadoms:

„Equals“ metodo laiko priklausomybė

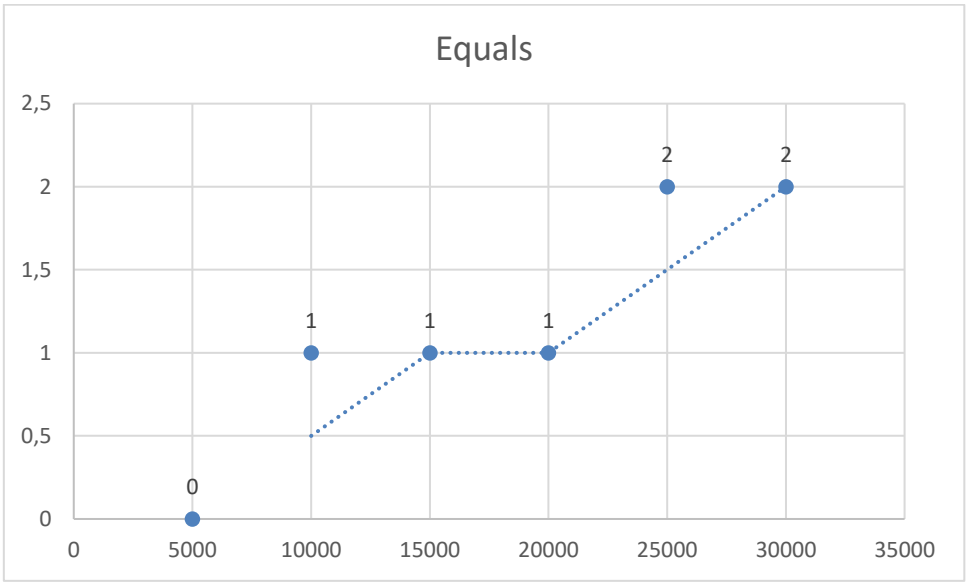
- Be papildomos apkrovos



- 50% apkrova



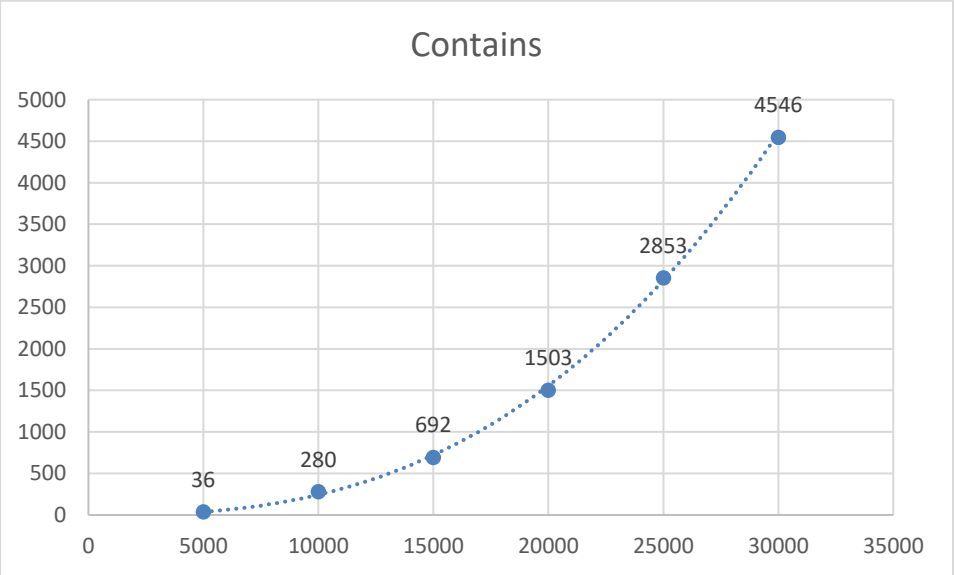
- 80% apkrova



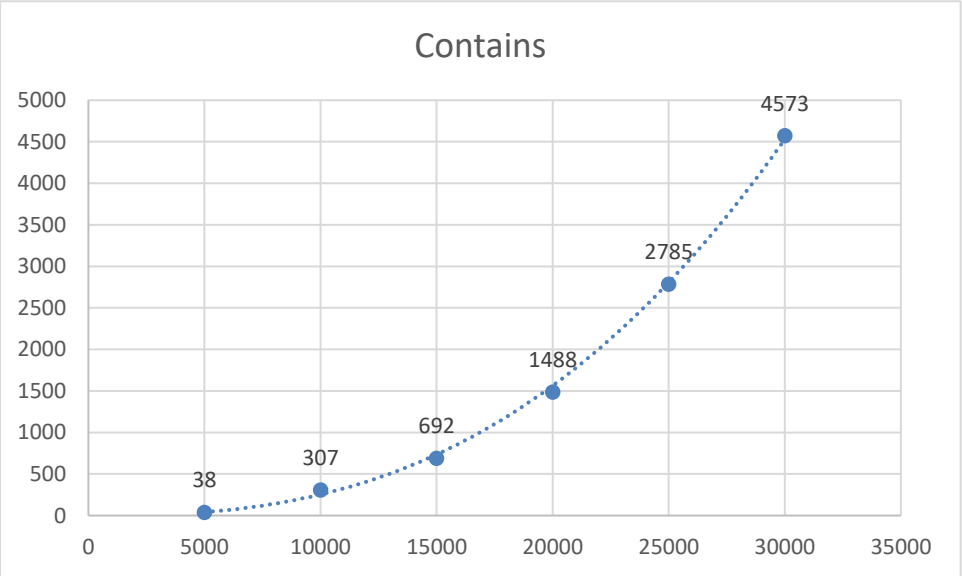
- Iš rezultatų matome, kad duomenų kiekis didelės įtakos neturi metodo atlikimo spartai

