Лабораторна робота №8

Основи введення/виведення Java SE

Мета: Оволодіння навичками управління введенням/виведенням даних з використанням класів платформи Java SE

1 ВИМОГИ

1) Забезпечити можливість збереження і відновлення масива об'єктів рішення завдання лабораторної роботи №7.

2) Забороняється використання стандартного протокола серіалізації.

3) Продемонструвати використання моделі Long Term Persistence.

4) Забезпечити діалог з користувачем у вигляді простого текстового меню.

5) При збереженні та відновленні даних забезпечити діалоговий режим вибору директорії з відображенням вмісту і можливістю переміщення по підкаталогах.

1.1 Розробник

- П.І.Б: Абдуллін О. Р.
- Группа: КІТ-119а
- Варіант: 1

1.2 Завдання

Кадрове агенство.
Дані про претендента: реєстраційний номер; досвід роботи - набір значень "спеціальність, стаж"; освіта; дата звільнення; вимоги до майбутньої роботи - набір необов'язкових властивостей у вигляді "спеціальність, умови праці, мінімальна зарплата".

2 ОПИС ПРОГРАМИ

2.1 Було використано наступні засоби:

File folder = new File (absolutePath) – отримання адреси каталогу;

listFiles.length() – визначення довжини масиву назв каталогів та файлів,
XMLEncoder encoder = new XMLEncoder(new BufferedOutputStream(new FileOutputStream(file))), encoder.writeObject(list.array), encoder.close() – серіалізація;

XMLDecoder decoder = new XMLDecoder(new BufferedInputStream(new FileInputStream(file))), list.array = (Client[]) decoder.readObject(), decoder.close() – десеріалізація.

2.2 Ієрархія та структура класів

Було створено 2 класи

1) public class Main - містить метод main.

2) public class RecruitinfAgency – клас, що містить масив типу Challanger та метод вивиду усіх данних.

Також було підключено класи Challanger, DemandsToWork та WorkExperience з попередньої лабораторної роботи.

2.3 Важливі фрагменти програми

Клас RecruitingAgency

```
public class RecruitingAgency {
	private int size = 0;
	Challanger[] mas = new Challanger[size];

	public int getSize() {
		return size;
	}
	public void setSize(int size) {
		this.size = size;
	}

	public void add(Challanger challanger) {
		Challanger[] newmas = new Challanger[size+1];
		for(int i = 0; i < size; i++)
			newmas[i] = mas[i];
		size++;
		newmas[size-1] = challanger;
		mas = newmas;
	}

	public boolean remove(int num) {
		if(num > 0 || num < size) {
			Challanger[] newmas = new Challanger[size-1];
			for (int i = 0; i < num; i++)
				newmas[i] = mas[i];
			for (int i = num, j = num + 1; j < size; i++, j++)
				newmas[i] = mas[j];
			size--;
			mas = newmas;
			return true;
		}
		return false;
	}
	public boolean removeID(int ID) {
		int num = -1;
		for(int i = 0; i < size; i++ ) {
			if(mas[i].getRegistrationNum() == ID) {
				num = i;
				break;
			}
		}
		if(num != -1)
			return remove(num);
		else
			return false;
	}

	public void clear() {
		size = 0;
		Challanger[] newmas = new Challanger[size];
		mas = newmas;
	}
```

```
	public void printID(int ID) {
		int num = -1;
		for(int i = 0; i < size; i++ ) {
			if(mas[i].getRegistrationNum() == ID) {
				num = i;
				break;
			}
		}
		if(num != -1)
			print(num+1);
		else
			System.out.println("Указан неверный ID.");
	}
	public void print(int num) {
		if(num > 0 || num < size) {
			System.out.println("ID: " + mas[num-1].getRegistrationNum());
			System.out.println("Образование: " + mas[num-1].getEducation());
			System.out.println("Дата увольнения: " + mas[num-
1].getDismissalDay()+"/"+mas[num-1].getDismissalMonth()+"/"+mas[num-
1].getDismissalYear());
			System.out.println("---Опыт работы---");
			System.out.println("Место предыдущей работы: " + mas[num-
1].getWorkExperience().getSpecialization());
