Отчет по лабораторной работе No.3

Студент: Перхуров В.А.

Группа: ИВМ-22

1. Постановка задачи

В процессе выполнения лабораторной работы необходимо выполнить следующие задачи:

1. ООП.
    a. Создать интерфейс
    b. Создать абстрактный класс
    c. Создать класс, имплементирующий интерфейс
    d. Создать класс-наследник абстрактного класса
2. Reflection
    a. Выгрузить все поля и методы класса с помощью рефлексии
    b. Вызвать несколько методов класса
    c. Вывести на экран всех предков класса
3. Collections
    a. Ознакомится со всеми коллекциями java (list, set, map) и их реализацией
    b. Продемонстрировать в программе работу с каждым видом реализации коллекции (list, set, map)
4. Generics
    a. Сделать дженерик на класс
    b. Сделать дженерик на метод

2. Разработка задачи

2.1 Структура проекта

Проект разделен на следующие директории:

docs

Данная документация

sources

Содержит папки с настройками и исходниками:

src/main/java/ru/rsatu

Директория, где хранится main-класс.

pojo

Директория, где хранятся созданные классы.

3. Информация о реализации

3.1 ООП

Для выполнения задания по реализации ООП были созданы следующие классы:

Интерфейс IDocument. Данный интерфейс предоставляет общие для всех документов методы без реализации (получение типа, номера и даты выдачи документа). Его листинг представлен далее:

Листинг 1. Листинг интерфейса IDocument

```
package ru.rsatu.pojo;

import java.util.Date;

/**
 * Интерфейс работы с документами
 */
public interface IDocument {
    /**
     * Получить наименование документа
     * @return имя документа в строковом виде
     */
    public String getType();

    /**
     * Получить номер документа
     * @return номер документа в строковом виде
     */
    public String getNumber();

    /**
     * Получить дату выдачи документа
     * @return дата выдачи документа
     */
    public Date getDateOfIssue();
}
```

После создания интерфейса был создан абстрактный класс Document, который реализовал общие интерфейсные методы. Его листинг представлен далее:

Листинг 2. Листинг абстрактного класса Document

```
package ru.rsatu.pojo;

import java.util.Date;

/**
 * Базовая реализация интерфейса работы с документами
 */
public abstract class Document implements IDocument {
    public Document(String name, String namber, Date dateOfIssue) {
        this.name = name;
        this.namber = namber;
        this.dateOfIssue = dateOfIssue;
    }

    @Override
    public String getType() {
        return name;
    }

    @Override
    public String getNumber() {
        return namber;
    }

    @Override
    public Date getDateOfIssue() {
        return dateOfIssue;
    }


    /**
     * Наименование документа
     */
    private String name;

    /**
     * Номер документа
     */
    private String namber;

    /**
     * Дата выдачи документа
     */
    private Date dateOfIssue;
}
```

После создания общего (абстрактного) класса были созданы 3 класса, которые описывают каждый свой тип документа.

Класс Passport унаследовал общие методы для работы с документами от абстрактного класса и добавил 2 новых метода, которые позволяют указать и прочитать место прописки. Листинг класса Passport представлен ниже.

Листинг 3. Листинг класса Passport

```
package ru.rsatu.pojo;

import java.util.Date;

public class Passport extends Document{

    public Passport(String name, String namber, Date dateOfIssue) {
        super(name, namber, dateOfIssue);
    }

    /**
     * Получить текущий адрес прописки
     * @return текущий адрес
     */
    public String getResidenceAddress() {
        return residenceAddress;
    }

    /**
     * Установить новый адрес прописки
     * @param residenceAddress - новый адрес
     */
    public void setResidenceAddress(String residenceAddress) {
        this.residenceAddress = residenceAddress;
    }

    /**
     * Адрес прописки
     */
    private String residenceAddress = "БОМЖ";

}
```

Далее был создан класс SNILS. Он унаследовал общие методы для работы с документами от абстрактного класса, но своих не добавил. Листинг класса SNILS представлен ниже.

Листинг 4. Листинг класса SNILS

```
package ru.rsatu.pojo;

import java.util.Date;

public class SNILS extends Document{
    public SNILS(String name, String namber, Date dateOfIssue) {
        super(name, namber, dateOfIssue);
    }
}
```

Далее был создан класс BirthCertificate. Он унаследовал общие методы для работы с документами от абстрактного класса и добавил 2 новых метода, которые позволяют узнать имена отца и матери. Листинг класса BirthCertificate представлен ниже.