„Contains“ metodo laiko priklausomybė

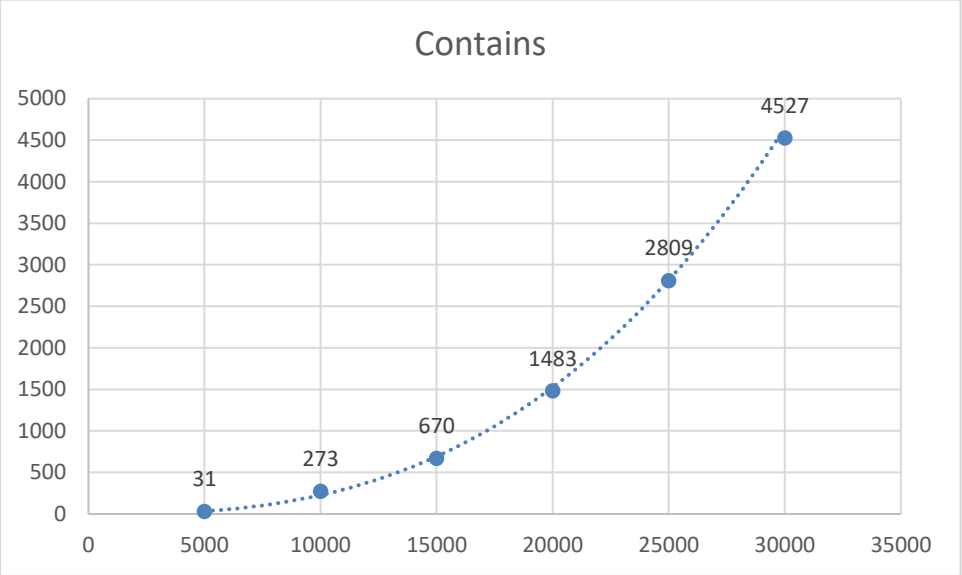
- Be apkrovos



- 50% apkrova



- 80% apkrova

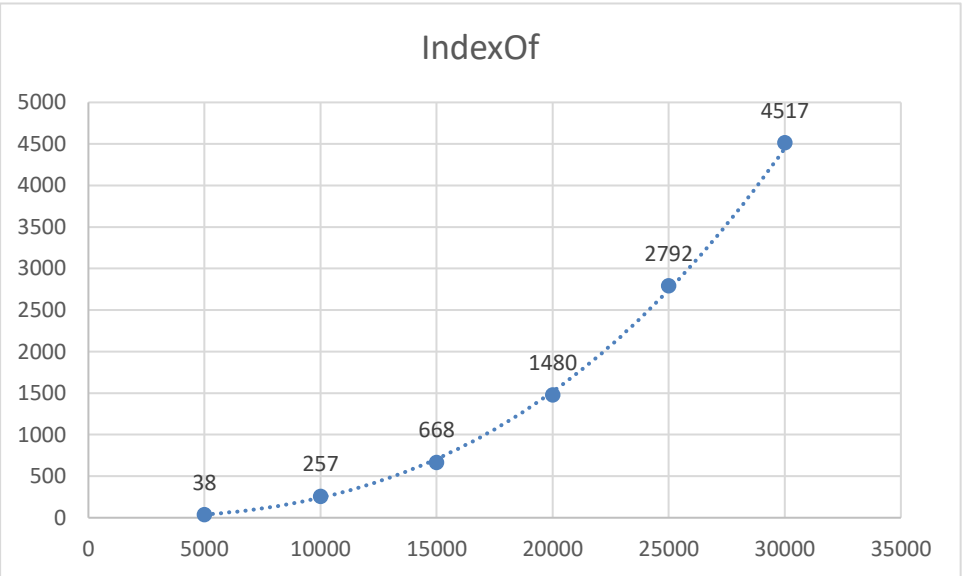


• Matome, kad rezultatai kinta laipsniškai, tarp skirtingų apkrovų duomenys skiriasi labai mažai, su apkrova buvo atliekami greičiau dėl didesnio procesoriaus dažnio.