			if(mas[num-1].getWorkExperience().getExperience() <= 4)
				System.out.println("Стаж: " + mas[num-
1].getWorkExperience().getExperience() + " год(а)");
			else
				System.out.println("Стаж: " + mas[num-
1].getWorkExperience().getExperience() + " лет");
			System.out.println("---Желания по будующей работе---" );
			if(mas[num-1].getDemandsToWork().getMinSalary() == 0 && mas[num-
1].getDemandsToWork().getSpecialization() == null && mas[num-
1].getDemandsToWork().getConditions() == null)
				System.out.println("Предендет не имеет никаких желаний по
будующей работе");
			else {
				if( mas[num-1].getDemandsToWork().getMinSalary() != 0)
					System.out.println("Желаемая минимальная зарплата: "
+  mas[num-1].getDemandsToWork().getMinSalary());
				else
					System.out.println("Желаемая минимальная зарплата:
Претендент не имеет пожеланий к этому пункту " );
				if(mas[num-1].getDemandsToWork().getSpecialization() !=
null)
					System.out.println("Желаемая будующая работа: " +
mas[num-1].getDemandsToWork().getSpecialization());
				else
					System.out.println("Желаемая будующая работа:
Претендент не имеет пожеланий к этому пункту");
				if(mas[num-1].getDemandsToWork().getConditions() != null)
					System.out.println("Желаемые условия будующей работы:
" + mas[num-1].getDemandsToWork().getConditions());
				else
					System.out.println("Желаемые условия будующей работы:
Претендент не имеет пожеланий к этому пункту");
			}
			System.out.println("----------------------------------------");
		}
		else
			System.out.println("Указан невеный елемент.");
	}
	/**
```

```
	 * Метод вывода массива данных
	 */
	public void printAll() {
		if(size > 0) {
			for(int i = 0 ; i < size; i++) {
				System.out.println("ID: " + mas[i].getRegistrationNum());
				System.out.println("Образование:  "  +
mas[i].getEducation());
				System.out.println("Дата  увольнения:  "  +
mas[i].getDismissalDay()+"/"+mas[i].getDismissalMonth()+"/"+mas[i].getDismissalYear()
);
				System.out.println("---Опыт работы---");
				System.out.println("Место  предыдущей  работы:  "  +
mas[i].getWorkExperience().getSpecialization());
				if(mas[i].getWorkExperience().getExperience() <= 4)
					System.out.println("Стаж:  "  +
mas[i].getWorkExperience().getExperience() + " год(а)");
				else
					System.out.println("Стаж:  "  +
mas[i].getWorkExperience().getExperience() + " лет");
				System.out.println("---Желания по будующей работе---" );
				if(mas[i].getDemandsToWork().getMinSalary()  ==  0  &&
mas[i].getDemandsToWork().getSpecialization()  ==  null  &&
mas[i].getDemandsToWork().getConditions() == null)
					System.out.println("Предендет  не  имеет  никаких
желаний по будующей работе");
				else {
					if( mas[i].getDemandsToWork().getMinSalary() != 0)
						System.out.println("Желаемая  минимальная
зарплата: " +  mas[i].getDemandsToWork().getMinSalary());
					else
						System.out.println("Желаемая  минимальная
зарплата: Претендент не имеет пожеланий к этому пунку " );
					if(mas[i].getDemandsToWork().getSpecialization()  !=
null)
						System.out.println("Желаемая будующая работа:
" + mas[i].getDemandsToWork().getSpecialization());
					else
						System.out.println("Желаемая будующая работа:
Претендент не имеет пожеланий к этому пунку");
					if(mas[i].getDemandsToWork().getConditions()  !=
null)
						System.out.println("Желаемые условия будующей
работы: " + mas[i].getDemandsToWork().getConditions());
					else
						System.out.println("Желаемые условия будующей
работы: Претендент не имеет пожеланий к этому пунку");
				}
				System.out.println("-------------------------------------
----");
			}
		}
		else
			System.out.println("Претендентов на роботу нету.");
	}
}
```