Листинг 5. Листинг класса BirthCertificate

```
package ru.rsatu.pojo;

import java.util.Date;

public class BirthCertificate extends Document{
    public BirthCertificate(String name, String namber, Date dateOfIssue) {
        this(name, namber, dateOfIssue, "-", "-");
    }
    public BirthCertificate(String name, String namber, Date dateOfIssue, String
mother, String father) {
        super(name, namber, dateOfIssue);
        this.mother = mother;
        this.father = father;
    }

    /**
     * Получить имя матери
     * @return имя матери
     */
    public String getMother() {
        return mother;
    }

    /**
     * Получить имя матери
     * @return имя матери
     */
    public String getFather() {
        return father;
    }

    /**
     * Мать
     */
    private String mother = "-";

    /**
     * Отец
     */
    private String father = "-";
}
```

3.2 Reflection

Для выполнения задания по ознакомлению с Reflection был создан отдельный класс, в рамках которого были реализованы статические методы для выполнения следующих пунктов задания:

1. Выгрузить все поля и методы класса с помощью рефлексии.
2. Вызвать несколько методов класса.
3. Вывести на экран всех предков класса.

Листинг класса представлен ниже.

Листинг 6. Листинг класса Reflection

```
package ru.rsatu.pojo;

import jdk.dynalink.Operation;

import java.lang.reflect.Field;
import java.lang.reflect.InvocationTargetException;
import java.lang.reflect.Method;

/**
 * Обёртка для проверки рефлексии
 */
public class Reflection {
    /**
     * Выгружаем все поля и методы класса
     * @param clazz - класс
     */
    static public void printAllFieldAndMethods(Class clazz) {
        System.out.println("---------------------------------------");
        System.out.println("- Выгружаем все поля и методы класса " + clazz.getName() +
" -" );
        System.out.println("---------------------------------------");
        System.out.println("- Поля:");
        Class superclass = clazz.getSuperclass();
        for (Field fld : superclass.getDeclaredFields()) {
            System.out.println(fld.getName());
        }
        System.out.println("- Методы:");
        for (Method fld : superclass.getDeclaredMethods()) {
            System.out.println(fld.getName());
        }
        System.out.println("/////////////////////////////////////");
    }

    /**
     * Выводим всех предков класса
     * @param clazz - класс
     */
    static public void printAllParents(Class clazz) {
        System.out.println( "---------------------------------------" );
        System.out.println( "- Выводим всех предков класса " + clazz.getName() + " -"
);
        System.out.println( "---------------------------------------" );
```

```
        Class superclass = clazz.getSuperclass();
        System.out.println("- Классы:");
        printParent( superclass );
        System.out.println("- Интерфейсы:");
        for (Class intrface : superclass.getInterfaces()) {
            System.out.println(intrface.getName());
        }
        System.out.println( "////////////////////////////////////" );
    }

    static private void printParent(Class clazz)
    {
        if( clazz == null ) // || Object.class.equals(clazz)
            return;
        System.out.println(clazz.getName());
        printParent(clazz.getSuperclass());
    }

    /**
     * Вызывать гетторы указанного класса
     * @param clazz - класс
     */
    static public void invokeGetMethods(IDocument doc) {
        System.out.println( "------------------------------------" );
        System.out.println( "- Вызываем часть методов класса " + doc.getClass()
.getName() + " -" );
        System.out.println( "------------------------------------" );
        for (Method method : doc.getClass().getMethods() ) {
            if( isGetter(method) ) {
                try {
                    System.out.println( "Метод '" + method.getName() + "' вернул: " +
method.invoke( doc ) );
                } catch (IllegalAccessException e) {
                    throw new RuntimeException(e);
                } catch (InvocationTargetException e) {
                    throw new RuntimeException(e);
                }
            }
        }
        System.out.println( "////////////////////////////////////" );
    }

    static private boolean isGetter(Method metod) {
        return metod.getName().startsWith("get") &&
               metod.getParameterCount() == 0 &&
               !void.class.equals(metod.getReturnType());
    }
}
```

3.3 Collections

Для выполнения задания по работе с коллекциями java в методе main были написаны примеры работы с типами list, set и map.