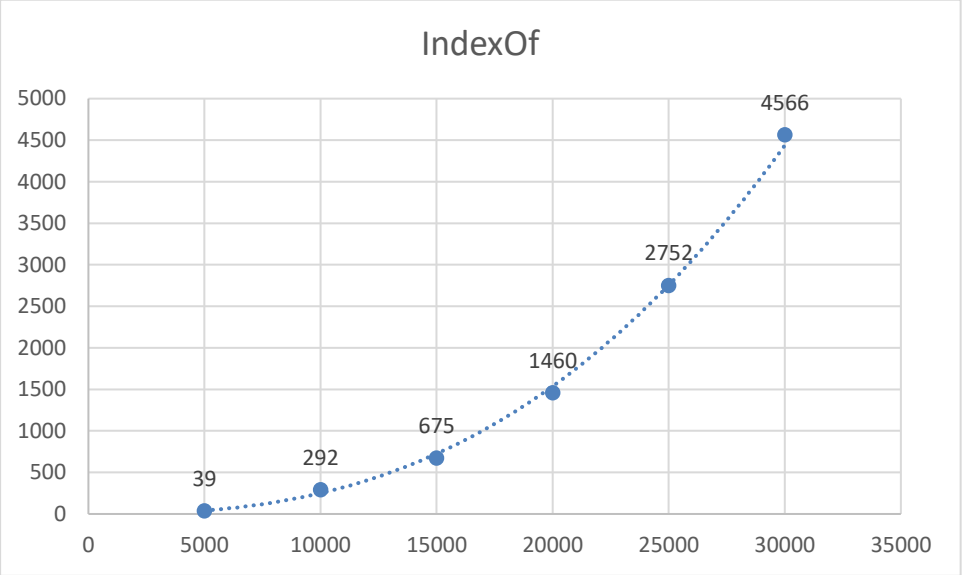
Metodo „IndexOf“ laiko priklausomybė

Laikas ms

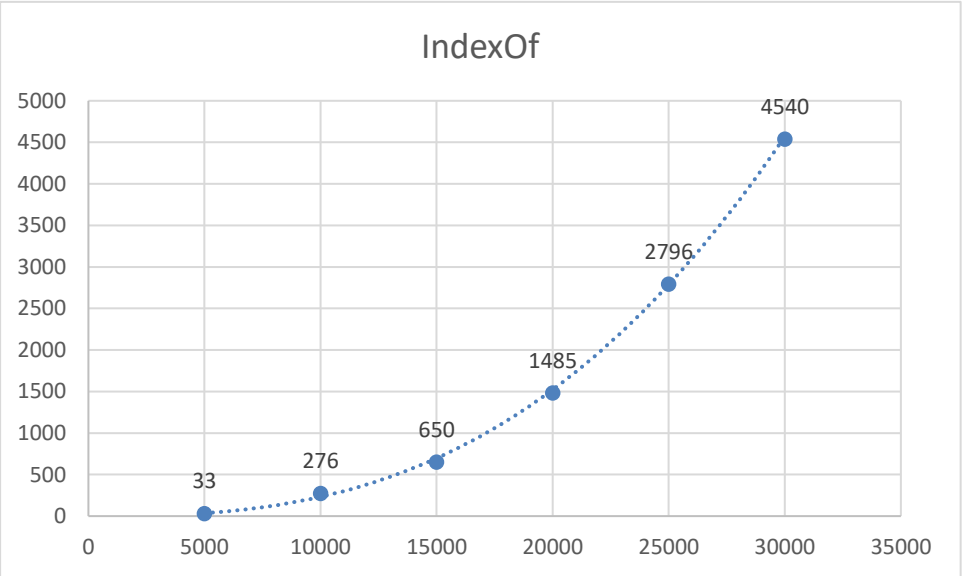
- Be apkrovos



- 50% apkrova



- 80% apkrova



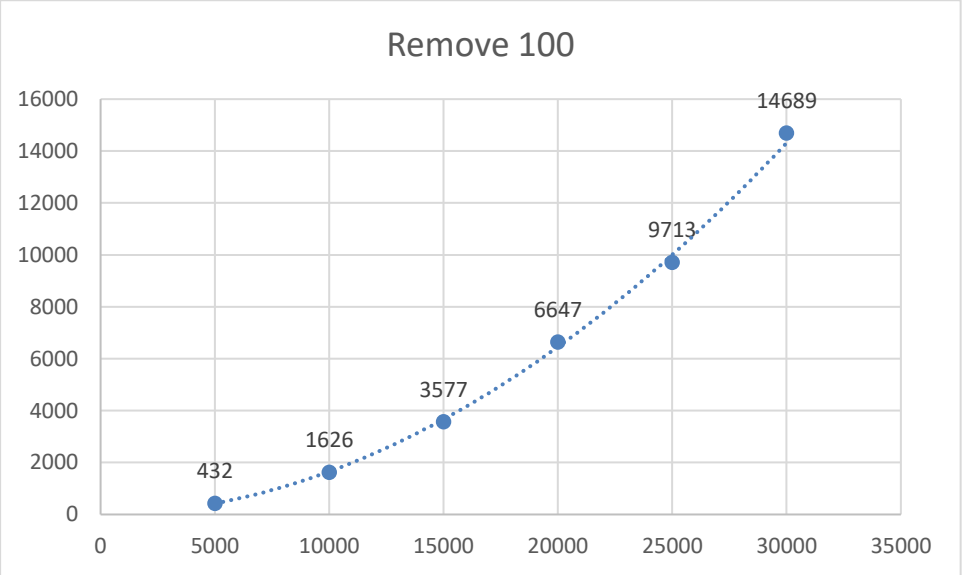
- Metodo „IndexOf‘ kodas

```
public int indexOf(Object O) {
		if (O == null) {
			for (int i = 0; i < size; i++) {
				if (get(i) == null) {
					return i;
				}
			}
		} else {
			for (int i = 0; i < size; i++) {
				if (O.equals(get(i))) {
					return i;
				}
			}
		}
		return -1;
	}
```

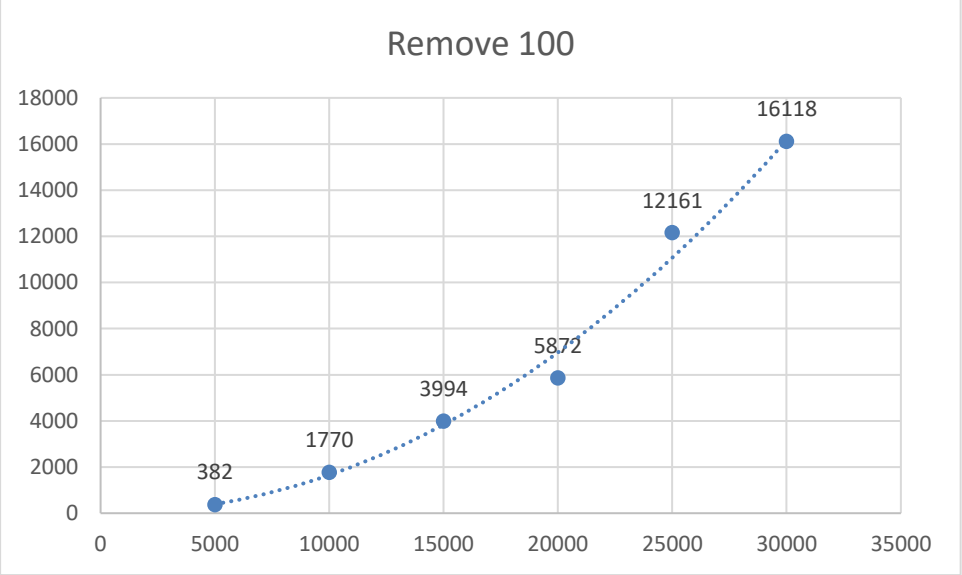
- Matome, kad rezultatai kinta laipsniškai, tarp skirtingų apkrovų duomenys skiriasi labai mažai, laiko priklausomybė buvo panaši su metodu „Contains“ todėl, kad tas metodas naudoja „IndexOf‘ metodą.