Клас Main

```
public class Main {

	public static void main(String[] args) {
		RecruitingAgency list = new RecruitingAgency();
		int id = 1;
		WorkExperience workExperience = new WorkExperience("Кассир",4);
		DemandsToWork demandsToWork = new DemandsToWork("Менеджер по
продажам",7800,"Наличие кофеварки на работе.");
		Challanger challanger = new Challanger(id++,"Среднее не
полное",13,05,2020,workExperience,demandsToWork);
		list.add(challanger);

		workExperience = new WorkExperience("Учитель",14);
		demandsToWork = new DemandsToWork(null,0,null);
		challanger = new Challanger(id++,"Высшее образование",15,10,2014,
workExperience,demandsToWork);

		list.add(challanger);

		workExperience = new WorkExperience("Бухгалтер",38);
		demandsToWork = new DemandsToWork("Бухгалтер",15000,"Оффис в цетре.");
		challanger = new Challanger(id++,"Высшее образование",14,03,2020,
workExperience,demandsToWork);

		list.add(challanger);

		boolean endprog = false;
		boolean endChange = false;
		boolean endSerialization = false;
		boolean endDeserializtion = false;
		boolean folderDeserialization = true;
		boolean forlderSerialization = true;
		Scanner inInt = new Scanner(System.in);
		Scanner inStr = new Scanner(System.in);
		int menu;
		int menuChange;
		int changeExperience;
		int changeDemands;
		int menuSerialization;
		int menuDeserialization;

		while(!endprog)
		{
			System.out.println("1. Show all challanger");
			System.out.println("2. Add challanger");
			System.out.println("3. Delete chellanger");
			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. Exit");
			System.out.println("Enter option: \n");
			try
			{
				menu =  inInt.nextInt();
			}
			catch(java.util.InputMismatchException e)
			{
				System.out.println("Error! Ошибка ввода.");
				endprog = true;
```

```
                        menu = 8;
                }

                switch(menu)
                {
                case 1:
                        list.printAll();
                        break;
                case 2:
                        System.out.println("Enter education of challanger: ");
                        String education = inStr.nextLine();
                        System.out.println("Enter day of dismissal: ");
                        int day = inInt.nextInt();
                        System.out.println("Enter month of dismissal: ");
                        int month = inInt.nextInt();
                        System.out.println("Enter year of dismissal: ");
                        int year = inInt.nextInt();
                        System.out.println("Enter pervious job: ");
                        String specializationPrevious = inStr.nextLine();
                        System.out.println("Enter experience of working: ");
                        int experience = inInt.nextInt();
                        System.out.println("Enter next job: ");
                        String specializationNext = inStr.nextLine();
                        System.out.println("Enter min salary: ");
                        int minSalary = inInt.nextInt();
                        System.out.println("Enter whishes to the next job: ");
                        String conditions = inStr.nextLine();

                        WorkExperience workExperienceAdd = new
WorkExperience(specializationPrevious, experience);
                        DemandsToWork demandsToWorkAdd = new
DemandsToWork(specializationNext,minSalary,conditions);
                        Challanger challangerAdd = new
Challanger(id++,education,day,month,year,workExperienceAdd,demandsToWorkAdd);
                        list.add(challangerAdd);
                        break;
                case 3:
                        System.out.println("Enter ID to delete: ");
                        int delete = inInt.nextInt();
                        if(list.removeID(delete))
                                System.out.println("Challanger was deleted
successfully.");
                        else
                                System.out.println("Error! Wrong ID.");
                        menu = 8;
                        break;
                case 4:
                        System.out.println("Enter ID to change information:");
                        int id1 = inInt.nextInt();
                        int position = 0;
                        for(; position < list.getSize(); position++)
                                if(list.mas[position].getRegistrationNum() == id1)
                                        break;
                        if(position == list.getSize())
                        {
                                System.out.println("There is no challanger with that
ID.");
                                break;
                        }
                        while (!endChange)
                        {
                                System.out.println("What do you want to change?");
```

```
                                    System.out.println("1. Education");
                                    System.out.println("2. Date of dismissal");
                                    System.out.println("3. Work experience");
                                    System.out.println("4. Demands to work");
                                    System.out.println("5. Stop changing.");
                                    menuChange = inInt.nextInt();
                                    switch (menuChange)
                                    {
                                    case 1:
                                          System.out.println("Enter new education: ");

      list.mas[position].setEducation(inStr.nextLine());
                                          break;
                                    case 2:
                                          System.out.println("Enter new day: ");

      list.mas[position].setDismissalDay(inInt.nextInt());
                                          System.out.println("Enter new day: ");

      list.mas[position].setDismissalMonth(inInt.nextInt());
                                          System.out.println("Enter new day: ");

      list.mas[position].setDismissalYear(inInt.nextInt());
                                          break;
                                    case 3:
                                          System.out.println("Information   about   work
experience: ");
                                          System.out.println("1.    Specialization    of
previous job");
                                          System.out.println("2.     Experience     (in
years)");

                                          changeExperience = inInt.nextInt();
                                          switch(changeExperience) {
                                          case 1:
                                                System.out.println("Enter  new  previous
specialization: ");

      list.mas[position].getWorkExperience().setSpecialization(inStr.nextLine());
                                                break;
                                          case 2:
                                                System.out.println("Enter  new  years  of
experience: ");

      list.mas[position].getWorkExperience().setExperience(inInt.nextInt());
                                                break;
                                          default:
                                                System.out.println("Error! Wrong num in
menu.");

                                                break;
                                          }
                                          break;
                                    case 4:
                                          System.out.println("Information about demands
to work: ");
                                          System.out.println("1. Specialization of next
job");
                                          System.out.println("2. Min salary");
                                          System.out.println("3. Conditions of work");

                                          changeDemands = inInt.nextInt();
                                          switch(changeDemands) {
```

```
                                        case 1:
                                               System.out.println("Enter    new    next
specialization: ");

      list.mas[position].getDemandsToWork().setSpecialization(inStr.nextLine());
                                               break;
                                        case 2:
                                               System.out.println("Enter    new    min
salary: ");

      list.mas[position].getDemandsToWork().setMinSalary(inInt.nextInt());
                                               break;
                                        case 3:
                                               System.out.println("Enter           new
conditions of work: ");

      list.mas[position].getDemandsToWork().setConditions(inStr.nextLine());
                                               break;
                                        default:
                                               System.out.println("Error! Wrong num in
menu.");
                                               break;
                                        }
                                        break;
                                 case 5:
                                        endChange = true;
                                        break;
                                 default:
                                        System.out.println("Error!   Wrong   num   in
menu.");
                                        break;
                                 }
                          }
                          break;
                   case 5:
                          list.clear();
                          System.out.println("List was cleared.");
                          break;
                   case 6:
                          String absolutePath = new File("").getAbsolutePath();
                          File folder = new File(absolutePath);
                          File[] listFiles = folder.listFiles();
                          String filename;
                          String currentDir = absolutePath;
                          String highestDir = folder.getName();

                          boolean leave = false;
                          position = 0;
                          System.out.print("Enter XML filename: ");
                          filename = inStr.nextLine();
                          if (filename.indexOf(".xml") == -1)
                                 filename += ".xml";
                          while(!endSerialization)
                          {
                                 position = 0;
                                 System.out.println("\nCurrent path: " + currentDir);
                                 System.out.println("XML file name: " + filename);
                                 System.out.println("\nFiles and directories in this
path:");
                                 for  (position  =  0;  position  <  listFiles.length;
position++)
```

```
                                        System.out.println(position + 1 + ". " +
listFiles[position].toString().substring(currentDir.length()+1));
                                System.out.println();
                                System.out.println("Serialization menu: ");
                                System.out.println("1. Write XML file in current
directory");
                                System.out.println("2. Go up one level folder");
                                System.out.println("3. Enter the folder");
                                System.out.println("4. End of serialization");
                                System.out.print("Enter option:");
                                menuSerialization = inInt.nextInt();
                                System.out.println();
                                switch(menuSerialization)
                                {
                                case 1:
                                        endSerialization = true;
                                        break;
                                case 2:
                                        if(folder.getName().equals(highestDir))
                                        {
                                                System.out.print("This is the highest
directory.");
                                                break;
                                        }
                                        currentDir      =      currentDir.substring(0,
currentDir.indexOf(folder.getName())-1);
                                        folder = new File(currentDir);
                                        listFiles = folder.listFiles();
                                        break;
                                case 3:
                                        while(forlderSerialization)
                                        {
                                                System.out.print("Choose the number of
folder:");
                                                position = inInt.nextInt();
                                                if(!listFiles[position-1].isDirectory()
|| position < 1 || position > listFiles.length)
                                                        System.out.println("Error.   That
is not a folder.");
                                                else
                                                {
                                                        currentDir = listFiles[position-
1].toString();
                                                        System.out.println("New   current
directory:" + currentDir);
                                                        folder = new File(currentDir);
                                                        listFiles = folder.listFiles();
                                                        forlderSerialization = false;
                                                }
                                        }
                                        break;
                                case 4:
                                        System.out.println("End of serialization");
                                        leave = true;
                                        endSerialization = true;
                                        break;
                                default:
                                        System.out.println("Error!   Wrong   num   in
menu.");
                                        break;
                                }
                        }
```

