Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

ИРКУТСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

|  |
| --- |
| Институт информационных технологий и анализа данных |

наименование института

ОТЧЕТ

к лабораторной работе №4 по дисциплине:

|  |
| --- |
| «Объектно-ориентированное программирование» |
| «Разработка классов с использованием механизмов  наследования, полиморфизма и инкапсуляции» |

наименование темы

Вариант индивидуальный

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Выполнил студент группы: | ИСТб-19-1 |  |  |  | Балдаев В.В. |
|  | шифр группы |  | подпись |  | Фамилия ИО |
| Проверил: | доцент |  |  |  | Маланова Т.В. |
|  | должность |  | подпись |  | Фамилия ИО |

Иркутск. 2020 г.

**Содержание**

[1 Постановка задачи 3](#_Toc59117204)

[2 Проектирование классов 4](#_Toc59117205)

[3 Структура меню 6](#_Toc59117206)

[4 Спецификации 7](#_Toc59117207)

[4.1 Класс Main 7](#_Toc59117208)

[4.2 Класс mainmenu 7](#_Toc59117209)

[4.3 Класс Info 7](#_Toc59117210)

[4.4 Класс Note 7](#_Toc59117211)

[4.5 Класс JustNote 8](#_Toc59117212)

[4.6 Класс FavorNote 9](#_Toc59117213)

[4.7 Класс AddressNote 10](#_Toc59117214)

[5 Описание методов классов и спецификации локальных переменных 10](#_Toc59117215)

[6 Таблица тестов 11](#_Toc59117216)

[7 Результаты тестирования 11](#_Toc59117217)

[8 Исходный код 14](#_Toc59117218)

[Список использованных источников 20](#_Toc59117219)

1 Постановка задачи

В соответствии с индивидуальным заданием описать иерархию классов, для каждого класса описать поля и соответствующие методы доступа к ним.

В зависимости от задания некоторые из этих классов (как минимум один) являются абстрактными классами и служат для выделения общих данных и поведения для других классов. Абстрактный класс должен содержать как минимум один абстрактный метод, реализация которого у его наследников должна различаться.

Помимо этого, в общую часть задания входит разработка класса, группирующего объекты описанных в соответствии с заданием классов.

Для выполнения задания необходимо создать некоторое количество объектов, добавить их в группу используя предусмотренные методы класса «группы» и для каждого из них вызвать унаследованный метод.

Индивидуальное задание

Реализовать следующие классы:

Заметка, заметка с адресом, избранная пометка, текстовая заметка.

Разработать класс-группу. Описать иерархию данных классов.

2 Проектирование классов

На рисунке 1 представлена UML диаграмма классов.