100 elementų šalinimo metodo laiko priklausomybė

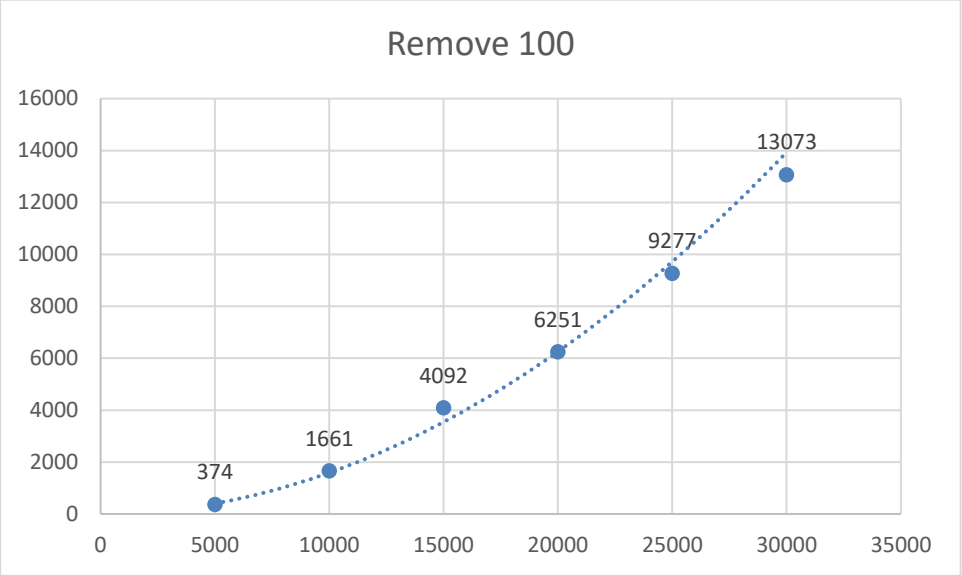
- Be apkrovos



- 50% apkrova



- 80% apkrova



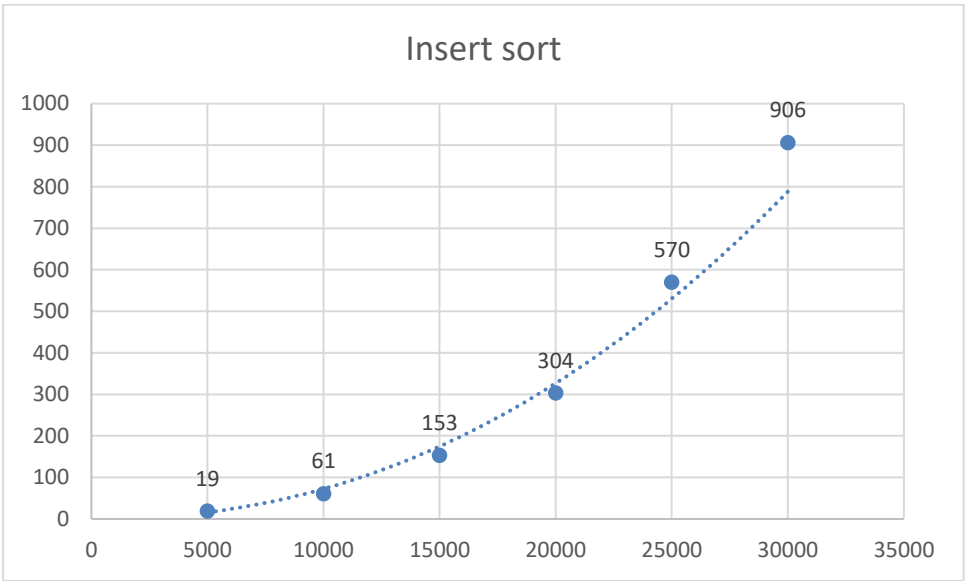
- Metodo „Remove“ kodas

```
public boolean Remove(Object O) {
		for(int i=0;i<size; i++){
			if(O.equals(get(i))){
				for(int j=i;j<size-1;j++) {
					if(current.next!=null)
						current.element = current.next.element;
					if(j+1==size){
						current.next = null;
					}
				}
				return true;
			}
		}
		return false;
	}
```

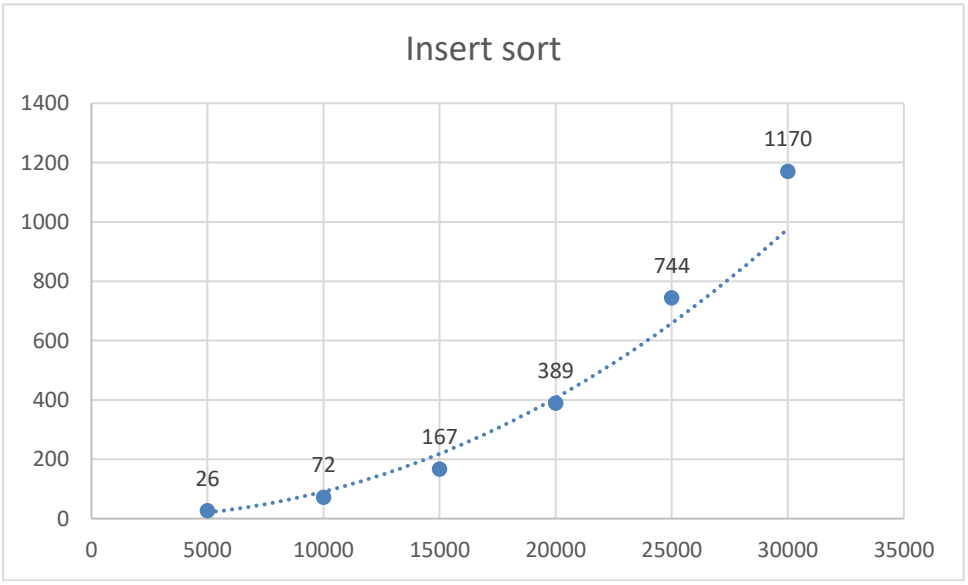
- Matome, kad rezultatai kinta laipsniškai, tarp skirtingų apkrovų duomenys labiau skiriasi, su 80% apkrova metodas veikė greičiausiai dėl pakilusio procesoriaus greičio.