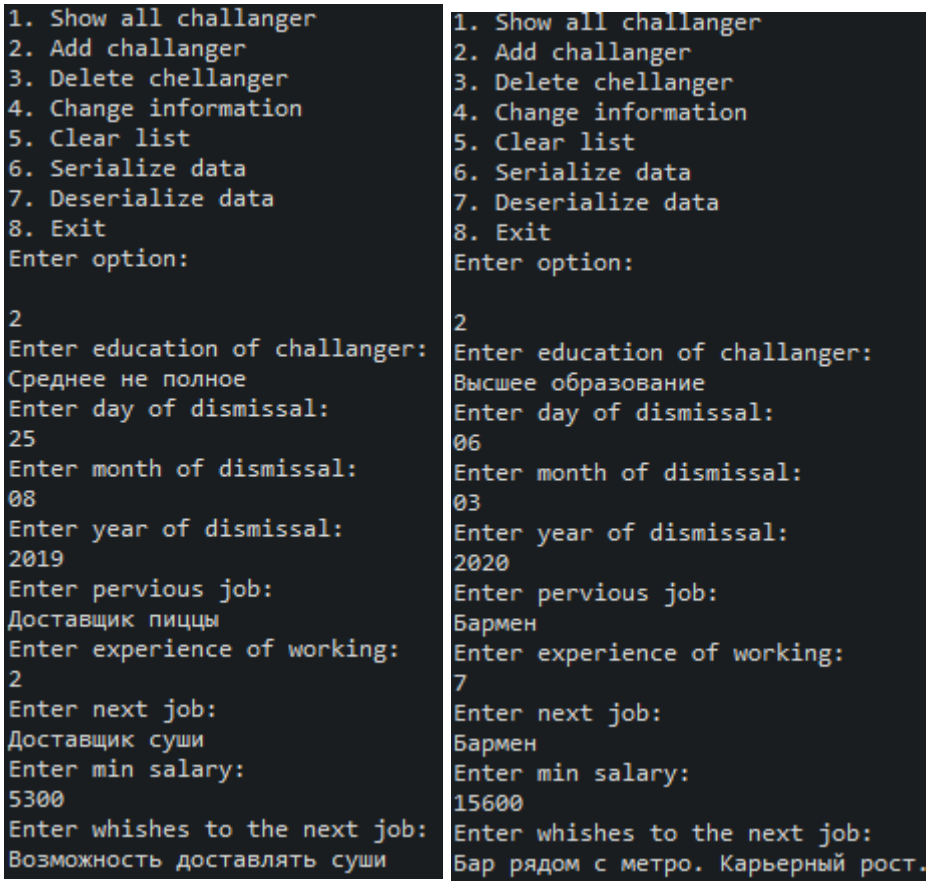
```
                        if(leave == true)
                              break;
                        absolutePath = currentDir;
                        folder = new File(absolutePath);
                        File file = new File(folder,filename);
                        try {
                              XMLEncoder    encoder    =    new    XMLEncoder(new
BufferedOutputStream(new FileOutputStream(file)));
                              encoder.writeObject(list.mas);
                              encoder.close();
                        }
                        catch (Exception e) {
                              System.out.println(e);
                              break;
                        }
                        System.out.println("File was written in this directory: " +
absolutePath);
                        System.out.println("Serialization was completed.");
                        break;
                  case 7:
                        absolutePath = new File("").getAbsolutePath();
                        folder = new File(absolutePath);
                        listFiles = folder.listFiles();
                        currentDir = absolutePath;
                        highestDir = folder.getName();
                        leave = false;
                        position = 0;

                        while(!endDeserializtion)
                        {
                              position = 0;
                              System.out.println("Current path: " + currentDir);
                              System.out.println("Files  and  directories  in  this
path:");
                              for  (position  =  0;  position  <  listFiles.length;
position++) {
                                    System.out.println(position  +  1  +  ".  "  +
listFiles[position].toString().substring(currentDir.length()+1));
                              }
                              System.out.println();
                              System.out.println("Deserialization menu:");
                              System.out.println("1.  Read  XML  file  in  current
directory");
                              System.out.println("2. Go up one level folder");
                              System.out.println("3. Enter the folder");
                              System.out.println("4. End of deserialization");
                              System.out.print("Enter option:");
                              menuDeserialization = inInt.nextInt();
                              System.out.println();
                              switch(menuDeserialization)
                              {
                              case 1:
                                    System.out.print("Enter ID of the file:");
                                    position = inInt.nextInt();
                                    if(listFiles[position-
1].getName().indexOf(".xml") == -1 || listFiles[position-1].isDirectory())
                                    {
                                          System.out.println("Error, that's not a
.XML file.");
                                          break;
                                    }
                                    endDeserializtion = true;
```

```
                                        break;
                                case 2:
                                        if(folder.getName().equals(highestDir))
                                        {
                                                System.out.println("This is the highest
directory.");
                                                break;
                                        }
                                        currentDir      =      currentDir.substring(0,
currentDir.indexOf(folder.getName())-1);
                                        folder = new File(currentDir);
                                        listFiles = folder.listFiles();
                                        break;
                                case 3:
                                        while(folderDeserialization)
                                        {
                                                System.out.print("Choose the number of
folder:");
                                                position = inInt.nextInt();
                                                if(!listFiles[position-1].isDirectory()
|| position < 1 || position > listFiles.length)
                                                        System.out.println("Error,
that's not a folder.");
                                                else
                                                {
                                                        currentDir = listFiles[position-
1].toString();
                                                        System.out.println("New   current
directory: " + currentDir);
                                                        folder = new File(currentDir);
                                                        listFiles = folder.listFiles();
                                                        folderDeserialization = false;
                                                }
                                        }
                                        break;
                                case 4:
                                        System.out.println("End of deserialization");
                                        leave = true;
                                        endDeserializtion = true;
                                        break;
                                default:
                                        System.out.println("Error!   Wrong   num   in
menu.");
                                        break;
                                }
                        }
                        if(leave == true)
                                break;
                        absolutePath  =  currentDir  +  "\\"  +  listFiles[position-
1].getName();
                        file = new File(absolutePath);
                        try
                        {
                                XMLDecoder    decoder    =    new    XMLDecoder(new
BufferedInputStream(new FileInputStream(file)));
                                list.mas = (Challanger[])decoder.readObject();
                                decoder.close();
                                list.setSize(list.mas.length);
                        }
                        catch (Exception e)
                        {
                                System.out.println(e);
```

```
                                break;
                        }
                        System.out.println("File was read from this directory: " +
listFiles[position-1]);
                        System.out.println("Deserialization was completed.");
                        break;
                case 8:
                        endprog = true;
                        inInt.close();
                        inStr.close();
                        break;
                default:
                        System.out.println("Error! Wrong num in menu.");
                        break;
                }
            }
        }
}
```