Контейнер list был заполнен заранее созданными объектами типа Passport, BirthCertificate и SNILS. Также туда был добавлен новый созданный обхект типа Passport. Далее в цикле было выведено в консоль содержимое контейнера.

Контейнер set был заполнен содержимым контейнера list. Также была проведена попытка добавления дубликатов объектов. Далее в цикле было выведено в консоль содержимое контейнера (дубликатов не было).

Контейнер map был заполнен содержимым контейнера list. Также была проведена попытка добавления дубликата объекта по ключу. Далее в цикле было выведено в консоль содержимое контейнера (дубликатов не было).

Пример работы с данными контейнерами представлен ниже.

Листинг 7. Листинг шабонного класса Pair

```
...
System.out.println("\n// 3 - Collection");
System.out.println("Работа с коллекцией List");
List<Document> list_of_docs = new ArrayList<>();
list_of_docs.add( passport );
list_of_docs.add( birthCertificate );
list_of_docs.add( snils );
list_of_docs.add(
    new Passport(
        "Паспорт гражданина Республики Беларусь",
        "MC1100586",
        new Date(101, 9, 11)
    )
);
for ( Document doc : list_of_docs ) {
    System.out.println( "-----------------------------------" );
    System.out.println( "Тип документа: " + doc.getType() );
    System.out.println( "Номер документа: " + doc.getNumber() );
    System.out.println( "///////////////////////////////////" );
}

System.out.println();
System.out.println("Работа с коллекцией Set");
Set<Document> set_of_docs = new HashSet<>();
set_of_docs.add( passport );
set_of_docs.addAll( list_of_docs );
set_of_docs.add( birthCertificate );
for ( Document doc : set_of_docs ) {
    System.out.println( "-----------------------------------" );
    System.out.println( "Тип документа: " + doc.getType() );
    System.out.println( "Номер документа: " + doc.getNumber() );
    System.out.println( "///////////////////////////////////" );
}

System.out.println();
System.out.println("Работа с коллекцией Map");
Map<String, Document> map_of_docs = new HashMap<>();
map_of_docs.put( snils.getNumber(), snils );
for ( Document doc : set_of_docs ) {
    map_of_docs.put( doc.getNumber(), doc );
}
map_of_docs.put( passport.getNumber(), passport );
map_of_docs.forEach((key, value) -> {
    System.out.println("-----------------------------------");
    System.out.println("Тип документа: " + value.getType());
    System.out.println("Номер документа: " + key);
    System.out.println( "///////////////////////////////////" );
});
...
```

3.4 Generics

Для выполнения задания по созданию шаблонных (Generics) метода и класса были созданы отдельный класс Pair и метод print.

Класс Pair хранит 2 поля и предоставляет доступ к ним посредством соответствующих геттеров и сеттеров. Также в данном классе был переопределён метод toString так, чтобы он выводил содержимое в виде строки. Листинг шаблонного класса представлен ниже.

Листинг 8. Листинг шабонного класса Pair

```
package ru.rsatu.pojo;

/**
 * Шаблонный класс пары
 * @param <First> тип левого значения
 * @param <Second> тип правого значения
 */
public class Pair<First,Second> {
    public Pair() {
    }
    public Pair(First first, Second second) {
        this.first = first;
        this.second = second;
    }


    public First getFirst() {
        return first;
    }

    public void setFirst(First first) {
        this.first = first;
    }

    public Second getSecond() {
        return second;
    }

    public void setSecond(Second second) {
        this.second = second;
    }

    @Override
    public String toString() {
        return "{\'" + first.toString() + "\', \'" + second.toString() + "\'}";
    }

    /**
     * Левое значение
     */
    private First first;
    /**
     * Правое занчение
     */
    private Second second;
}
```

Метод print принимает значение указанного типа и выводит его в консоль посредством

вызова метода toString. Листинг шаблонного метода представлен ниже.