„InsertSort“ metodo laiko priklausomybė

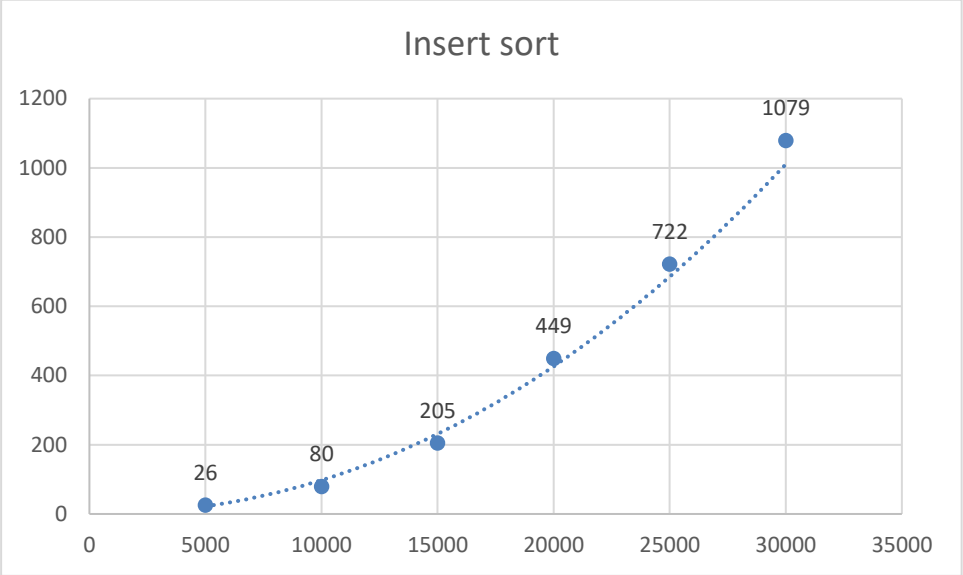
- Be apkrovos



- 50% apkrova



- 80% apkrova



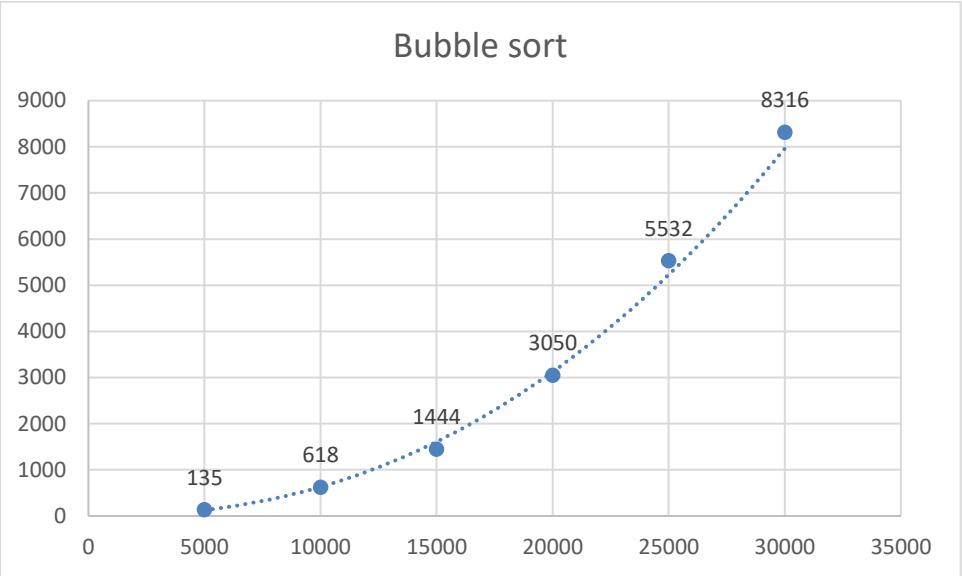
- Metodo „InsertSort“ kodas

```
public void sortInsertion(Comparator<E> c) {
		if (first == null) {
			return;
		}

		Node<E> sorted = null;

		Node<E> current = first;

		while (current != null) {
			Node<E> next = current.next;
			if (sorted == null || c.compare(sorted.element, current.element) >= 0) {
				current.next = sorted;
				sorted = current;
			} else {
				Node<E> search = sorted;
				while (search.next != null &&
c.compare(search.next.element,current.element) < 0) {
					search = search.next;
				}
				current.next = search.next;
				search.next = current;
			}
			current = next;
		}

		first = sorted;
	}
```