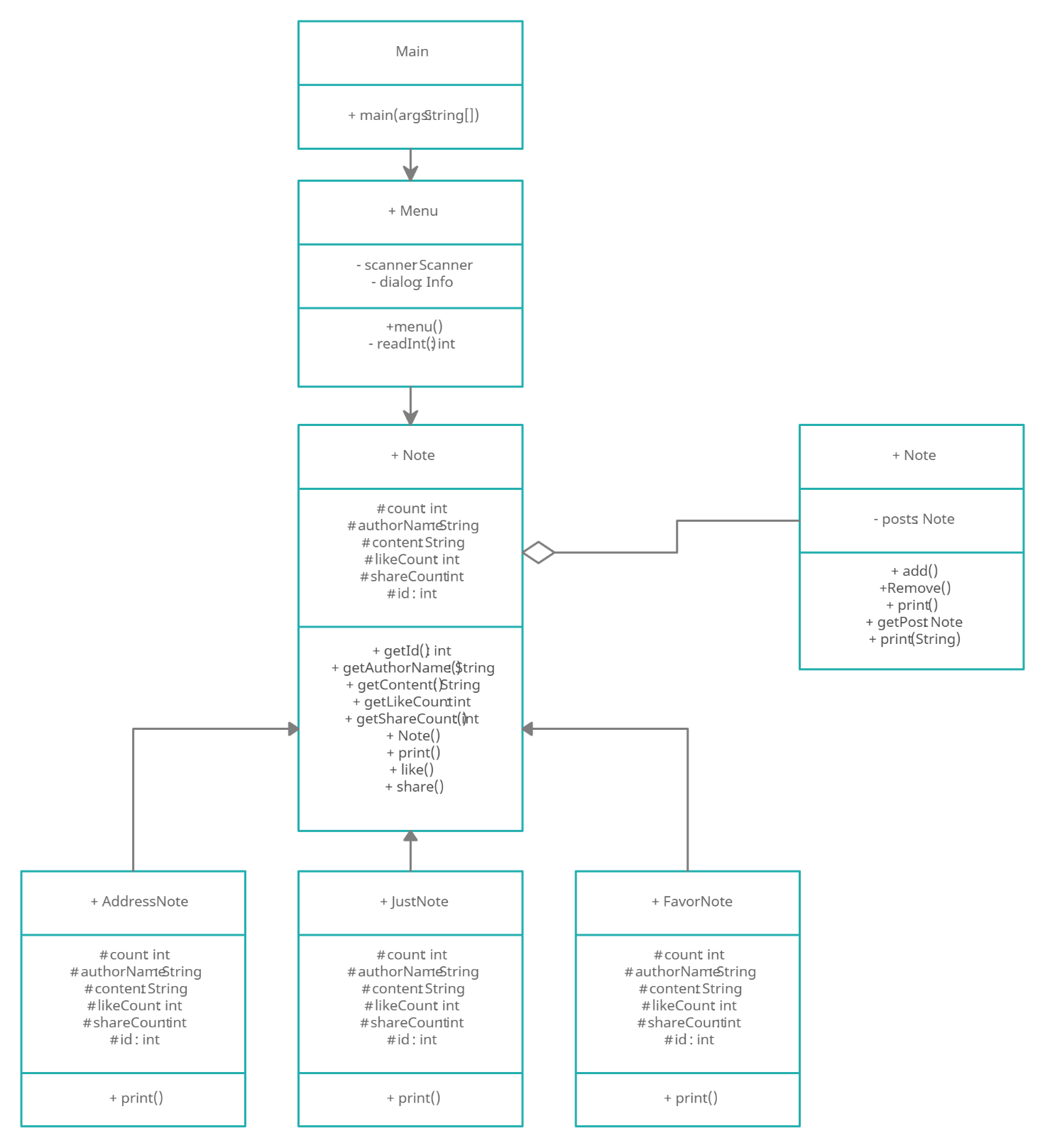


Рисунок 1 – UML диаграмма классов

Main – Точка входа в программу

mainmenu – Пользовательский псевдо интерфейс

Note – Абстрактный родительский класс записки

Info – Класс-группа для работы со списком заметок

JustNote – Класс-наследник «Обычная заметка»

FavorNote – Класс-наследник «Избранная заметка»

AddressNote – Класс-наследник «Заметка с выделенным адресом»

3 Структура меню

На рисунке 2 представлена структура пользовательского меню.

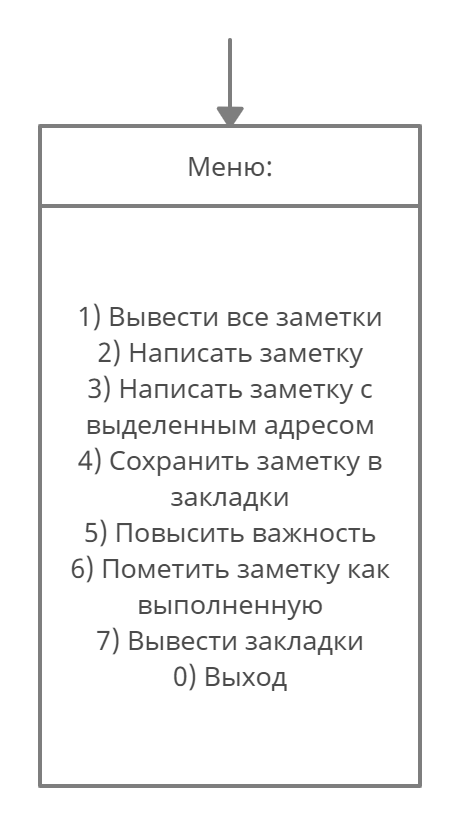


Рисунок 2 – Структура пользовательского меню

4 Спецификации

# 4.1 Класс Main

Таблица 1 – Таблица методов

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Имя** | **Тип** | **Статический** | **Уровень доступа** | **Аргументы** | **Назначение** |
| main | void | Да | public | String[] args | Точка входа в  программу |

# Класс mainmenu

Этот класс реализует пользовательский интерфейс меню.

