**Лабораторна робота №10**

**Обробка параметризованих контейнерів**

**Мета:** Розширення функціональності параметризованих класів.

**1 ВИМОГИ**

1. Розробити параметризовані методи (Generic Methods) для обробки колекцій об'єктів згідно прикладної задачі.
2. Продемонструвати розроблену функціональність (створення, управління та обробку власних контейнерів) в діалоговому та автоматичному режимах.

* Автоматичний режим виконання програми задається параметром командного рядка -auto. Наприклад, java ClassName -auto.
* В автоматичному режимі діалог з користувачем відсутній, необхідні данні генеруються, або зчитуються з файлу.

1. Забороняється використання алгоритмів з Java Collections Framework.
   1. **Розробник**

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

Реалізувати сортування за датою реєстрації, за кількістю властивостей в розділі "відомості про себе", за кількістю властивостей в розділі "вимоги до партнера".

1. **ОПИС ПРОГРАМИ**
   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("Lab10.xml"));

encoder.writeObject(container); – нестандартна серіалізація;

XMLDecoder decoder = new XMLDecoder(new BufferedInputStream(new FileInputStream("Lab10.xml")));

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

ObjectOutputStream oos = new ObjectOutputStream(new BufferedOutputStream(newFileOutputStream("Lab10.ser")));

oos.writeObject(container);

oos.flush(); – стандартна серіалізація;

ObjectInputStream ois = new ObjectInputStream(new BufferedOutputStream(new FileInputStream("Lab10.ser")));

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

* 1. **Ієрархія та структура класів**

Було створено класи Main (головний клас програми), Client (клас, що містить всі поля та методи прикладної області «Бюро знайомств»), ClientList (клас-контейнер), Node (клас-покажчик на елемент) та 4 класи, що реалізують інтерфейс Comparator для сортування за певними критеріями.

* 1. **Важливі фрагменти програми**

Клас Main

package ua.khpi.oop.zanochkyn10;

import java.beans.XMLDecoder;

import java.beans.XMLEncoder;

import java.io.BufferedInputStream;

import java.io.BufferedOutputStream;

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.ObjectInputStream;

import java.io.ObjectOutputStream;

import java.util.Calendar;

import java.util.GregorianCalendar;

import java.util.Scanner;

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...");

String gender;

GregorianCalendar date;

InfoAboutYourself info;

PartnerRequirements requirements;

gender = "Male";

String name = "Yehor";

int age = 19;

int height = 185;

String eyeColour = "Blue";

String[] clientHobbies = {"Video games", "Music"};

info = new InfoAboutYourself(name, age, height, eyeColour, clientHobbies);

String partnerGender = "Female";

int minAge = 18;

int maxAge = 25;

String[] partnerHobbies = {};

requirements = new PartnerRequirements(partnerGender, minAge, maxAge, partnerHobbies);

date = new GregorianCalendar();

container.add(new Client(gender, indexGenerator(container), date, info, requirements));

gender = "Female";

name = "Katya";

age = 18;

height = 175;

eyeColour = "Green";

String[] clientHobbies2 = {"Music"};

info = new InfoAboutYourself(name, age, height, eyeColour, clientHobbies2);

partnerGender = "Male";

minAge = 18;

maxAge = 25;

String[] partnerHobbies2 = {"Music"};

requirements = new PartnerRequirements(partnerGender, minAge, maxAge, partnerHobbies2);

date = new GregorianCalendar();

container.add(new Client(gender, indexGenerator(container), date, info, requirements));

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());

System.out.println("Change the second client's hobby...");

String[] clientHobbies3 = {"Art"};

container.getElement(1).getInformation().setClientHobby(clientHobbies3);

System.out.println("Second client's hobby - changed.");

System.out.println("\nOutput the container...");

System.out.println("\n" + container.toString());

System.out.println("Sorting the container by date (descending)...");

container.sort(new RegistrationDateComparator(), 2);

System.out.println("Container sorted");

System.out.println("\nOutput the container...");

System.out.println("\n" + container.toString());

System.out.println("Removing first client from the container...");

container.remove(0);

System.out.println("First client removed.");

System.out.println("\nOutput the container...");

System.out.println("\n" + container.toString());

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

}

private static void menu(ClientList<Client> container)

{

String gender;

GregorianCalendar date;

InfoAboutYourself info;

PartnerRequirements requirements;

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("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("Enter gender:");

gender = inStr.nextLine();

System.out.println("\nEnter information about yourself");

System.out.println("Name:");

String name = inStr.nextLine();

System.out.println("Age:");

int age = inInt.nextInt();

System.out.println("Height:");

int height = inInt.nextInt();

System.out.println("Eye colour:");

String eyeColour = inStr.nextLine();

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:");

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

clientHobbies[i] = inStr.nextLine();

}

info = new InfoAboutYourself(name, age, height, eyeColour, clientHobbies);

System.out.println("\nEnter partner requirements");