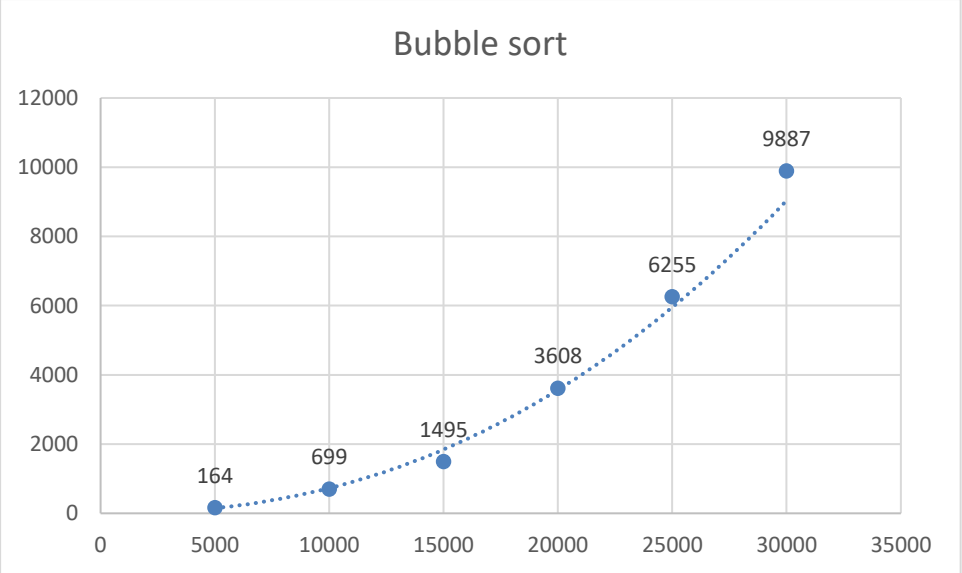
- Matome, kad rezultatai kinta laipsniškai, tarp skirtingų apkrovų laikas skiriasi.