Таблица 2 – Таблица полей

|  |  |  |  |
| --- | --- | --- | --- |
| **Имя** | **Тип** | **Уровень доступа** | **Назначение** |
| scanner | Scanner | private | Вывод информации в консоль |
| dialog | Info | private | Экземпляр класса-группы |

Таблица 3 – Таблица методов

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Имя** | **Тип** | **Стати**  **ческий** | **Уровень доступа** | **Аргументы** | **Назначение** |
| menu | void | Да | public |  | Меню первого уровня |
| readInt | int | Да | private |  | Чтения и проверка введённой команды |

# Класс Info

Класс-группа заметки

Таблица 4 – Таблица полей

|  |  |  |  |
| --- | --- | --- | --- |
| **Имя** | **Тип** | **Уровень доступа** | **Назначение** |
| posts | Note | private | Список всех заметок |

Таблица 5 – Таблица методов

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Имя** | **Тип** | **Стати**  **ческий** | **Уровень доступа** | **Аргументы** | **Назначение** |
| GetPost | int | Нет | public |  | Получение заметки |
| add | void | Нет | public | post | Добавление новой заметки |
| Remove | void | Нет | public | int id | Удаление заметки |
| print | void | Нет | public |  | Вывод всех заметок |
| print | void | Нет | public | String author | Вывод всех заметок одного автора |

# Класс Note

Абстрактный класс записок

Таблица 6 – Таблица полей

|  |  |  |  |
| --- | --- | --- | --- |
| **Имя** | **Тип** | **Уровень доступа** | **Назначение** |
| count | int | protected | Номер заметки |
| authorName | String | protected | Автор |
| likeCount | int | protected | Количество плюсов рейтинга |
| shareCount | int | protected | Количество добавивших в закладки |
| content | String | protected | Текст заметки |
| id | int | protected | Айди заметки |

Таблица 7 – Таблица конструкторов

|  |  |  |
| --- | --- | --- |
| **Уровень**  **доступа** | **Аргументы** | **Описание** |
| public | String author,  String content,  String address | Конструктор |

Таблица 8 – Таблица методов

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Имя** | **Тип** | **Стати**  **ческий** | **Уровень доступа** | **Аргументы** | **Назначение** |
| getId | int | Нет | public |  | Получение id |
| getAuthorName | String | Нет | public |  | Получение имени автора |
| getContent | String | Нет | public |  | Получение текста |
| getLikeCount | int | Нет | public |  | Получение кол-ва рейтинга |
| getShareCount | int | Нет | public |  | Получение кол-ва помещений в закладки |
| print | void | Нет | public |  | Вывод заметки в консоль |
| share |  | Нет | public |  | id |

# Класс JustNote

Класс-наследник Просто заметка

Таблица 9 – Таблица полей

|  |  |  |  |
| --- | --- | --- | --- |
| **Имя** | **Тип** | **Уровень доступа** | **Назначение** |
| id | int | protected | id заметки |
| authorName | String | protected | Имя автора |
| content | String | protected | Текст заметки |
| likeCount | int | protected | Количество плюсов рейтинга |
| shareCount | int | protected | Количество закладок |
| count | int | protected | Счётчик для id |

Таблица 10 – Таблица конструкторов

|  |  |  |
| --- | --- | --- |
| **Уровень**  **доступа** | **Аргументы** | **Описание** |
| public | String author,  String content,  String address | Конструктор |

Таблица 11 – Таблица методов

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Имя** | **Тип** | **Стати**  **ческий** | **Уровень доступа** | **Аргументы** | **Назначение** |
| print | void | Нет | public |  | Вывод заметки в консоль |

