Лабораторна робота №14

Паралельне виконання. Ефективність використання

Мета: Вимірювання часу паралельних та послідовних обчислень. Демонстрація ефективності паралельної обробки.

1 ВИМОГИ

1. Забезпечити вимірювання часу паралельної обробки елементів контейнера за допомогою розроблених раніше методів.
2. Додати до алгоритмів штучну затримку виконання для кожної ітерації циклів поелементної обробки контейнерів, щоб загальний час обробки був декілька секунд.
3. Реалізувати послідовну обробку контейнера за допомогою методів, що використовувались для паралельної обробки та забезпечити вимірювання часу їх роботи.
4. Порівняти час паралельної і послідовної обробки та зробити висновки про ефективність розпаралелювання:
   - результати вимірювання часу звести в таблицю;
   - обчислити та продемонструвати у скільки разів паралельне виконання швидше послідовного.

1.1 Розробник

- П.І.Б: Заночкин Є. Д.
- Группа: КІТ-119а
- Варіант: 7

2 ОПИС ПРОГРАМИ

2.1 Засоби ООП:

Scanner inInt, inStr = new Scanner(System.in) – для введення обраних опцій користувачем з клавіатури;

XMLEncoder encoder = new XMLEncoder(new BufferedOutputStream(new FileOutputStream("Lab14.xml"));

encoder.writeObject(container); – нестандартна серіалізація;

XMLDecoder decoder = new XMLDecoder(new BufferedInputStream(new FileInputStream("Lab14.xml")));

container = (ClientList<Client>) decoder.readObject(); – нестандартна десеріалізація;

ObjectOutputStream oos = new ObjectOutputStream(new BufferedOutputStream(newFileOutputStream("Lab14.ser")));

oos.writeObject(container);

oos.flush(); – стандартна серіалізація;

ObjectInputStream ois = new ObjectInputStream(new BufferedOutputStream(new FileInputStream("Lab14.ser")));

container = (ClientList<Client>) ois.readObject(); – стандартна десеріалізація;

Pattern pattern = Pattern.compile() – компілює регулярний вираз у шаблон;

Matcher matcher = pattern.matcher(data); – створює matcher, який буде відповідати даному вводу для цього шаблону.

2.2 Ієрархія та структура класів

Було створено класи Main (головний клас програми), ClientList (клас-контейнер), 4 класи, що реалізують інтерфейс Comparator для сортування за певними критеріями, клас MyThread (реалізує інтерфейс Runnable для роботи з потоками), а також підключено класи з попередньої роботи: Client, InfoAboutYourself, PartnerRequirements та Node.