РЕЗУЛЬТУТ ВИКОНАННЯ РОБОТИ ПРОГРАММИ



а) б)

Рисунок 8.1 – Результат роботи програми

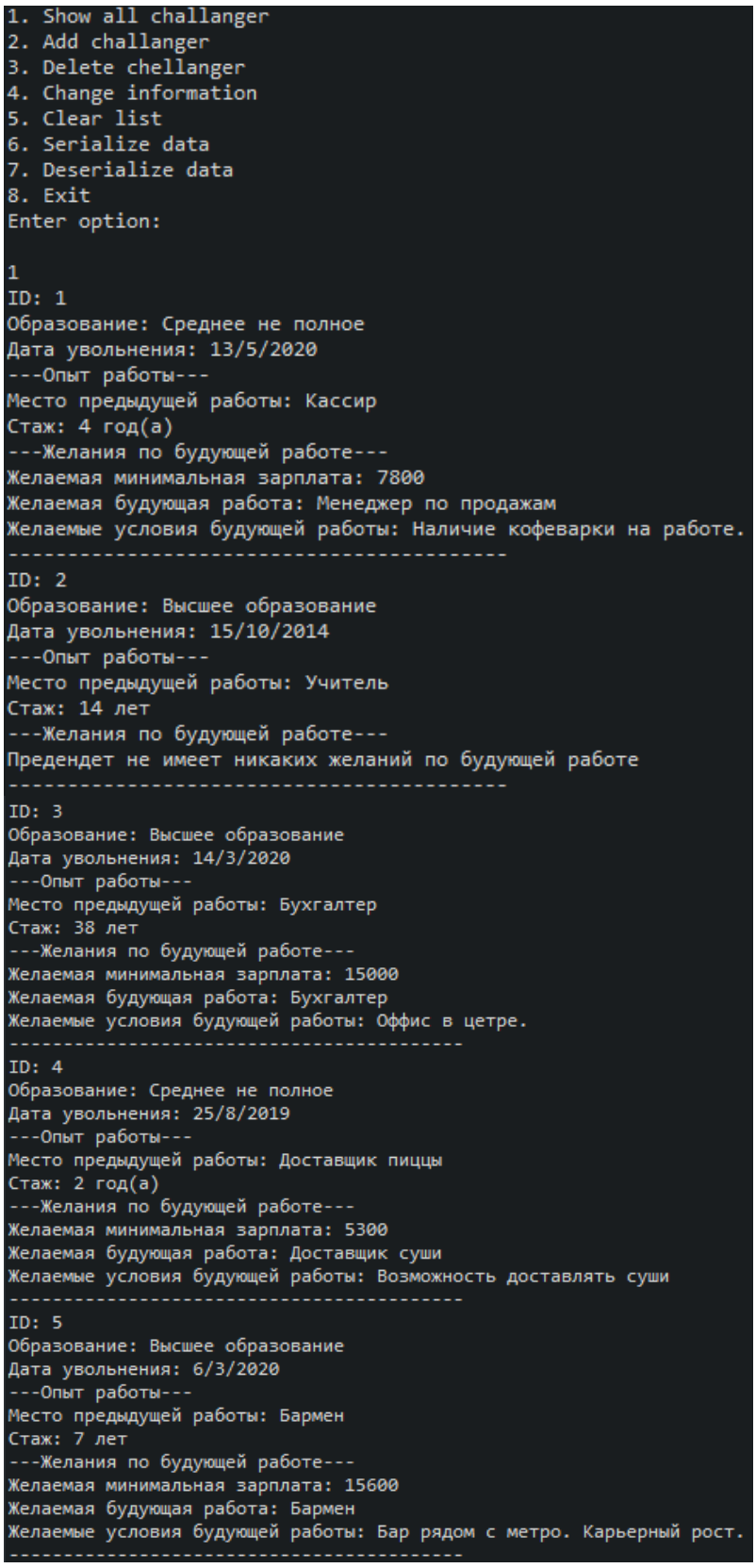


Рисунок 8.2 – Результат роботи програми

```
Enter option:

6
Enter XML filename: fiveChallangers

Current path: C:\Users\drftg\eclipse-workspace\abdullin
XML file name: fiveChallangers.xml

Files and directories in this path:
1. .classpath
2. .project
3. .settings
4. 4.xml
5. 5.xml
6. bin
7. doc
8. fivechallangers.xml
9. newclass.xml
10. newfile.xml
11. newsuper.xml
12. onemoretry.xml
13. Serialization.ser
14. src
15. third.xml
16. ua.khpi.oop.zanochkyn03.jar
17. ыусщтвекн.xml

Serialization menu:
1. Write XML file in current directory
2. Go up one level folder
3. Enter the folder
4. End of serialization
Enter option:1

File was written in this directory: C:\Users\drftg\eclipse-workspace\abdullin
Serialization was completed.
```

Рисунок 8.3 – Результат роботи програми

```
3
Enter ID to delete:
2
Challanger was deleted successfully.
1. Show all challanger
2. Add challanger
3. Delete chellanger
4. Change information
5. Clear list
6. Serialize data
7. Deserialize data
8. Exit
Enter option:

3
Enter ID to delete:
3
Challanger was deleted successfully.
1. Show all challanger
2. Add challanger
3. Delete chellanger
4. Change information
5. Clear list
6. Serialize data
7. Deserialize data
8. Exit
Enter option:

3
Enter ID to delete:
4
Challanger was deleted successfully.
```