# Класс FavorNote

Класс-наследник Заметка-закладка

Таблица 12 – Таблица полей

|  |  |  |  |
| --- | --- | --- | --- |
| **Имя** | **Тип** | **Уровень доступа** | **Назначение** |
| id | int | protected | id заметки |
| authorName | String | protected | Имя автора |
| content | String | protected | Текст заметки |
| likeCount | int | protected | Количество плюсов рейтинга |
| shareCount | int | protected | Количество закладок |
| count | int | protected | Счётчик для id |

Таблица 13 – Таблица конструкторов

|  |  |  |
| --- | --- | --- |
| **Уровень**  **доступа** | **Аргументы** | **Описание** |
| public | String author,  String content,  String address | Конструктор |

Таблица 14 – Таблица методов

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Имя** | **Тип** | **Стати**  **ческий** | **Уровень доступа** | **Аргументы** | **Назначение** |
| print | void | Нет | public |  | Вывод заметки в консоль |

# Класс AddressNote

Класс-наследник Заметка с выделенным адресом

Таблица 15 – Таблица полей

|  |  |  |  |
| --- | --- | --- | --- |
| **Имя** | **Тип** | **Уровень доступа** | **Назначение** |
| id | int | protected | id заметки |
| authorName | String | protected | Имя автора |
| content | String | protected | Текст заметки |
| likeCount | int | protected | Количество плюсов рейтинга |
| shareCount | int | protected | Количество закладок |
| count | int | protected | Счётчик для id |

Таблица 16 – Таблица конструкторов

|  |  |  |
| --- | --- | --- |
| **Уровень**  **доступа** | **Аргументы** | **Описание** |
| public | String author,  String content,  String address | Конструктор |

Таблица 17 – Таблица методов

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Имя** | **Тип** | **Стати**  **ческий** | **Уровень доступа** | **Аргументы** | **Назначение** |
| print | void | Нет | public |  | Вывод заметки в консоль |

5 Описание методов классов и спецификации локальных переменных

Таблица 18 – Таблица локальных переменных

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменной | Описание переменных | Тип данных |
| **public class Menu Пользовательский интерфейс** | | | |
| 1 | id | Номер заметки для удаления | Int |
| 2 | readed | Команда меню для проверки | Int |
| 3 | choice | Команда меню | Int |
| 4 | author | Имя автора | String |
| 5 | likeCount | Количество плюсов рейтинга | Int |
| 6 | shareCount | Количество помещений в закладки | Int |
| public class AddressNote, JustNote, FavorNote | | | |
| 1 | stringBuilder | Строка на вывод в консоль | StringBuilder |

6 Таблица тестов

Таблица 19 – Проектирование тестов

|  |  |  |
| --- | --- | --- |
| **№** | **Действия** | **Результат** |
| 1 | Выбор несуществующего варианта действия | Сообщение о  некорректном вводе |
| 2 | Выбор раздела «Написать заметку» | Переход к созданию заметки |
| 3 | Выбор раздела «Написать заметку с выделенным адресом» | Переход к созданию заметки |
| 4 | Выбор раздела «Повысить важность» | Переход к вводу данных нужной заметки |
| 5 | Выбор раздела «Пометить заметку как выполненную» | Заметка удалена |
| 6 | Выбор раздела «Вывести закладки» | Вывод всех помеченных заметок |
| 7 | Выбор раздела «Вывести все заметки» | Вывод всех заметок |
| 8 | Выход | Завершение программы |