2.3 Важливі фрагменти програми

Клас Main

```
package ua.khpi.oop.zanochkyn14;

import java.beans.XMLDecoder;
import java.beans.XMLEncoder;
import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.GregorianCalendar;
import java.util.Scanner;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
import ua.khpi.oop.zanochkyn10.Client;
import ua.khpi.oop.zanochkyn10.InfoAboutYourself;
import ua.khpi.oop.zanochkyn10.PartnerRequirements;

public class Main
{
	public static void main(String[] args)
	{
		ClientList<Client> container = new ClientList<Client>();
		for(String str: args)
		{
			if(str.equals("-a") || str.equals("-auto"))
			{
				auto(container);
				return;
			}
			else if(str.equals("-d") || str.equals("-dialog"))
			{
				menu(container);
				return;
			}
		}
		menu(container);
	}

	private static void auto(ClientList<Client> container)
	{
		System.out.println("Size of container: " + container.getSize());
		System.out.println("\nAdding elements...");
		File file = new File("Lab12-data.txt");
		int countClientHobbies, countPartnerHobbies;
		String[] clientHobbies, partnerHobbies;
		GregorianCalendar date;
		InfoAboutYourself info;
		PartnerRequirements requirements;
		try
		{
			Scanner reader = new Scanner(file);
			while (reader.hasNextLine())
			{
		          String data = reader.nextLine();
		          Pattern   pattern   =   Pattern.compile("^((Male|Female),\\s([a-zA-Z]+),\\s(([1-9])|([1-
9][0-9])),\\s(([1-9])|([1-9][0-9])|([1-2][0-9][0-9])),\\s([a-zA-Z]+),\\s([0-4]),\\s" +
				"([a-zA-Z]+|[a-zA-Z]+\\s[a-zA-Z]+)(,\\s([a-zA-Z]+|[a-zA-Z]+\\s[a-zA-
Z]+))*,\\s(Male|Female),\\s(([1-9])|([1-9][0-9])),\\s(([1-9])|([1-9][0-9])),\\s([0-4]),\\s" +
				"([a-zA-Z]+|[a-zA-Z]+\\s[a-zA-Z]+)(,\\s([a-zA-Z]+|[a-zA-Z]+\\s[a-zA-
Z]+))*)");
```

```
                Matcher matcher = pattern.matcher(data);
              if (matcher.matches())
              {
                String[] tmp = data.split(",\\s");
                    if(Integer.parseInt(tmp[5]) == 0)
                    {
                        countClientHobbies = 0;
                            clientHobbies = new String[countClientHobbies];
                    }
                    else
                    {
                        countClientHobbies = Integer.parseInt(tmp[5]);
                        clientHobbies = new String[countClientHobbies];
                        for (int i = 6, j = 0; i < 6 + countClientHobbies; i++, j++)
                            clientHobbies[j] = tmp[i];
                    }
                    if(Integer.parseInt(tmp[9 + countClientHobbies]) == 0)
                    {
                        countPartnerHobbies = 0;
                        partnerHobbies = new String[countPartnerHobbies];
                    }
                    else
                    {
                        if(countClientHobbies == 0)
                        {
                            countPartnerHobbies = Integer.parseInt(tmp[9 + 1]);
                            partnerHobbies = new String[countPartnerHobbies];
                            if(countPartnerHobbies != 0)
                                for (int i = 9 + 1 + 1, j = 0; i < tmp.length;
i++, j++)
                                    partnerHobbies[j] = tmp[i];
                        }
                        else
                        {
                            countPartnerHobbies = Integer.parseInt(tmp[9 +
countClientHobbies]);
                            partnerHobbies = new String[countPartnerHobbies];
                            for (int i = 9 + countClientHobbies + 1, j = 0; i <
tmp.length; i++, j++)
                                partnerHobbies[j] = tmp[i];
                        }
                    }
                    info = new InfoAboutYourself(tmp[1], Integer.parseInt(tmp[2]),
Integer.parseInt(tmp[3]), tmp[4], clientHobbies);
                    int pos;
                    if(countClientHobbies == 0)
                        pos = 7;
                    else
                        pos = countClientHobbies + 6;
                    requirements = new PartnerRequirements(tmp[pos],
Integer.parseInt(tmp[pos+1]), Integer.parseInt(tmp[pos+2]), partnerHobbies);
                    date = new GregorianCalendar();
                    container.add(new Client(tmp[0], indexGenerator(container), date, info,
requirements));
              }
             }
                reader.close();
            }
            catch (FileNotFoundException e)
            {
                e.printStackTrace();
            }
            System.out.println("Elements added.");
            System.out.println("\nSize of container: " + container.getSize());
            System.out.println("\nOutput the container...");
            System.out.println("\n" + container.toString());
            Pattern patternAgeDifference = Pattern.compile("^([0-5])");
            Pattern patternHobby = Pattern.compile("^(Morning runs)");
            Pattern patternMale = Pattern.compile("^(Male)");
            Pattern patternFemale = Pattern.compile("^(Female)");
            Matcher matcherHobby1, matcherHobby2, matcherAge, matcherGenderMale,
matcherGenderFemale;
            ArrayList<Integer> positions = new ArrayList<>();
            boolean hobbyCheck1 = false, foundCouple = false;
            System.out.println("Finding all combinations of couples with heterosexual partners with
an age difference of no more than 5 years for morning runs...\n");
            for(int i = 0; i < container.getSize(); i++)
            {
                clientHobbies = container.getElement(i).getInformation().getClientHobby();
                partnerHobbies = container.getElement(i).getRequirements().getPartnerHobby();
                if(clientHobbies.length != 0 && partnerHobbies.length != 0)
                {
                    for(int a = 0; a < clientHobbies.length; a++)
                    {
                        matcherHobby1 = patternHobby.matcher(clientHobbies[a]);
                        if(matcherHobby1.matches())
                        {
                            hobbyCheck1 = true;
                            break;
                        }
                    }
                    if(hobbyCheck1 == true)
                        for(int b = 0; b < partnerHobbies.length; b++)
                        {
                            matcherHobby2 =
patternHobby.matcher(partnerHobbies[b]);
                            if(matcherHobby2.matches())
                                positions.add(i);
                        }
                }
            }
            int num = 1;
            if(!positions.isEmpty())
                for(int i = 0; i < container.getSize(); i++)
                {
                    if(positions.contains(i))
                        for(int j = i + 1; j < container.getSize(); j++)
                            if(positions.contains(j))
                            {
                                int ageDifference =
Math.abs(container.getElement(i).getInformation().getAge() - container.getElement(j).getInformation().getAge());
                                matcherAge =
patternAgeDifference.matcher(Integer.toString(ageDifference));
                                if(matcherAge.matches())
                                {
                                    matcherGenderMale =
patternMale.matcher(container.getElement(i).getClientGender());
                                    if(matcherGenderMale.matches())
                                    {
                                        matcherGenderFemale        =
patternFemale.matcher(container.getElement(j).getClientGender());

    if(matcherGenderFemale.matches())
                                        {

    System.out.println("Couple   "   +   num   +   ":\n"   +   container.getElement(i).toString()   +   "\n"   +
container.getElement(j).toString() + "\n");
                                            foundCouple        =
true;

                                            num++;
                                        }
                                    }
                                    else
                                    {
                                        matcherGenderMale          =
patternMale.matcher(container.getElement(j).getClientGender());

    if(matcherGenderMale.matches())
                                        {

    System.out.println("Couple   "   +   num   +   ":\n"   +   container.getElement(i).toString()   +   "\n"   +
container.getElement(j).toString() + "\n");
                                            foundCouple        =
true;

                                            num++;
                                        }
                                    }

                                }
                            }
                }
            if(foundCouple != true)
                System.out.println("There is no matching couples.");
            System.out.println("End.");
        }

        private static void menu(ClientList<Client> container)
        {
            String gender = "";
            String partnerGender;
            String name;
            GregorianCalendar date;
            InfoAboutYourself info;
            PartnerRequirements requirements;
            Pattern patternName = Pattern.compile("^([a-zA-Z]+)");
            Pattern patternAge = Pattern.compile("^(([1-9])|([1-9][0-9]))");
            Pattern patternHeight = Pattern.compile("^(([1-9])|([1-9][0-9])|([1-2][0-9][0-9]))");
            Pattern patternEyeColour = Pattern.compile("^([a-zA-Z]+)");
            Pattern patternHobby = Pattern.compile("^[a-zA-Z]+|[a-zA-Z]+\\s[a-zA-Z]+");
            boolean endCheck = true;
            Scanner inInt = new Scanner(System.in);
            Scanner inStr = new Scanner(System.in);
            while (endCheck)
            {
                System.out.println("Menu:");
                System.out.println("1. Show clients");
                System.out.println("2. Add client");
                System.out.println("3. Remove client");
                System.out.println("4. Change information");
				System.out.println("5. Clear list");
				System.out.println("6. Serialize data");
				System.out.println("7. Deserialize data");
				System.out.println("8. Count elements in a container");
				System.out.println("9. Sort the container");
				System.out.println("10. Finding all combinations of couples with heterosexual
partners with some age difference for morning runs");
				System.out.println("11. Threads task");
				System.out.println("0. Exit");
				System.out.println("Enter your option:");
				int option = inInt.nextInt();
				System.out.println();
				switch (option)
				{
				case 1:
					if(container.getSize() > 0)
						System.out.println(container.toString());
					else
						System.out.println("Container is empty.\n");
					break;
				case 2:
					System.out.println("Choose gender:\n1. Male\n2. Female");
					int genderOption = inInt.nextInt();
					if(genderOption == 1)
					{
						gender = "Male";
						partnerGender = "Female";
					}
					else
					{
						gender = "Female";
						partnerGender = "Male";
					}
					System.out.println("\nEnter information about yourself");
					System.out.println("Name:");
					name = inStr.nextLine();
					name = stringRegexCheck(name, patternName);
					System.out.println("Age:");
					int age = inInt.nextInt();
					age = intRegexCheck(age, patternAge);
					System.out.println("Height:");
					int height = inInt.nextInt();
					height = intRegexCheck(height, patternHeight);
					System.out.println("Eye colour:");
					String eyeColour = inStr.nextLine();
					eyeColour = stringRegexCheck(eyeColour, patternEyeColour);