Рисунок 8.4 – Результат роботи програми

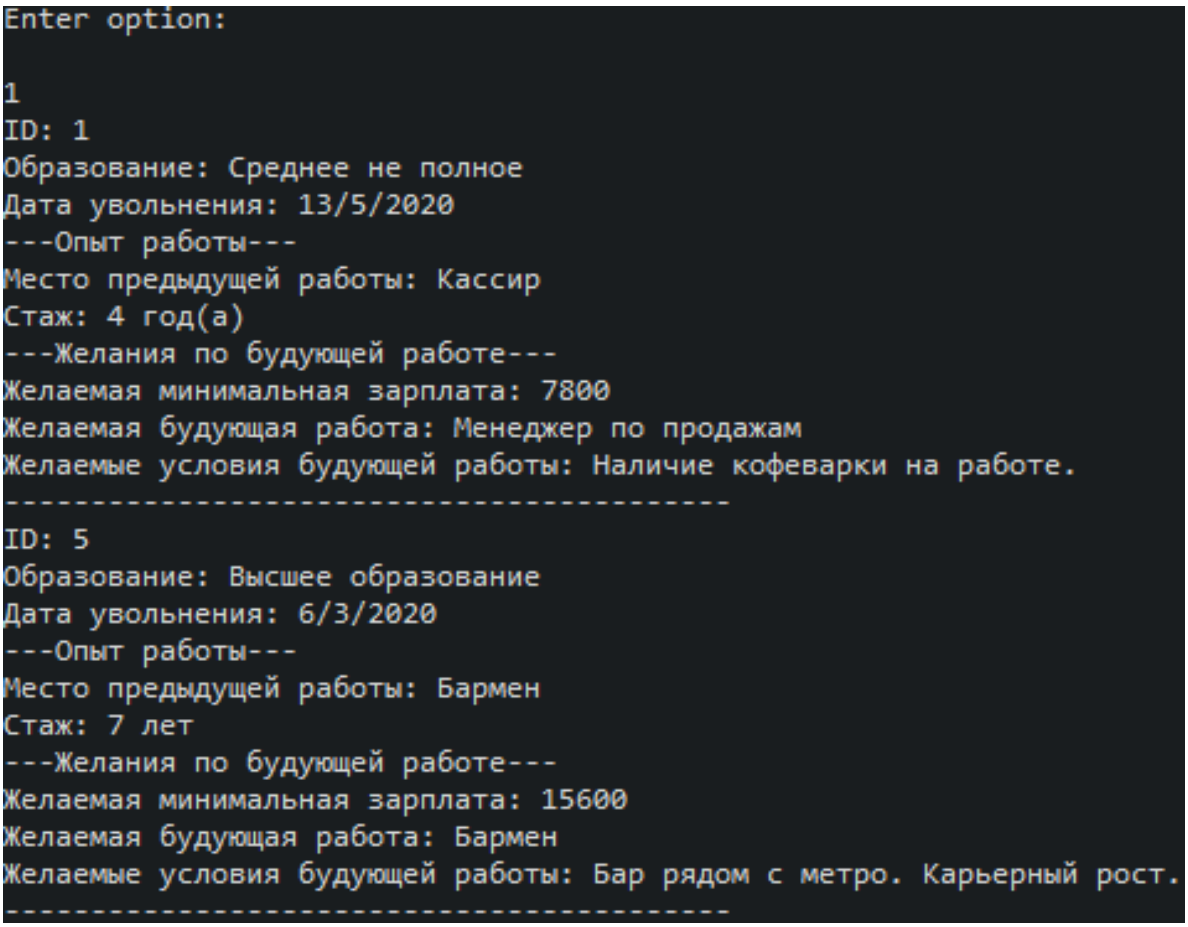


Рисунок 8.5 – Результат роботи програми

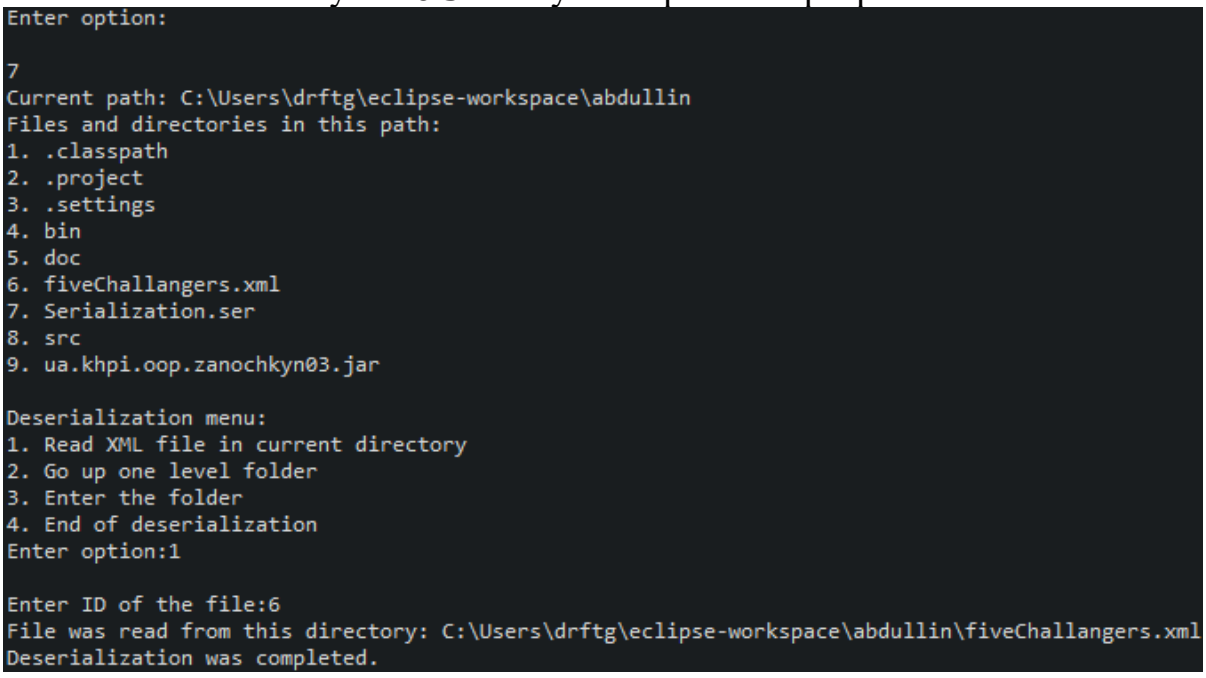


Рисунок 8.6 – Результат роботи програми

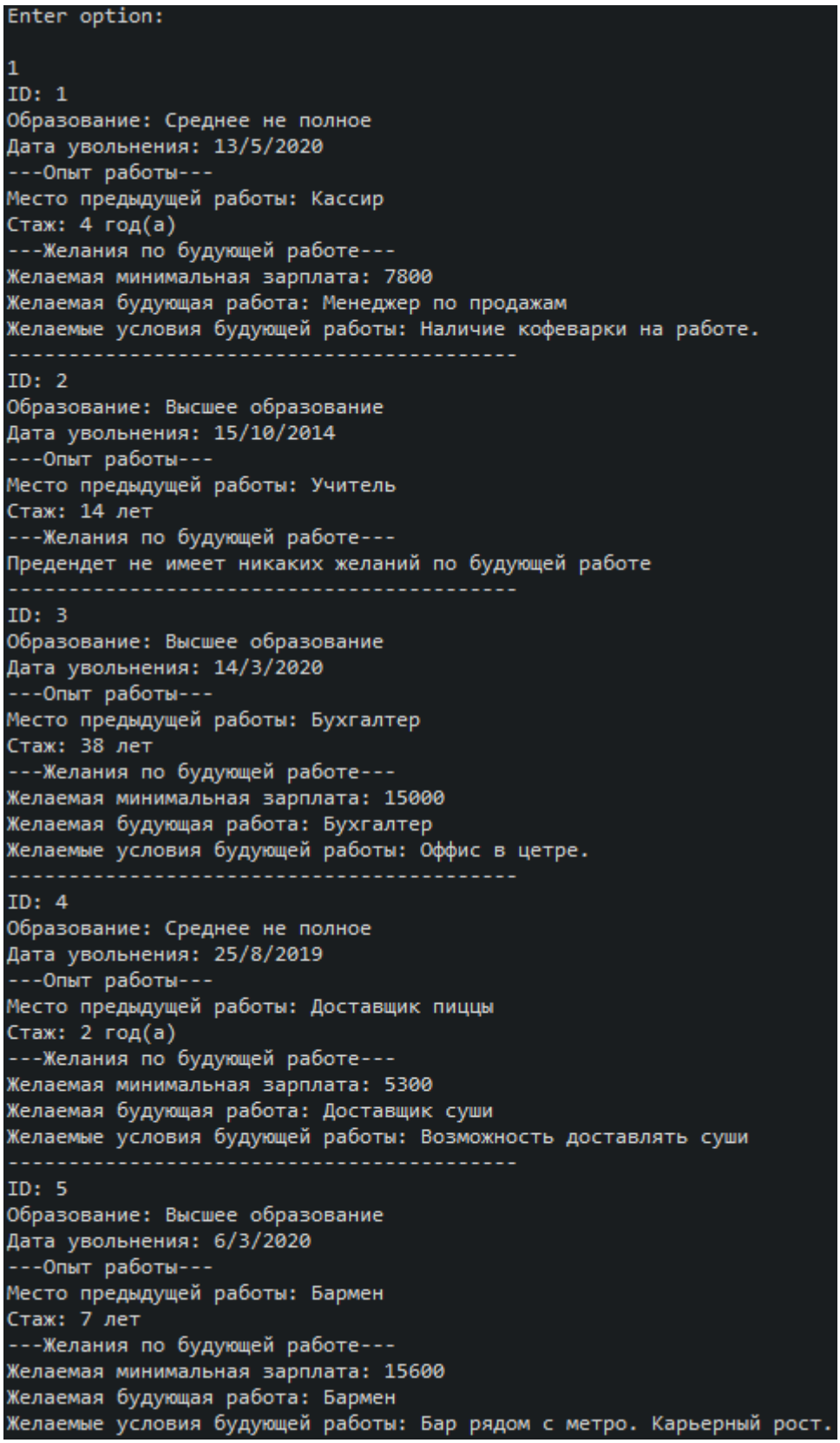