7 Результаты тестирования

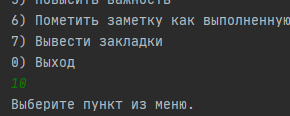


Рисунок 3 – Результат теста 1

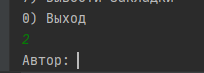


Рисунок 4 – Результат теста 2

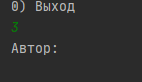


Рисунок 5 – Результат теста 3

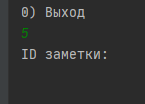


Рисунок 6 – Результат теста 4

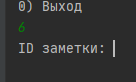


Рисунок 7 – Результат теста 5

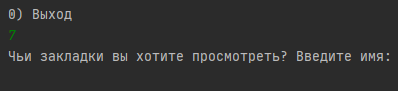


Рисунок 8 – Результат теста 6

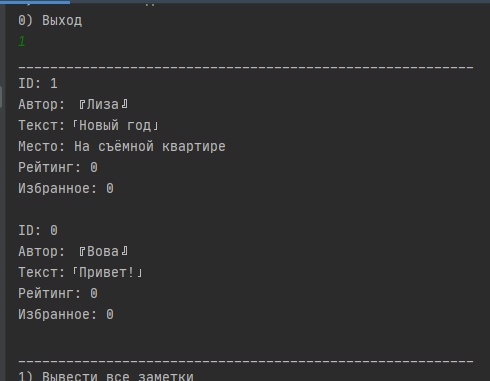


Рисунок 9 – Результат теста 7

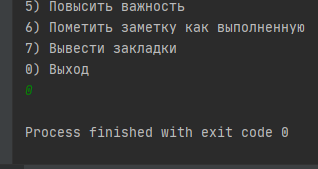


Рисунок 10 – Результат теста 8

8 Исходный код

package com.company;  
  
public class Main {  
 public static void main(String[] args) {mainmenu.menu();}  
   
}

**mainmenu**

package com.company;  
  
import java.util.Scanner;  
  
public class mainmenu {  
 private static final Info dialog;  
 private static final Scanner scanner = new Scanner(System.in);  
  
  
 static{  
 dialog = new Info();  
 dialog.add(new JustNote("Вова", "Привет!"));  
 dialog.add(new AddressNote( "Лиза", "Новый год", "На съёмной квартире"));  
 }  
  
 public static void menu(){  
 int choice = -1;  
 do{  
 System.out.println("1) Вывести все заметки");  
 System.out.println("2) Написать заметку");  
 System.out.println("3) Написать заметку с выделенным адресом");  
 System.out.println("4) Сохранить заметку в закладки");  
 System.out.println("5) Повысить важность");  
 System.out.println("6) Пометить заметку как выполненную");  
 System.out.println("7) Вывести закладки");  
 System.out.println("0) Выход");  
  
 String author;  
 String content;  
 String address;  
 int id;  
 Note srcPost;  
  
  
 choice = readInt();  
 switch (choice){  
 case 1:  
 System.out.println("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");  
 dialog.print();  
 System.out.println("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");  
 break;  
 case 2:  
 System.out.print("Автор: ");  
 author = scanner.nextLine();  
 System.out.print("Текст: ");  
 content = scanner.nextLine();  
 dialog.add(new JustNote( author, content));  
 break;  
 case 3:  
 System.out.print("Автор: ");  
 author = scanner.nextLine();  
 System.out.print("Текст: ");  
 content = scanner.nextLine();  
 System.out.print("Адрес: ");  
 address = scanner.nextLine();  
 dialog.add(new AddressNote( author, content, address));  
 break;  
 case 4:  
 System.out.print("Автор: ");  
 author = scanner.nextLine();  
 System.out.print("Текст: ");  
 content = scanner.nextLine();  
 System.out.print("ID заметки: ");  
 id = readInt();  
  
 srcPost = dialog.getPost(id);  
 if (srcPost != null) {  
 if(!(srcPost instanceof FavorNote)){  
 srcPost.share();  
 srcPost.like();  
 dialog.add(new FavorNote(author, content, srcPost));  
 }else{  
 System.out.println("Вы чего? Нельзя выбирать уже помеченное!");  
 }  
 }  
 else  
 System.out.println("Нет такой заметки!");  
 break;  
 case 5:  
 System.out.print("ID заметки: ");  
 id = readInt();  
 srcPost = dialog.getPost(id);  
 if (srcPost != null) {  
 srcPost.like();  
 }  
 else  
 System.out.println("Нет такой заметки");  
 break;  
 case 6:  
 System.out.print("ID заметки: ");  
 id = readInt();  
 dialog.Remove(id);  
 break;  
 case 7:  
 System.out.print("Чьи закладки вы хотите просмотреть? Введите имя: ");  
 author = scanner.nextLine();  
 dialog.print(author);  
 break;  
 case 0:  
  
 break;  
 default:  
 System.out.println("Выберите пункт из меню.");  
 }  
 }while (choice != 0);  
 }  
  
 private static int readInt() {  
 int readed = -1;  
 try{  
 readed = Integer.parseInt(scanner.nextLine());  
 }catch(NumberFormatException ex)  
 {  
 System.out.println("Вводите только цифры");  
 readed = readInt();  
 }  
 return readed;  
 }  
}