					System.out.println("Enter count of client's hobbies:");
					int countClientHobbies = inInt.nextInt();
					String[] clientHobbies = new String[countClientHobbies];
					if(countClientHobbies != 0)
					{
						System.out.println("Enter client's hobbies (max 2 words):");
						for(int i = 0; i < countClientHobbies; i++)
						{
							String hobby = inStr.nextLine();
							hobby = stringRegexCheck(hobby, patternHobby);
							clientHobbies[i] = hobby;
						}
					}
```

```
                    info = new InfoAboutYourself(name, age, height, eyeColour,
clientHobbies);
                    System.out.println("\nEnter partner requirements");
                    System.out.println("Min age:");
                    int minAge = inInt.nextInt();
                    minAge = intRegexCheck(minAge, patternAge);
                    System.out.println("Max age:");
                    int maxAge = inInt.nextInt();
                    maxAge = intRegexCheck(maxAge, patternAge);
                    System.out.println("Enter count of partner's hobbies:");
                    int countPartnerHobbies = inInt.nextInt();
                    String[] partnerHobbies = new String[countPartnerHobbies];
                    if(countPartnerHobbies != 0)
                    {
                        System.out.println("Enter partner's hobbies (max 2 words):");
                        for(int i = 0; i < countPartnerHobbies; i++)
                        {
                            String hobby = inStr.nextLine();
                            hobby = stringRegexCheck(hobby, patternHobby);
                            partnerHobbies[i] = hobby;
                        }
                    }
                    requirements = new PartnerRequirements(partnerGender, minAge,
maxAge, partnerHobbies);
                    date = new GregorianCalendar();
                    container.add(new Client(gender, indexGenerator(container), date, info,
requirements));
                    System.out.println("\n" + container.toString());
                    break;
                case 3:
                    System.out.println("Enter client's ID to remove him:");
                    int id = inInt.nextInt();
                    int size = container.getSize();
                    for(int i = 0; i < container.getSize(); i++)
                        if(container.getElement(i).getId() == id)
                        {
                            container.remove(i);
                            break;
                        }
                    if(size == container.getSize())
                        System.out.println("\nThere is no such client");
                    else
                        System.out.println("\nClient removed");
                    System.out.println();
                    break;
                case 4:
                    System.out.println("Enter client's ID to change his information:");
                    id = inInt.nextInt();
                    int index = 0;
                    for(index = 0; index < container.getSize(); index++)
                        if(container.getElement(index).getId() == id)
                            break;
                    if(index == container.getSize())
                    {
                        System.out.println("\nThere is no client with that ID.\n");
                        break;
                    }
                    boolean endCheck2 = true;
                    int option2 = 0;
                    while(endCheck2)
```

```
				{
					System.out.println("\n"						+
container.getElement(index).toString() + "\n");
					System.out.println("Which information you want to change?");
					System.out.println("1. Gender");
					System.out.println("2. ID");
					System.out.println("3. Registration date");
					System.out.println("4. Information about yourself");
					System.out.println("5. Partner requirements");
					System.out.println("6. End of change");
					System.out.println("Enter option:");
					option2 = inInt.nextInt();
					switch(option2)
					{
					case 1:
						if(container.getElement(index).getClientGender()	==
"Male")

	container.getElement(index).setClientGender("Female");
						else

	container.getElement(index).setClientGender("Male");
						break;
					case 2:
						System.out.println("\nEnter new ID (e.g. 10):");
						container.getElement(index).setId(inInt.nextInt());
						break;
					case 3:
						Pattern			patternYear			=
Pattern.compile("^(?!^0)\\d{4}$");

						Pattern	patternMonth	=	Pattern.compile("^(([1-
9])|([1][0-2]))");

						Pattern	patternDay	=	Pattern.compile("^(([1-
9])|([12][0-9])|([3][01]))");

						Pattern	patternHour	=	Pattern.compile("^(([0-
9])|([1][0-9])|([2][0-4]))");

						Pattern	patternMinute	=	Pattern.compile("^(([0-
9])|([1-5][0-9])|([6][0]))");

						GregorianCalendar	newDate		=	new
GregorianCalendar();

						System.out.println("\nEnter registration year:");
						int value = inInt.nextInt();
						value = intRegexCheck(value, patternYear);
						newDate.set(Calendar.YEAR, value);
						System.out.println("Enter registration month:");
						value = inInt.nextInt();
						value = intRegexCheck(value, patternMonth);
						newDate.set(Calendar.MONTH, value-1);
						System.out.println("Enter registration day:");
						value = inInt.nextInt();
						value = intRegexCheck(value, patternDay);
						newDate.set(Calendar.DAY_OF_MONTH, value);
						System.out.println("Enter registration hour:");
						value = inInt.nextInt();
						value = intRegexCheck(value, patternHour);
						newDate.set(Calendar.HOUR_OF_DAY, value);
						System.out.println("Enter registration minute:");
						value = inInt.nextInt();
						value = intRegexCheck(value, patternMinute);
						newDate.set(Calendar.MINUTE, value);
```

```
                                                newDate.set(Calendar.SECOND, 0);
                                                container.getElement(index).setDate(newDate);
                                                break;
                                        case 4:
                                                System.out.println("\nInformation about yourself:");
                                                System.out.println("1. Name");
                                                System.out.println("2. Age");
                                                System.out.println("3. Height");
                                                System.out.println("4. Eye colour");
                                                System.out.println("5. Hobbies");
                                                System.out.println("6. Change all information");
                                                System.out.println("Enter option:");
                                                int option3 = inInt.nextInt();
                                                System.out.println();
                                                switch(option3)
                                                {
                                                case 1:
                                                        System.out.println("Enter new name:");
                                                        name = inStr.nextLine();
                                                        name = stringRegexCheck(name,
patternName);

        container.getElement(index).getInformation().setName(name);
                                                        break;
                                                case 2:
                                                        System.out.println("Enter new age:");
                                                        age = inInt.nextInt();
                                                        age = intRegexCheck(age, patternAge);

        container.getElement(index).getInformation().setAge(age);
                                                        break;
                                                case 3:
                                                        System.out.println("Enter new height:");
                                                        height = inInt.nextInt();
                                                        height = intRegexCheck(height,
patternHeight);

        container.getElement(index).getInformation().setHeight(height);
                                                        break;
                                                case 4:
                                                        System.out.println("Enter new eye colour:");
                                                        eyeColour = inStr.nextLine();
                                                        eyeColour = stringRegexCheck(eyeColour,
patternEyeColour);

        container.getElement(index).getInformation().setEyeColour(eyeColour);
                                                        break;
                                                case 5:
                                                        System.out.println("Enter new count of
client's hobbies:");
                                                        countClientHobbies = inInt.nextInt();
                                                        clientHobbies = new
String[countClientHobbies];
                                                        if(countClientHobbies != 0)
                                                        {
                                                                System.out.println("Enter client's
hobbies (max 2 words):");
                                                                for(int i = 0; i < countClientHobbies;
i++)
                                                                {
```

```
                                        String hobby =
inStr.nextLine();
                                        hobby =
stringRegexCheck(hobby, patternHobby);
                                        clientHobbies[i] = hobby;
                                    }
                                }

    container.getElement(index).getInformation().setClientHobby(clientHobbies);
                                break;
                            case 6:
                                System.out.println("Enter new name:");
                                name = inStr.nextLine();
                                name = stringRegexCheck(name,
patternName);
                                System.out.println("Enter new age:");
                                age = inInt.nextInt();
                                age = intRegexCheck(age, patternAge);
                                System.out.println("Enter new height:");
                                height = inInt.nextInt();
                                height = intRegexCheck(height,
patternHeight);
                                System.out.println("Enter new eye colour:");
                                eyeColour = inStr.nextLine();
                                eyeColour = stringRegexCheck(eyeColour,
patternEyeColour);
                                System.out.println("Enter new count of
client's hobbies:");
                                countClientHobbies = inInt.nextInt();
                                clientHobbies = new
String[countClientHobbies];
                                if(countClientHobbies != 0)
                                {
                                    System.out.println("Enter client's
hobbies (max 2 words):");
                                    for(int i = 0; i < countClientHobbies;
i++)
                                    {
                                        String hobby =
inStr.nextLine();
                                        hobby =
stringRegexCheck(hobby, patternHobby);