System.out.println("Gender:");

String partnerGender = inStr.nextLine();

System.out.println("Min age:");

int minAge = inInt.nextInt();

System.out.println("Max age:");

int maxAge = inInt.nextInt();

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:");

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

partnerHobbies[i] = inStr.nextLine();

}

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("There 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();

System.out.println();

switch(option2)

{

case 1:

System.out.println("Enter new gender:");

container.getElement(index).setClientGender(inStr.nextLine());

break;

case 2:

System.out.println("Enter new ID:");

container.getElement(index).setId(inInt.nextInt());

break;

case 3:

GregorianCalendar newDate = new GregorianCalendar();

System.out.println("Enter registration year:");

int value = inInt.nextInt();

newDate.set(Calendar.YEAR, value);

System.out.println("Enter registration month:");

value = inInt.nextInt();

newDate.set(Calendar.MONTH, value-1);

System.out.println("Enter registration day:");

value = inInt.nextInt();

newDate.set(Calendar.DAY\_OF\_MONTH, value);

System.out.println("Enter registration hour:");

value = inInt.nextInt();

newDate.set(Calendar.HOUR\_OF\_DAY, value);

System.out.println("Enter registration minute:");

value = inInt.nextInt();

newDate.set(Calendar.MINUTE, value);

newDate.set(Calendar.SECOND, 0);

container.getElement(index).setDate(newDate);

break;

case 4:

System.out.println("Information 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:");

container.getElement(index).getInformation().setName(inStr.nextLine());

break;

case 2:

System.out.println("Enter new age:");

container.getElement(index).getInformation().setAge(inInt.nextInt());

break;

case 3:

System.out.println("Enter new height:");

container.getElement(index).getInformation().setHeight(inInt.nextInt());

break;

case 4:

System.out.println("Enter new eye colour:");

container.getElement(index).getInformation().setEyeColour(inStr.nextLine());

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("\nEnter new client's hobbies:");

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

clientHobbies[i] = inStr.nextLine();

}

container.getElement(index).getInformation().setClientHobby(clientHobbies);

break;

case 6:

System.out.println("Enter new name:");

name = inStr.nextLine();

System.out.println("Enter new age:");

age = inInt.nextInt();

System.out.println("Enter new height:");

height = inInt.nextInt();

System.out.println("Enter new eye colour:");

eyeColour = inStr.nextLine();

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

countClientHobbies = inInt.nextInt();

clientHobbies = new String[countClientHobbies];

if(countClientHobbies != 0)

{

System.out.println("\nEnter new client's hobbies:");

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

clientHobbies[i] = inStr.nextLine();

}

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("Partner 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();

System.out.println();

switch(option3)

{

case 1:

System.out.println("Enter new gender:");

container.getElement(index).getRequirements().setPartnerGender(inStr.nextLine());

break;

case 2:

System.out.println("Enter new min age:");

container.getElement(index).getRequirements().setMinAge(inInt.nextInt());

break;

case 3:

System.out.println("Enter new max age:");

container.getElement(index).getRequirements().setMaxAge(inInt.nextInt());

break;

case 4:

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

countPartnerHobbies = inInt.nextInt();

partnerHobbies = new String[countPartnerHobbies];

if(countPartnerHobbies != 0)

{

System.out.println("\nEnter partner's hobbies:");

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

partnerHobbies[i] = inStr.nextLine();

}

container.getElement(index).getRequirements().setPartnerHobby(partnerHobbies);

break;

case 5:

System.out.println("Enter new gender:");

partnerGender = inStr.nextLine();

System.out.println("Enter new min age:");

minAge = inInt.nextInt();

System.out.println("Enter new max age:");

maxAge = inInt.nextInt();

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

countPartnerHobbies = inInt.nextInt();

partnerHobbies = new String[countPartnerHobbies];

if(countPartnerHobbies != 0)

{

System.out.println("\nEnter partner's hobbies:");

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

partnerHobbies[i] = inStr.nextLine();

}

requirements = new PartnerRequirements(partnerGender, minAge, maxAge, partnerHobbies);

container.getElement(index).setRequirements(requirements);

break;

default:

System.out.println("Wrong command.");

break;

}

break;

case 6:

endCheck2 = false;

break;

default:

System.out.println("Wrong 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("Lab10.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("Lab10.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("Lab10.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("Lab10.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 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;

}

}

Клас Client

package ua.khpi.oop.zanochkyn10;

import java.io.Serializable;

import java.util.GregorianCalendar;

public class Client implements Serializable

{

private static final long serialVersionUID = 8633968308489911794L;

/\*

\* Змінні

\*/

private String gender;

private int id;

private GregorianCalendar registrationDate;

private InfoAboutYourself information;

private PartnerRequirements requirements;

/\*

\* Конструктори класу

\*/

public Client(String gender, int id, GregorianCalendar date, InfoAboutYourself info, PartnerRequirements requirements)