**Info**

package com.company;  
  
import java.util.ArrayList;  
  
public class Info {  
  
 private final ArrayList<Note> posts = new ArrayList();  
  
 public void add(Note post) {  
 this.posts.add(post);  
 }  
 public void Remove(int id){  
 posts.remove(id);  
 }  
  
 public void print(){  
 for (int i = posts.size()-1; i >= 0; i--)  
 {  
 posts.get(i).print();  
 }  
 }  
  
 public Note getPost(int id) {  
 for(Note post: posts){  
 if (post.getId() == id)  
 {  
 return post;  
 }  
 }  
 return null;  
 }  
  
 public void print(String author) {  
 for (Note post : posts)  
 {  
 if (post.getAuthorName().equals(author) && post instanceof FavorNote) post.print();  
 }  
 }  
}

**Note**

package com.company;  
  
public abstract class Note {  
 protected static int count = 0;  
 protected String authorName;  
 protected String content;  
 protected int likeCount;  
 protected static int shareCount;  
 protected int id;  
  
 public int getId() {  
 return id;  
 }  
  
 public String getAuthorName() {  
 return authorName;  
 }  
  
 public String getContent() {return content;}  
  
 public int getLikeCount() {  
 return likeCount;  
 }  
  
 public int getShareCount() {  
 return shareCount;  
 }  
  
 public void like(){  
 likeCount++;  
 }  
  
 public Note(){  
 this.id = count;  
 count++;  
 }  
  
 public abstract void share();  
 public abstract void print();  
}

**FavorNote**

package com.company;  
  
public class FavorNote extends Note{  
  
 protected Note source;  
  
 public FavorNote(String author, String content, Note source) {  
 this.authorName = author;  
 this.content = content;  
 this.source = source;  
 }  
  
  
 @Override  
 public void share() {  
  
 }  
  
 @Override  
 public void print(){  
 String stringBuilder = "ID: " + this.id + "\n" +  
 "Автор: 『" + this.getAuthorName() + "』\n" +  
 "Текст:「" + this.getContent() + "」\n" +  
 "Источник: " + "\n" +  
 "ID: " + this.source.id + "\n" +  
 "Автор: 『" + this.source.getAuthorName() + "』\n" +  
 "Текст:「" + this.source.getContent() + "」\n";  
 System.out.println(stringBuilder);  
 }  
}

**AddressNote**

package com.company;  
  
public class AddressNote extends Note{  
  
 protected String address;  
 public AddressNote(String author, String content, String address){  
  
 this.authorName = author;  
 this.content = content;  
 this.address = address;  
 }  
  
 @Override  
 public void share() {  
 shareCount++;  
 }  
  
 @Override  
 public void print(){  
 String stringBuilder = "ID: " + this.id + "\n" +  
 "Автор: 『" + this.getAuthorName() + "』\n" +  
 "Текст:「" + this.getContent() + "」\n" +  
 "Место: " + this.address + "\n" +  
 "Рейтинг: " + this.getLikeCount() + "\n" +  
 "Избранное: " + this.getShareCount() + "\n";  
 System.out.println(stringBuilder);  
 }  
}

**JustNote**

package com.company;  
  
public class JustNote extends Note {  
  
 public JustNote(String author, String content){  
  
 this.authorName = author;  
 this.content = content;  
 }  
   
 @Override  
 public void share() {  
 }  
   
 @Override  
 public void print(){  
 String stringBuilder = "ID: " + this.id + "\n" +  
 "Автор: 『" + this.getAuthorName() + "』\n" +  
 "Текст:「" + this.getContent() + "」\n" +  
 "Рейтинг: " + this.getLikeCount() + "\n" +  
 "Избранное: " + this.getShareCount() + "\n";  
 System.out.println(stringBuilder);  
 }  
}

Список использованных источников

1. Объектно-ориентированное программирование. Метод. указания по

выполнению лабораторных работ / сост.: В.Л. Аршинский. – Иркутск : Изд-во

ИРНИТУ, 2017. – 24 c.