                                        clientHobbies[i] = hobby;
                                    }
                                }
                                info = new InfoAboutYourself(name, age,
height, eyeColour, clientHobbies);

    container.getElement(index).setInformation(info);
                                break;
                            default:
                                System.out.println("Wrong command.");
                                break;
                            }
                            break;
                        case 5:
                            System.out.println("\nPartner requirements:");
                            System.out.println("1. Gender");
                            System.out.println("2. Min age");
                            System.out.println("3. Max age");
```

```
                                        System.out.println("4. Hobbies");
                                        System.out.println("5. Change all requirements");
                                        System.out.println("Enter option:");
                                        option3 = inInt.nextInt();
                                        switch(option3)
                                        {
                                        case 1:

        if(container.getElement(index).getRequirements().getPartnerGender() == "Male")

        container.getElement(index).getRequirements().setPartnerGender("Female");
                                            else

        container.getElement(index).getRequirements().setPartnerGender("Male");
                                            break;
                                        case 2:
                                            System.out.println("\nEnter new min age:");
                                            minAge = inInt.nextInt();
                                            minAge      =       intRegexCheck(minAge,
    patternAge);

        container.getElement(index).getRequirements().setMinAge(minAge);
                                            break;
                                        case 3:
                                            System.out.println("\nEnter new max age:");
                                            maxAge = inInt.nextInt();
                                            maxAge      =       intRegexCheck(maxAge,
    patternAge);

        container.getElement(index).getRequirements().setMaxAge(maxAge);
                                            break;
                                        case 4:
                                            System.out.println("\nEnter   new   count   of
    partner's hobbies:");

                                            countPartnerHobbies = inInt.nextInt();
                                            partnerHobbies            =            new
    String[countPartnerHobbies];

                                            {
                                                System.out.println("Enter    partner's
    hobbies (max 2 words):");

                                                for(int    i    =    0;    i    <
    countPartnerHobbies; i++)

                                                {
                                                    String       hobby       =
    inStr.nextLine();

                                                    hobby                   =
    stringRegexCheck(hobby, patternHobby);

                                                    partnerHobbies[i] = hobby;
                                                }
                                            }

        container.getElement(index).getRequirements().setPartnerHobby(partnerHobbies);
                                            break;
                                        case 5:

        if(container.getElement(index).getRequirements().getPartnerGender() == "Male")
                                                partnerGender = "Female";
                                            else
                                                partnerGender = "Male";
                                            System.out.println("\nEnter new min age:");
```

```
                                minAge = inInt.nextInt();
                                minAge = intRegexCheck(minAge,
patternAge);

                                System.out.println("Enter new max age:");
                                maxAge = inInt.nextInt();
                                maxAge = intRegexCheck(maxAge,
patternAge);

                                System.out.println("Enter new count of
partner's hobbies:");

                                countPartnerHobbies = inInt.nextInt();
                                partnerHobbies = new
String[countPartnerHobbies];

                                {
                                    System.out.println("Enter partner's
hobbies (max 2 words):");

                                    for(int i = 0; i <
countPartnerHobbies; i++)

                                    {
                                        String hobby =
inStr.nextLine();

                                        hobby =
stringRegexCheck(hobby, patternHobby);

                                        partnerHobbies[i] = hobby;
                                    }
                                }
                                requirements = new
PartnerRequirements(partnerGender, minAge, maxAge, partnerHobbies);

    container.getElement(index).setRequirements(requirements);
                                break;
                            default:
                                System.out.println("\nWrong command.");
                                break;
                            }
                            break;
                        case 6:
                            endCheck2 = false;
                            System.out.println();
                            break;
                        default:
                            System.out.println("\nWrong command.");
                            break;
                        }
                    }
                    break;
                case 5:
                    container.clear();
                    System.out.println("Container cleared.\n");
                    break;
                case 6:
                    System.out.println("Choose the method");
                    System.out.println("1. Standard serialization");
                    System.out.println("2. XML serialization");
                    System.out.println("3. End");
                    System.out.println("Enter your option:");
                    option2 = inInt.nextInt();
                    System.out.println();
                    switch(option2)
                    {
                    case 1:
```

```
                                        try(ObjectOutputStream oos = new ObjectOutputStream(new
BufferedOutputStream(new FileOutputStream("Lab12.ser"))))
                                        {
                                                oos.writeObject(container);
                                                oos.flush();
                                                System.out.println("Serialization successful.\n");
                                        }
                                        catch(Exception ex)
                                        {
                                                System.out.println(ex.getMessage() + "\n");
                                        }
                                        break;
                                case 2:
                                        try(XMLEncoder encoder = new XMLEncoder(new
BufferedOutputStream(new FileOutputStream("Lab12.xml"))))
                                        {
                                                encoder.writeObject(container);
                                                System.out.println("Serialization successful.\n");
                                        }
                                        catch(Exception ex)
                                        {
                                                System.out.println(ex.getMessage() + "\n");
                                        }
                                        break;
                                case 3:
                                        break;
                                default:
                                        System.out.println("Wrong command.\n");
                                        break;
                                }
                                break;
                        case 7:
                                System.out.println("Choose the method");
                                System.out.println("1. Standard deserialization");
                                System.out.println("2. XML deserialization");
                                System.out.println("3. End");
                                System.out.println("Enter your option");
                                option2 = inInt.nextInt();
                                System.out.println();
                                switch(option2)
                                {
                                case 1:
                                        try(ObjectInputStream ois = new ObjectInputStream(new
BufferedInputStream(new FileInputStream("Lab12.ser"))))
                                        {
                                                container.clear();
                                                container = (ClientList<Client>) ois.readObject();
                                                System.out.println("Deserialization successful.\n");
                                        }
                                        catch(Exception ex)
                                        {
                                                System.out.println(ex.getMessage());
                                        }
                                        break;
                                case 2:
                                        try(XMLDecoder decoder = new XMLDecoder(new
BufferedInputStream(new FileInputStream("Lab12.xml"))))
                                        {
                                                container.clear();
                                        container = (ClientList<Client>)
decoder.readObject();
                                        System.out.println("Deserialization successful.\n");
                                    }
                                    catch(IOException ex)
                                    {
                                        System.out.println(ex.getMessage());
                                    }
                                    break;
                                case 3:
                                    break;
                                default:
                                    System.out.println("Wrong command.\n");
                                    break;
                                }
                                break;
                            case 8:
                                System.out.println("There is/are " + container.getSize() + " elements in
a container\n");
                                break;
                            case 9:
                                if(container.getSize() == 0)
                                {
                                    System.out.println("Empty container.\n");
                                    break;
                                }
                                System.out.println("Choose the method:");
                                System.out.println("1. Sort by ID");