„BubbleSort“ metodo laiko priklausomybė

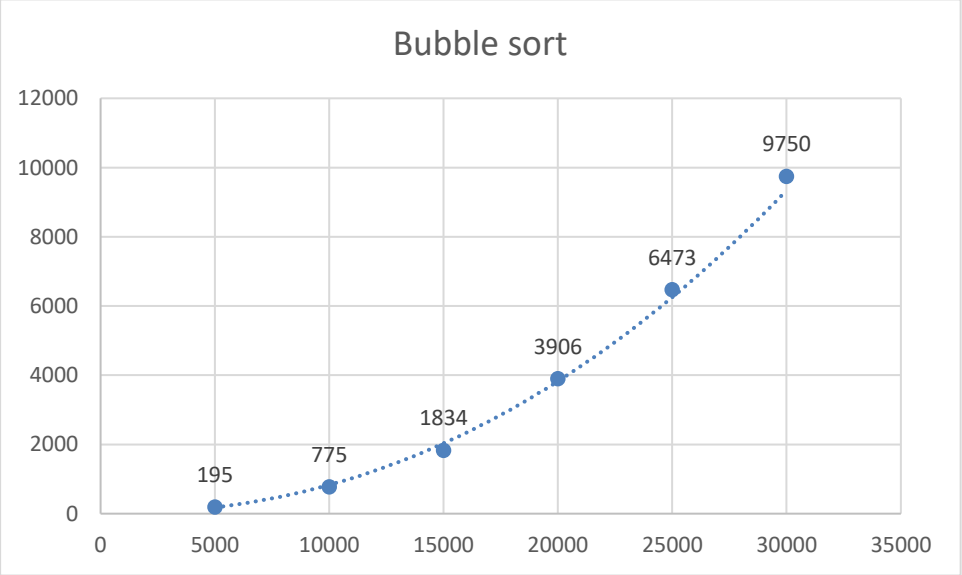
- Be apkrovos



- 50% apkrova



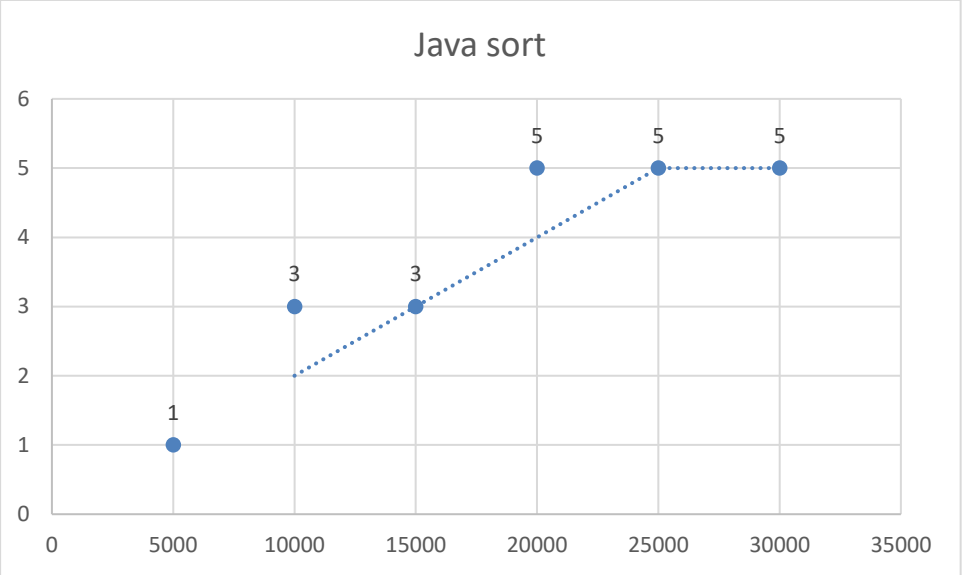
- 80% apkrova



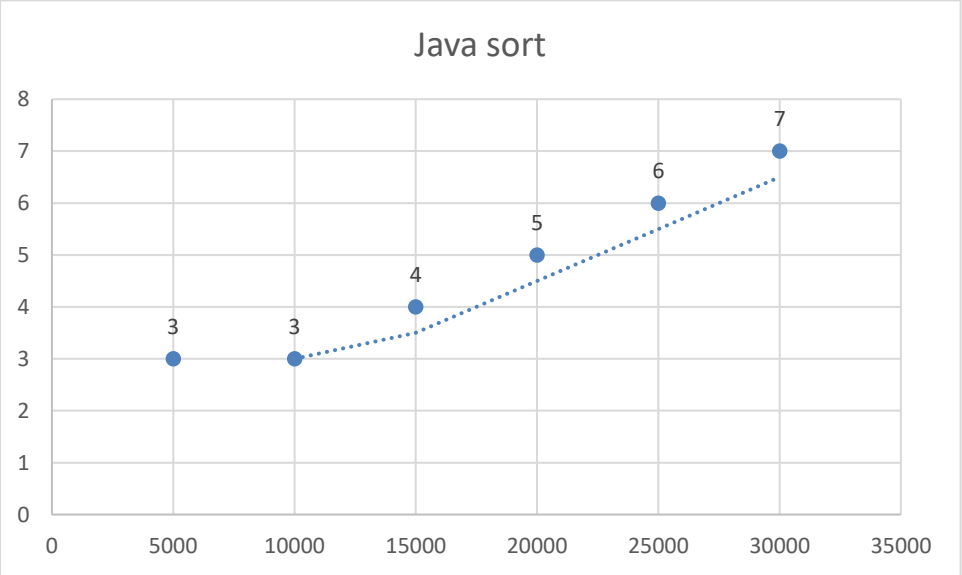
• Matome, kad rezultatai kinta laipsniškai, tarp be apkrovos ir su apkrova atliktų testų greičiai skiriasi per 1500 milisekundžių, kas reiškia, kad šis metodas reikalauja ne mažo kiekio resursų.

„JavaSort“ metodo laiko priklausomybė

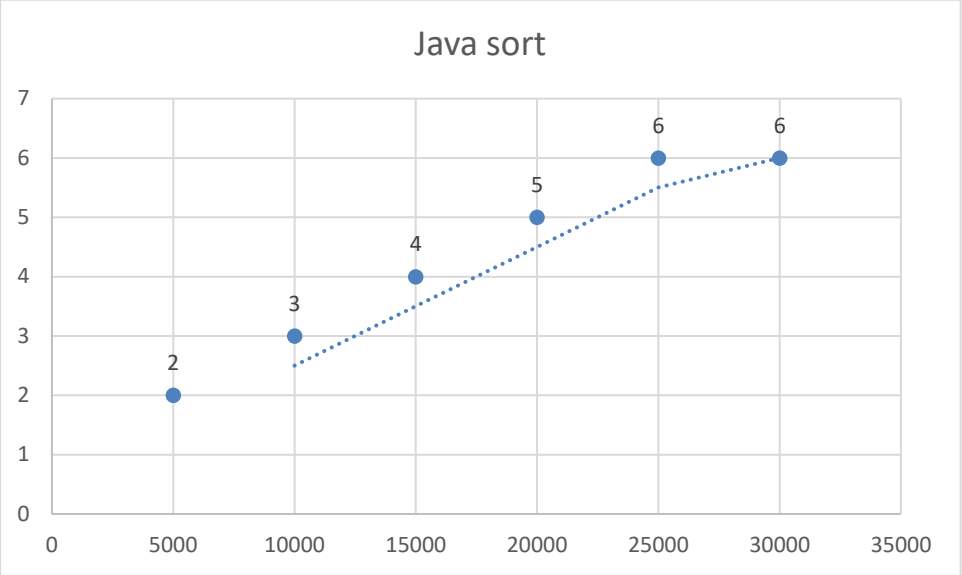
- Be akprovos



- 50% apkrova



- 80% apkrova



- Matome, kad rezultatai kinta tiesiškai, metodas tarp skirtingų apkrovų veikia panašių greičiu, kas reiškia, kad yra efektyviausias iš visų trijų rikiavimo metodų

Viso tyrimo išvados

Apibendrinant, ištyrėme duotos „ListKTU“ sąrašo klasės, bei savo sukurtų metodų veikimo spartą priklausomai nuo duomenų kiekio, bei kompiuterio apkrovos, galėjome pastebėti, kad efektyviausias rikiavimo metodas yra „JavaSort“ ir labiausiai neefektyvus yra „BubbleSort“.