Рисунок 8.7 – Результат роботи програми

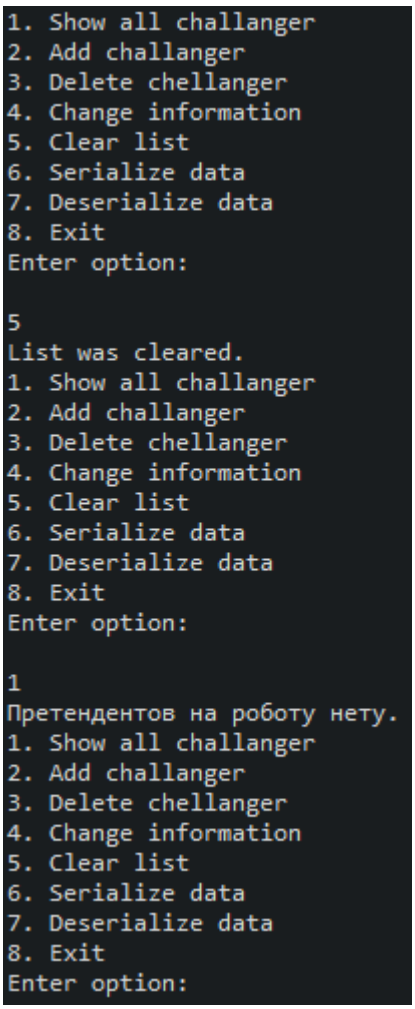


Рисунок 8.8 – Результат роботи програми

Висновок

Під час виконання лабораторної роботи було набуто навичок роботи з основами введення/виведення у середовищі Eclipse IDE.