                                System.out.println("2. Sort by registration date");
                                System.out.println("3. Sort by count of client's hobbies");
                                System.out.println("4. Sort by count of partner's hobbies");
                                System.out.println("Enter your option:");
                                option = inInt.nextInt();
                                System.out.println("\n1. Ascending");
                                System.out.println("2. Descending");
                                option2 = inInt.nextInt();
                                System.out.println();
                                switch (option)
                                {
                                case 1:
                                    container.sort(new IdComparator(), option2);
                                    System.out.println("Container sorted\n");
                                    break;
                                case 2:
                                    container.sort(new RegistrationDateComparator(), option2);
                                    System.out.println("Container sorted\n");
                                    break;
                                case 3:
                                    container.sort(new ClientHobbiesComparator(), option2);
                                    System.out.println("Container sorted\n");
                                    break;
                                case 4:
                                    container.sort(new PartnerHobbiesComparator(), option2);
                                    System.out.println("Container sorted\n");
                                    break;
                                default:
                                    System.out.println("Wrong command\n");
                                    break;
                                }
                                break;
```

```
                case 10:
                    if(container.getSize() == 0)
                    {
                        System.out.println("Empty container.\n");
                        break;
                    }
                    System.out.println("Enter the max age difference (max 9 years):");
                    maxAge = inInt.nextInt();
                    if(maxAge > 9)
                    {
                        System.out.println("\nYou enter wrong max age.\n");
                        break;
                    }
                    System.out.println();
                    String str = "^([" + 0 + "-" + maxAge + "])";
                    Pattern patternAgeDifference = Pattern.compile(str);
                    Pattern patternHobbyRuns = Pattern.compile("^(Morning runs)");
                    Pattern patternMale = Pattern.compile("^(Male)");
                    Pattern patternFemale = Pattern.compile("^(Female)");
                    Matcher matcherHobby1, matcherHobby2, matcherAge,
matcherGenderMale, matcherGenderFemale;
                    ArrayList<Integer> positions = new ArrayList<>();
                    boolean hobbyCheck1 = false, foundCouple = false;
                    for(int i = 0; i < container.getSize(); i++)
                    {
                        clientHobbies =
container.getElement(i).getInformation().getClientHobby();
                        partnerHobbies =
container.getElement(i).getRequirements().getPartnerHobby();
                        if(clientHobbies.length != 0 && partnerHobbies.length != 0)
                        {
                            for(int a = 0; a < clientHobbies.length; a++)
                            {
                                matcherHobby1 =
patternHobbyRuns.matcher(clientHobbies[a]);
                                if(matcherHobby1.matches())
                                {
                                    hobbyCheck1 = true;
                                    break;
                                }
                            }
                            if(hobbyCheck1 == true)
                                for(int b = 0; b < partnerHobbies.length; b++)
                                {
                                    matcherHobby2 =
patternHobbyRuns.matcher(partnerHobbies[b]);
                                    if(matcherHobby2.matches())
                                        positions.add(i);
                                }
                        }
                    }
                    int num = 1;
                    if(!positions.isEmpty())
                        for(int i = 0; i < container.getSize(); i++)
                        {
                            if(positions.contains(i))
                                for(int j = i + 1; j < container.getSize(); j++)
                                    if(positions.contains(j))
                                    {
```

```
                                                                    int        ageDifference        =
Math.abs(container.getElement(i).getInformation().getAge() - container.getElement(j).getInformation().getAge());
                                                                    matcherAge                      =
patternAgeDifference.matcher(Integer.toString(ageDifference));
                                                                    if(matcherAge.matches())
                                                                    {

      matcherGenderMale = patternMale.matcher(container.getElement(i).getClientGender());

      if(matcherGenderMale.matches())
                                                                          {

      matcherGenderFemale = patternFemale.matcher(container.getElement(j).getClientGender());

      if(matcherGenderFemale.matches())
                                                                                {

      System.out.println("Couple   "   +   num   +   ":\n"   +   container.getElement(i).toString()   +   "\n"   +
container.getElement(j).toString() + "\n");

      foundCouple = true;

      num++;
                                                                                }
                                                                          }
                                                                          else
                                                                          {

      matcherGenderMale = patternMale.matcher(container.getElement(j).getClientGender());

      if(matcherGenderMale.matches())
                                                                                {

      System.out.println("Couple   "   +   num   +   ":\n"   +   container.getElement(i).toString()   +   "\n"   +
container.getElement(j).toString() + "\n");

      foundCouple = true;

      num++;
                                                                                }
                                                                          }

                                                                    }
                                                              }
                                      }
                                if(foundCouple != true)
                                      System.out.println("There is no matching couples.\n");
                                break;
                          case 11:
                                final int ARR_SIZE = 10000;
                                final int NUMBER_OF_THREADS;
                                final int NUMBER_OF_ITERATIONS;
                                int option1;
                                long time1, time2;
                                System.out.println("Adding new elements...");
                                for(int i = 0; i < ARR_SIZE; i++)
                                {
                                      String[] hobbies = {Integer.toString(i)};
                                      info   =   new   InfoAboutYourself(Integer.toString(i),   i,   i,
Integer.toString(i), hobbies);
```

```
                        requirements = new PartnerRequirements(Integer.toString(i), i,
i, hobbies);
                        date = new GregorianCalendar();
                        container.add(new Client(Integer.toString(i), i, date, info,
requirements));
                    }
                    System.out.println(container.toString());
                    System.out.println("Calculations:");
                    System.out.println("1. Parallel");
                    System.out.println("2. Serial");
                    option1 = inInt.nextInt();
                    System.out.println();
                    if(option1 != 1 && option1 != 2)
                    {
                        System.out.println("You have entered the wrong command");
                        break;
                    }
                    if(option1 == 1)
                    {
                        NUMBER_OF_THREADS = 3;
                        NUMBER_OF_ITERATIONS = 1;
                    }
                    else
                    {
                        NUMBER_OF_THREADS = 1;
                        NUMBER_OF_ITERATIONS = 3;
                    }
                    MyThread[] threads = new MyThread[NUMBER_OF_THREADS];
                    try
                    {
                        for(int i = 0; i < NUMBER_OF_THREADS; i++)
                        {
                            threads[i] = new MyThread(container, "Thread " +
(i+1), NUMBER_OF_ITERATIONS);
                            threads[i].thread.start();
                        }
                        time1 = System.currentTimeMillis();
                        for(int i = 0; i < NUMBER_OF_THREADS; i++)
                            threads[i].thread.join();
                        time2 = System.currentTimeMillis();
                        System.out.println("Time result: " + (double)(time2 -
time1)/1000 + " seconds");
                    }
                    catch(InterruptedException ex)
                    {
                        System.out.println("Thread has been interrupted.");
                    }
                    System.out.println();
                    container.clear();
                    break;
                case 0:
                    endCheck = false;
                    container.clear();
                    inInt.close();
                    inStr.close();
                    break;
                default:
                    System.out.println("Wrong command\n");
                    break;
                }
```

```
		}
		System.out.println("End.");
	}

	public static int indexGenerator(ClientList<Client> arr)
	{
		arr.sort(new IdComparator(), 1);
		int index = 1;
		for(int i = 0; i < arr.getSize(); i++)
			if(index == arr.getElement(i).getId())
				index++;
			else
				return index;
		return index;
	}

	public static int intRegexCheck(int value, Pattern pattern)
	{
		Matcher matcher;
		Scanner in = new Scanner(System.in);
		boolean ready = false;
		do
		{
			matcher = pattern.matcher(Integer.toString(value));
			if(!matcher.matches())
			{
				System.out.println("You've entered the wrong data. Try again:");
				value = in.nextInt();
			}
			else
				ready = true;
		}
		while(!ready);
		return value;
	}

	public static String stringRegexCheck(String value, Pattern pattern)
	{
		Matcher matcher;
		Scanner in = new Scanner(System.in);
		boolean ready = false;
		do
		{
			matcher = pattern.matcher(value);
			if(!matcher.matches())
			{
				System.out.println("You've entered the wrong data. Try again:");
				value = in.nextLine();
			}
			else
				ready = true;
		}
		while(!ready);
		return value;
	}
}
```