{

this.gender = gender;

this.id = id;

this.registrationDate = date;

this.information = info;

this.requirements = requirements;

}

public Client()

{

}

/\*

\* Геттери та сеттери

\*/

public String getClientGender()

{

return gender;

}

public void setClientGender(String gender)

{

this.gender = gender;

}

public int getId()

{

return id;

}

public void setId(int id)

{

this.id = id;

}

public GregorianCalendar getDate()

{

return registrationDate;

}

public void setDate(GregorianCalendar date)

{

this.registrationDate = date;

}

public InfoAboutYourself getInformation()

{

return information;

}

public void setInformation(InfoAboutYourself info)

{

this.information = info;

}

public PartnerRequirements getRequirements()

{

return requirements;

}

public void setRequirements(PartnerRequirements requirements)

{

this.requirements = requirements;

}

public String toString()

{

return "ID - " + id + "\nRegistration date - " + registrationDate.getTime() + "\nGender - " + gender + "\n\n" +

"Information about yourself:\nName - " + getInformation().getName() + "\nAge - " + getInformation().getAge() +

"\nHeight - " + getInformation().getHeight() + "\nEye colour - " + getInformation().getEyeColour() +

"\nHobbies - " + hobbiesToString(getInformation().getClientHobby()) + "\n\n" +

"Partner requirements:\nGender - " + getRequirements().getPartnerGender() + "\nMin age - " + getRequirements().getMinAge() +

"\nMax age - " + getRequirements().getMaxAge() + "\nHobbies - " + hobbiesToString(getRequirements().getPartnerHobby()) +

"\n----------------------------------------";

}

public String hobbiesToString(String[] arr)

{

int size = arr.length;

if(size == 0)

return "No hobbies";

StringBuilder sb = new StringBuilder();

int i = 1;

for(String temp : arr)

{

if(i != size)

sb.append(temp + ", ");

else

sb.append(temp);

i++;

}

return sb.toString();

}

}

Клас ClientList

package ua.khpi.oop.zanochkyn10;

import java.io.Serializable;

import java.util.Comparator;

import java.util.Iterator;

import java.util.NoSuchElementException;

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

\*/

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

\*/

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 Node

package ua.khpi.oop.zanochkyn10;

import java.io.Serializable;

public class Node<T> implements Serializable

{

private static final long serialVersionUID = -2673405972360871471L;

public T element;

public Node<T> next;

public Node() {}

public Node(T el)

{

super();

this.element = el;

}

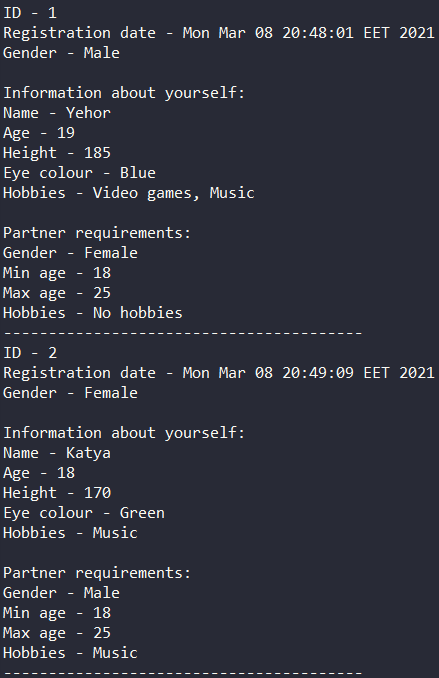
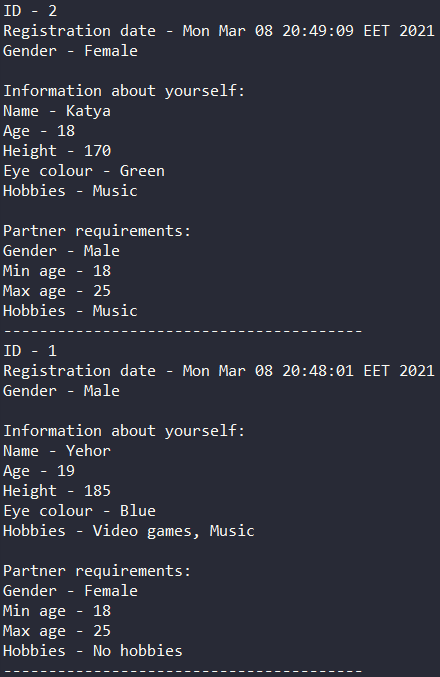
}

1. **ВАРІАНТИ ВИКОРИСТАННЯ**

Додано 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. Закінчити виконання програми (0 команда меню).
11. **РЕЗУЛЬТАТИ РОБОТИ ПРОГРАМИ**

**** ****

а) б)

Рисунок 10.1 – Результат роботи програми (сортування) у середовищі Eclipse

**Висновок**

Під час виконання лабораторної роботи було набуто навички роботи з розробки параметризованих методів в середовищі Eclipse IDE.