Листинг 9. Листинг шабонного метода print

```
public static <T> void print(T value) {
    System.out.println("Было передано значение вида: " + value.toString());
}
```

4. Результаты выполнения

В результате выполнения задания было:

- Создано 5 классов:
  - 1 интерфейс (IDocument).
  - 1 абстрактный класс (Document), унаследованный от интерфейса IDocument и имлементирующий его.
  - 3 класса-наследника (наследовались от абстрактного класса Document).
- Ознакомился с Reflection:
  - С помощью рефлексии были выгружены все поля и методы созданных классов.
  - С помощью рефлексии были вызваны несколько методов переданного класса.
  - С помощью рефлексии были выведены на экран все предки переданного класса.
- Ознакомился со всеми коллекциями java (list, set, map) и их реализацией:
  - Была продемонстрирована работа с каждым видом реализации коллекции (list, set, map).
- Ознакомился с Generics:
  - Был создан дженерик на класс.
  - Был создан дженерик на методы.

Результатом работы программы является вывод следующий информации в консоль:

```
/usr/lib/jvm/java-19-jdk/bin/java
-javaagent:/usr/share/idea/lib/idea_rt.jar=40931:/usr/share/idea/bin
-Dfile.encoding=UTF-8 -Dsun.stdout.encoding=UTF-8 -Dsun.stderr.encoding=UTF-8
-classpath
/mnt/7999679B46EE88E1/4_Projects/modern_technologies_of_industrial_software_developmen
t/lr_3/sources/target/classes ru.rsatu.Main

// 1 - ООП
Список документов:
Паспорт гражданина РФ / 58 64 563145 / 2008-04-20
Свидетельство о рождении / 567139 / 1994-04-15
СНИЛС / 681-254-936-85 / 1994-04-26
```

```
// 2 - Reflection
-------------------------------------
- Выгружаем все поля и методы класса ru.rsatu.pojo.BirthCertificate -
-------------------------------------
- Поля:
name
namber
dateOfIssue
- Методы:
getType
getNumber
getDateOfIssue
/////////////////////////////////////
-------------------------------------
- Выводим всех предков класса ru.rsatu.pojo.SNILS -
-------------------------------------
- Классы:
ru.rsatu.pojo.Document
java.lang.Object
- Интерфейсы:
ru.rsatu.pojo.IDocument
/////////////////////////////////////
-------------------------------------
- Вызываем часть методов класса ru.rsatu.pojo.Passport -
-------------------------------------
Метод 'getResidenceAddress' вернул: БОМЖ
Метод 'getType' вернул: Паспорт гражданина РФ
Метод 'getNumber' вернул: 58 64 563145
Метод 'getDateOfIssue' вернул: Sun Apr 20 00:00:00 MSD 2008
Метод 'getClass' вернул: class ru.rsatu.pojo.Passport
/////////////////////////////////////

// 3 - Collection
Работа с коллекцией List
---------------------------------
Тип документа: Паспорт гражданина РФ
Номер документа: 58 64 563145
/////////////////////////////////
---------------------------------
Тип документа: Свидетельство о рождении
Номер документа: 567139
/////////////////////////////////
---------------------------------
Тип документа: СНИЛС
Номер документа: 681-254-936-85
/////////////////////////////////
---------------------------------
Тип документа: Паспорт гражданина Республики Беларусь
Номер документа: MC1100586
/////////////////////////////////
```

```
Работа с коллекцией Set
---------------------------------
Тип документа: Паспорт гражданина РФ
Номер документа: 58 64 563145
/////////////////////////////////
---------------------------------
Тип документа: Свидетельство о рождении
Номер документа: 567139
/////////////////////////////////
---------------------------------
Тип документа: Паспорт гражданина Республики Беларусь
Номер документа: MC1100586
/////////////////////////////////
---------------------------------
Тип документа: СНИЛС
Номер документа: 681-254-936-85
/////////////////////////////////

Работа с коллекцией Map
---------------------------------
Тип документа: Паспорт гражданина Республики Беларусь
Номер документа: MC1100586
/////////////////////////////////
---------------------------------
Тип документа: СНИЛС
Номер документа: 681-254-936-85
/////////////////////////////////
---------------------------------
Тип документа: Свидетельство о рождении
Номер документа: 567139
/////////////////////////////////
---------------------------------
Тип документа: Паспорт гражданина РФ
Номер документа: 58 64 563145
/////////////////////////////////

//4 - Generic
Работа с шаблонным методом и классом:
Было передано значение вида: {'10', '20'}
Было передано значение вида: {'ten', '20'}

Process finished with exit code 0
```

Данный вывод был получен после выполнения основного метода программы, код которой приведён ниже:

Листинг 10. Листинг main-класса

```
package ru.rsatu;

import ru.rsatu.pojo.*;
```

```
import javax.print.Doc;
import java.lang.reflect.InvocationTargetException;
import java.text.SimpleDateFormat;
import java.util.*;

public class Main {
    public static void main(String[] args) throws InvocationTargetException,
IllegalAccessException {
        // 1 - ООП
        System.out.println("\n// 1 - ООП");
        Document passport = new Passport(
            "Паспорт гражданина РФ",
            "58 64 563145",
            new Date(108, 3, 20)
        );
        Document birthCertificate = new BirthCertificate(
            "Свидетельство о рождении",
            "567139",
            new Date(94, 3, 15),
            "Иванова М.Д.",
            "Иванов С.В."
        );
        Document snils = new SNILS(
            "СНИЛС",
            "681-254-936-85",
            new Date(94, 3, 26)
        );
        SimpleDateFormat df = new SimpleDateFormat("yyyy-MM-dd");
        System.out.println(
            "Список документов:" +
            "\n" + passport.getType()         + " / "  + passport.getNumber()
+ " / "  + df.format( passport.getDateOfIssue() ) +
            "\n" + birthCertificate.getType() + " / "  + birthCertificate.getNumber()
+ " / "  + df.format( birthCertificate.getDateOfIssue() ) +
            "\n" + snils.getType()            + " / "  + snils.getNumber()
+ " / "  + df.format( snils.getDateOfIssue() )
        );


        // 2 - Reflection
        System.out.println("\n// 2 - Reflection");
        Reflection.printAllFieldAndMethods( BirthCertificate.class );
        Reflection.printAllParents( SNILS.class );
        Reflection.invokeGetMethods( passport );

        // 3 - Collection
        System.out.println("\n// 3 - Collection");
        System.out.println("Работа с коллекцией List");
        List<Document> list_of_docs = new ArrayList<>();
        list_of_docs.add( passport );
```

```
        list_of_docs.add( birthCertificate );
        list_of_docs.add( snils );
        list_of_docs.add(
            new Passport(
                "Паспорт гражданина Республики Беларусь",
                "MC1100586",
                new Date(101, 9, 11)
            )
        );
        for ( Document doc : list_of_docs ) {
            System.out.println( "----------------------------------" );
            System.out.println( "Тип документа: " + doc.getType() );
            System.out.println( "Номер документа: " + doc.getNumber() );
            System.out.println( "//////////////////////////////////" );
        }

        System.out.println();
        System.out.println("Работа с коллекцией Set");
        Set<Document> set_of_docs = new HashSet<>();
        set_of_docs.add( passport );
        set_of_docs.addAll( list_of_docs );
        set_of_docs.add( birthCertificate );
        for ( Document doc : set_of_docs ) {
            System.out.println( "----------------------------------" );
            System.out.println( "Тип документа: " + doc.getType() );
            System.out.println( "Номер документа: " + doc.getNumber() );
            System.out.println( "//////////////////////////////////" );
        }

        System.out.println();
        System.out.println("Работа с коллекцией Map");
        Map<String, Document> map_of_docs = new HashMap<>();
        map_of_docs.put( snils.getNumber(), snils );
        for ( Document doc : set_of_docs ) {
            map_of_docs.put( doc.getNumber(), doc );
        }
        map_of_docs.put( passport.getNumber(), passport );
        map_of_docs.forEach((key, value) -> {
            System.out.println("----------------------------------");
            System.out.println("Тип документа: " + value.getType());
            System.out.println("Номер документа: " + key);
            System.out.println( "//////////////////////////////////" );
        });

        // 4 - Generic
        System.out.println("\n//4 - Generic");
        System.out.println("Работа с шаблонным методом и классом:");
        Pair<Integer,Integer> int_to_int_pair = new Pair<>(10, 20);
        print( int_to_int_pair );
        Pair<String,Integer> str_to_int_pair = new Pair<>("ten", 20);
        print( str_to_int_pair );
```

```
    }

    public static <T> void print(T value) {
        System.out.println("Было передано значение вида: " + value.toString());
    }

}
```

5. Вывод

В ходе выполнения лабораторной работы были получены базовые навыки работы с ООП, Reflection, Generics и коллекциями (list, set, map).