Клас ClientList

```
package ua.khpi.oop.zanochkyn14;

import java.io.Serializable;
import java.util.Comparator;
import java.util.Iterator;
import java.util.NoSuchElementException;
import ua.khpi.oop.zanochkyn10.Client;
import ua.khpi.oop.zanochkyn10.Node;

public class ClientList<T> implements Serializable, Iterable<T>
{
	private static final long serialVersionUID = 5493313651067238933L;
	public Node<T> head;
	private int size;

	/*
	 * Getter and setter for size
	 */
	public int getSize() { return size; }
	public void setSize(int size) { this.size = size; }

	/*
	 * Method (add) that add a new client into container
	 */
	public void add(T el)
	{
		Node<T> temp = new Node<T>();
		if(head == null)
			head = new Node<T>(el);
		else
		{
			temp = head;
			while(temp.next != null)
				temp = temp.next;
			temp.next = new Node<T>(el);
		}
		size++;
	}

	/*
	 * Method (remove) that remove a client from container
	 */
	public void remove(int id)
	{
		Node<T> temp = head;
		if(head != null)
		{
			if(id == 0)
				head = head.next;
			else
			{
				for(int i = 0; i < id - 1; i++)
					temp = temp.next;
				if(temp.next != null)
					temp.next = temp.next.next;
				else
					temp.next = null;
			}
			size--;
		}
		else
			System.out.println("Container is empty.");
	}

	/*
	 * Method (clear) that clear the container
	 */
	public void clear()
	{
		this.head = null;
		size = 0;
	}

	/*
	 * Method (toArray[]) that return container as an array
	 */
	public Object[] toArray()
	{
		Object[] arr = new Object[size];
		for(int i = 0; i < size; i++)
			arr[i] = getElement(i);
		return arr;
	}

	/*
	 * Method (getElement) that return a specific element from container
	 */
	public T getElement(int id)
	{
		if(id < 0 || id >= size)
		{
			System.out.println("Wrong id.");
			return null;
		}
		Node<T> temp = head;
		for(int i = 0; i < id; i++)
			temp = temp.next;
		return temp.element;
	}

	/*
	 * Method (toString) that return a container as a string
	 */
	public String toString()
	{
		StringBuilder sb = new StringBuilder();
		for(T value : this)
			sb.append(value + "\n");
		return sb.toString();
	}

	@SuppressWarnings("unchecked")
	public void sort(Comparator<T> comp, int option)
	{
		Object[] arr = this.toArray();
		Object temp;
		boolean flag;
		if(option == 1)
```

```
                do
                {
                        flag = false;
                        for(int i = 0; i < size - 1; i++)
                                if(comp.compare((T)arr[i], (T)arr[i+1]) == 1)
                                {
                                        flag = true;
                                        temp = arr[i];
                                        arr[i] = arr[i+1];
                                        arr[i+1] = temp;
                                }
                }
                while(flag == true);
        else
                do
                {
                        flag = false;
                        for(int i = 0; i < size - 1; i++)
                                if(comp.compare((T)arr[i], (T)arr[i+1]) == -1)
                                {
                                        flag = true;
                                        temp = arr[i+1];
                                        arr[i+1] = arr[i];
                                        arr[i] = temp;
                                }
                }
                while(flag == true);
        this.clear();
        for (Object i : arr)
                this.add((T) i);
}

public Iterator<T> iterator()
{
        return new Iterator<T>()
        {
                int index = 0;
                boolean check = false;

                /*
                 * Method that returns true if the iteration has more elements
                 */
                @Override
                public boolean hasNext()
                {
                        return index < size;
                }

                /*
                 * Method that returns the next element in the iteration
                 */
                @Override
                public T next()
                {
                        if (index == size)
                                throw new NoSuchElementException();
                        check = true;
                        return getElement(index++);
                }
```

```
				/*
				 * Method that removes from the container the last element returned by this
iterator
				 */
				@Override
				public void remove()
				{
					if (check)
					{
						ClientList.this.remove(index - 1);
						check = false;
					}
					else
						throw new IllegalStateException();
				}
			};
		}
	}

	class RegistrationDateComparator implements Comparator<Client>
	{
		public int compare(Client o1, Client o2)
		{
			if(o1.getDate().getTimeInMillis() > o2.getDate().getTimeInMillis())
				return 1;
			else if(o1.getDate().getTimeInMillis() < o2.getDate().getTimeInMillis())
				return -1;
			else
				return 0;
		}
	}

	class ClientHobbiesComparator implements Comparator<Client>
	{
		public int compare(Client o1, Client o2)
		{
			if(o1.getInformation().getClientHobby().length                    >
o2.getInformation().getClientHobby().length)
				return 1;
			else              if(o1.getInformation().getClientHobby().length        <
o2.getInformation().getClientHobby().length)
				return -1;
			else
				return 0;
		}
	}

	class PartnerHobbiesComparator implements Comparator<Client>
	{
		public int compare(Client o1, Client o2)
		{
			if(o1.getRequirements().getPartnerHobby().length                  >
o2.getRequirements().getPartnerHobby().length)
				return 1;
			else             if(o1.getRequirements().getPartnerHobby().length       <
o2.getRequirements().getPartnerHobby().length)
				return -1;
			else
				return 0;
		}
```

```
}

class IdComparator implements Comparator<Client>
{
	public int compare(Client o1, Client o2)
	{
		if(o1.getId() > o2.getId())
			return 1;
		else if(o1.getId() < o2.getId())
			return -1;
		else
			return 0;
	}
}
```

Клас MyThread

```
package ua.khpi.oop.zanochkyn14;

import ua.khpi.oop.zanochkyn10.Client;

public class MyThread implements Runnable
{
	private boolean isActive;
	Thread thread;
	private ClientList<Client> container;
	private int time;

	MyThread(ClientList<Client> container, String name, int time)
	{
		this.container = container;
		isActive = true;
		thread = new Thread(this, name);
		this.time = time;
	}

	void disable()
	{
		isActive = false;
	}

	@Override
	public void run()
	{
		long countTime = 0;
		long temp = 0;
		for(int i = 0; i < time; i++)
		{
			try
			{
				temp = count();
			}
			catch (InterruptedException e)
			{
				e.printStackTrace();
			}
			countTime += temp;
		}
		System.out.println("Time spent: " + countTime + " milliseconds");
	}
```

```
		private long count() throws InterruptedException
		{
			long count = 0;
			long begin = System.currentTimeMillis();
			Thread.currentThread().sleep(1000);
			for(Client i : container)
				if(isActive)
					count += i.getInformation().getAge();
				else
				{
					System.out.println(Thread.currentThread().getName()	+	"	was
stopped.");
					return -1;
				}
			System.out.println(Thread.currentThread().getName() + ": " + count);
			System.out.println(Thread.currentThread().getName() + " finished");
			return (System.currentTimeMillis() - begin);
		}
	}
```

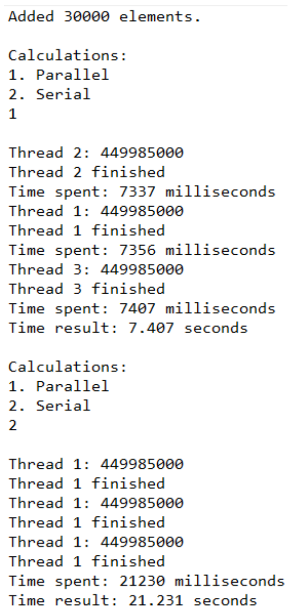
3 ВАРІАНТИ ВИКОРИСТАННЯ

Можливість виконання програми в автоматичному режимі, якщо ввести у командному рядку аргументи –a або –auto та у діалоговому режимі – аргументи –d або –dialog.

У діалоговому режимі було розроблено меню, яке дозволяє користувачу:

1. Вивести усі елементи у консоль (1 команда меню) ;
2. Додати елемент у контейнер (2 команда меню);
3. Видалити елемент з контейнеру (3 команда меню);
4. Редагувати один з елементів (4 команда меню);
5. Очистити контейнер (5 команда меню);
6. Серіалізувати контейнер у файл (6 команда меню);
7. Десеріалізувати контейнер (7 команда меню);
8. Визначити кількість елементів у контейнері (8 команда меню);
9. Сортування контейнера (9 команда меню);
10. Знайти всі комбінації пар (10 команда меню);
11. Виконати завдання з потоками (11 команда меню);
12. Закінчити виконання програми (0 команда меню).

4 РЕЗУЛЬТАТИ РОБОТИ ПРОГРАМИ



```
Added 30000 elements.

Calculations:
1. Parallel
2. Serial
1

Thread 2: 449985000
Thread 2 finished
Time spent: 7337 milliseconds
Thread 1: 449985000
Thread 1 finished
Time spent: 7356 milliseconds
Thread 3: 449985000
Thread 3 finished
Time spent: 7407 milliseconds
Time result: 7.407 seconds

Calculations:
1. Parallel
2. Serial
2

Thread 1: 449985000
Thread 1 finished
Thread 1: 449985000
Thread 1 finished
Thread 1: 449985000
Thread 1 finished
Time spent: 21230 milliseconds
Time result: 21.231 seconds
```

Рисунок 14.1 – Результат роботи програми у середовищі Eclipse

Висновок

Під час виконання лабораторної роботи було набуто навички роботи з паралельною обробкою та багатопоточністю і визначенням ефективності паралельної обробки даних в середовищі Eclipse IDE.